МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної інженерії

# КВАЛІФІКАЦІЙНА РОБОТА

## Пояснювальна записка

другий (магістерський)

(рівень вищої освіти)

Дослідження методів оптимальної фільтрації шумів для розпізнавання зображень

Виконав:
студент 2 курсу групи ІПЗм-21-4
Лавров О. Є.
(прізвище, ініціали)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-наукова
Керівник доц., Работягов А.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. Кафедри _____ З.В. Дудар

2023

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення
(повна назва)

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

# ЗАВДАННЯ

## НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Лаврова Олександра Євгеновича
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження методів оптимальної фільтрації шумів для розпізнавання зображень

затверджена наказом університету від « 29 » березня 2023 р. № 302 Ст

2. Термін подання студентом роботи до екзаменаційної комісії «22» травня 2023 р.

3. Вихідні дані до роботи *встановлений календарний план роботи, методичні вказівки до оформлення пояснювальної записки, методи фільтрації шумів для розпізнавання зображень.*

4. Перелік питань, що потрібно опрацювати в роботі *аналіз предметної галузі, причини виникнення шумів на зображеннях, гістограма розподілу яскравості, огляд фільтрів для зниження шуму, виділення конутурів, аналіз існуючих методів фільтрації шуму, дослідження впливу фільтрів на виділення контурів на зашумленому зображенні та кількісна оцінка шуму.*

# КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної галузі та постановка задачі | 26.01.2023 | виконано |
| 2 | Аналіз існуючих методів | 28.01.2023 | виконано |
| 3 | Дослідження впливу фільтрів, на виділення контурів на зашумлених зображеннях | 15.04.2023 | виконано |
| 4 | Підготовка пояснювальної записки | 26.04.2023 | виконано |
| 5 | Підготовка презентації та доповіді | 11.04.2023 | виконано |
| | Перевірка на академічний плагіат | 12.04.2023 | виконано |
| 6 | Нормоконтроль | 14.05.2023 | виконано |
| 7 | Рецензування | 16.05.2023 | виконано |
| 8 | Занесення диплома в електронний архів | 17.05.2023 | виконано |
| 9 | Попередній захист | 18.05.2023 | виконано |
| 10 | Допуск до захисту | 19.05.2023 | виконано |

Дата видачі завдання ___ _____ 2023 р.

Студент _____
(підпис)

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

## РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 68 с., 24 рис., 1 табл., 11 джерел.

ОПТИМАЛЬНА ФІЛЬТРАЦІЯ ШУМІВ, ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ОПТИМАЛЬНОЇ ФІЛЬТРАЦІЇ ШУМІВ, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, ШУМИ.

Об'єктом дослідження є методи обробки зображень для зменшення впливу шумів на якість розпізнавання зображення.

Метою дослідження є аналіз існуючих методів оптимальної фільтрації шумів для підвищення якості розпізнавання, оцінка їх ефективності, переваг та недоліків в процесі фільтрації шумів на зображенні, зменшення впливу шумів в процесі розпізнавання, визначення оптимальних параметрів фільтрації шумів.

В результаті виконання дослідження проведено аналіз предметної галузі, систематичний огляд методів фільтрації шумів, експерименти щодо визначення оптимальних параметрів фільтрації шумів, порівняння ефективності застосування методів для вирішення практичних задач.

COMPARATIVE ANALYSIS OF OPTIMAL NOISE FILTERING METHODS, IMAGE RECOGNITION, NOISE, OPTIMAL NOISE FILTERING

The object of research is image processing methods to reduce the impact of noise on the quality of image recognition.

The purpose of the study is to analyze the existing methods of optimal noise filtering to improve the quality of recognition, evaluate their effectiveness, advantages and disadvantages in the process of filtering noise on the image, reduce the impact of noise in the process of recognition, and determine the optimal parameters of noise filtering.

As a result of the research, an analysis of the subject area, a systematic review of noise filtering methods, experiments to determine the optimal parameters of noise

filtering, and a comparison of the effectiveness of methods for solving practical problems were carried out.

Я, Лавров Олександр Євгенович, студент гр. ІПЗм-21-4, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів оптимальної фільтрації шумів для розпізнавання зображень», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

# ЗМІСТ

# ВСТУП

Розпізнавання образів – це процес виявлення і класифікації об'єктів на зображенні або відео. Зазвичай цей процес використовується для автоматичного аналізу зображень, наприклад, для виявлення обличчя людини, розпізнавання транспортних засобів на дорогах, розпізнавання рукописних символів тощо.

Розпізнавання образів є процесом, що має на меті перетворення вхідних зображень на семантичні описи об'єктів на цих зображеннях. Ціль розпізнавання може бути різною, включаючи виділення окремих елементів на зображенні або класифікацію всього зображення в цілому. Розпізнавання образів використовується в багатьох галузях, таких як системи розпізнавання тексту, створення тривимірних моделей людини на основі фотографій та інших областях.

Наприклад, розпізнавання образів може бути використано для автоматичної ідентифікації клітин на зображенні мікроскопа, що допомагає лікарям в діагностиці хвороб. Автомобільні компанії використовують розпізнавання образів для автоматичного виявлення знаків дорожнього руху та інших транспортних засобів на дорозі.

Шум на зображеннях – це непотрібні зміни яскравості або колірних значень, які виникають в результаті різних факторів, таких як шум знімальної апаратури, компресія зображень, артефакти обробки зображень та інші. Шум на зображенні може погіршити якість зображення та зробити його непридатним для подальшої обробки та аналізу.

Шуми на зображеннях можуть впливати на якість розпізнавання об'єктів на них. Шуми можуть спотворювати деталі зображення, зміщувати контраст та яскравість, створювати хибні контури та інші артефакти. Це може призвести до неправильної інтерпретації зображення, помилок у розпізнаванні об'єктів та погіршення точності роботи алгоритмів обробки зображень. Для підвищення точності розпізнавання об'єктів на зображеннях необхідно використовувати методи оптимальної фільтрації шумів, які дозволяють зменшити вплив шумів на якість зображення.

Вплив шумів на розпізнавання образів може бути критичним, особливо у завданнях комп'ютерного зору, де точність визначення об'єктів зображення є ключовим чинником для успішного виконання завдання. Шуми на зображенні можуть змінювати колір, контраст, текстуру та інші властивості, що може призвести до спотворення форми та розміру об'єктів, а також до помилкового визначення їх типу та положення.

Для вирішення цієї проблеми застосовуються методи оптимальної фільтрації шумів, які дозволяють зменшити вплив шумів на зображення та підвищити точність розпізнавання об'єктів. Вибір найбільш ефективного методу оптимальної фільтрації шумів може залежати від конкретної задачі та властивостей зображення, тому потрібно проведення досліджень та аналізу різних методів з метою вибору найбільш придатного для конкретного завдання.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

## 1.1 Причини виникнення шумів

Дослідження методів оптимальної фільтрації шумів для розпізнавання зображень – це важлива задача в області обробки зображень. Шуми на фото можуть негативно впливати на розпізнавання образів. Шуми на фото можуть бути спричинені різними факторами, такими як низьке освітлення, знижена якість зйомки, наявність тремтіння камери та іншими артефактами зображення.

Джерела шуму можуть бути різними:

- шум знімальної апаратури – це шум, який виникає в результаті електричних та теплових шумів знімальної апаратури, таких як матриці камер, об'єктиви, фільтри та інші елементи;
- шум освітлення це шум, який виникає в результаті нерівномірного освітлення на знімку, наприклад, при зйомці в слабкому світлі або в нічний час;
- шум компресії це шум, який виникає в результаті стиснення зображення для збереження на диску або передачі по мережі;
- шум обробки зображень це шум, який виникає в результаті обробки зображень, такої як збільшення розміру зображення або покращення різкості;
- шум середовища це шум, який виникає в результаті навколишнього середовища, такого як шум на вулиці або шум на публіці.

Шуми можуть призвести до спотворення кольорів, контурів та текстур об'єктів на фото, що може ускладнити їхнє розпізнавання. Крім того, шуми можуть призвести до помилкових ознак або видалення важливих деталей об'єктів на зображенні [1].

Для покращення розпізнавання об'єктів на фото можна використовувати різні методи обробки зображень, такі як зниження шумів, збільшення контрасту та різкості зображення. Також можна використовувати точніші методи

розпізнавання об'єктів, такі як нейронні мережі, які можуть бути навчені на великій кількості фотографій з різними рівнями шуму.

Метою фільтрації шумів є покращення якості зображення шляхом зменшення рівня шуму на зображенні. Процес фільтрації шумів спрямований на відновлення чіткості та якості зображення, яке було пошкоджене шумами з різних джерел.

Для досягнення мети фільтрації шумів використовуються різні методи та алгоритми, які базуються на різних підходах до обробки зображень. Наприклад, методи фільтрації можуть базуватися на статистичних аналізах зображень, які дозволяють визначити параметри шумів та ефективно їх видалити. Також можуть використовуватися методи, які базуються на матричних операціях, фільтрах частот та інших математичних підходах.

Важливим етапом процесу фільтрації шумів є оцінка якості відфільтрованого зображення. Це дозволяє оцінити ефективність методів фільтрації та порівняти різні підходи до відновлення зображення. Оцінка результату також допомагає визначити, чи вдалося досягнути мети фільтрації шумів та чи є результати задовільними.

На рисунок 1.1 зображено модель процесу спотворення яка передбачає дію деякого спотворюючого оператора $H$ на вхідне зображення $f(x,y)$ і додавання адитивного шуму $n(x,y)$, що дає спотворене зображення $g(x,y)$.

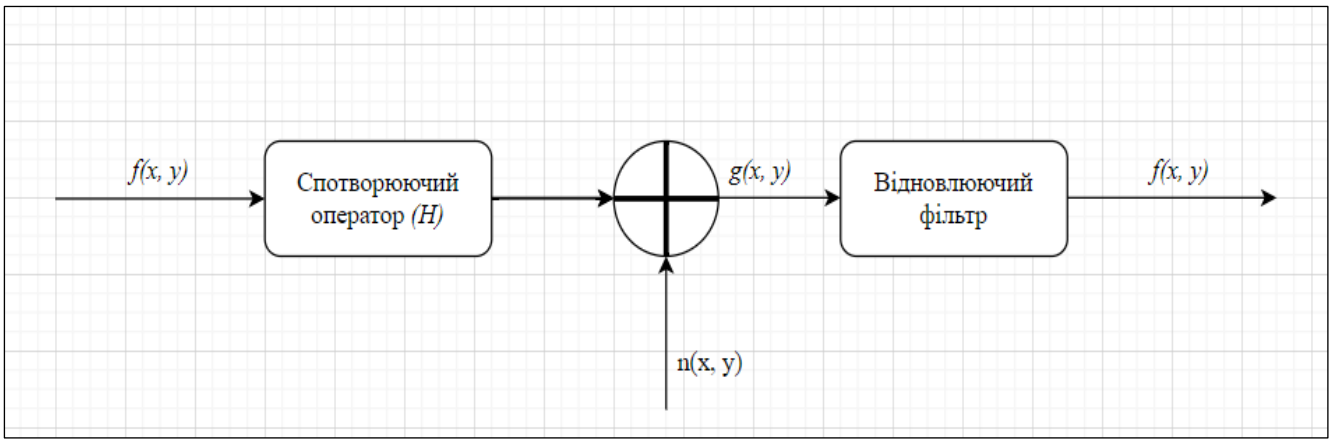


Рисунок 1.1 – Модель процесу спотворення / відновлення зображення

Якщо $H$ – лінійний оператор, то спотворене зображення може бути представлене в просторової області у вигляді (формула 1.1):

$$g(x,y) = h(x,y) * f(x,y) + n(x,y) \quad (1.1)$$

де $h(x,y)$ – функція, що представляє спотворюючий оператор в просторової області (ядро фільтра);

* – символ двовимірної згортки.

У випадку, коли шум на зображенні не залежить від просторових координат і немає кореляції між значеннями шумової складової та значеннями елементів зображення, поведінка шуму базується на статистичних властивостях яскравості компонентів шуму в моделі.

## 1.2 Постановка задачі

Вплив різних методів фільтрації шумів на якість розпізнавання зображень, враховуючи характер шуму та специфічну мету розпізнавання.

В ході виконання роботи буде проведено дослідницький аналіз, порівняльна оцінка, моделювання та експериментальне тестування. В процесі дослідження буде вивчено специфіку шумів, їх вплив на зображення та параметри різних фільтрів для гнучкого використання методів та пошуку оптимальних параметрів фільтрації.

Також в ході виконання роботи буде проведено аналіз, оцінку різних методів фільтрації шумів, зменшення впливу шумів на процес розпізнавання зображень, визначення оптимальних параметрів фільтрації шумів та підтвердження значення та ефективності застосування методів фільтрації шумів для вирішення практичних задач.

У даному дослідженні основними задачами є:

– провести аналіз предметної галузі та систематичний огляд існуючих методів фільтрації шумів, включаючи фільтр Гауса, фільтр Собеля, білатеральний фільтр та фільтр середнього значення;

- оцінити ефективність, переваги та недоліки кожного з цих фільтрів у процесі фільтрації шумів на зображеннях;
- провести експерименти для визначення оптимальних параметрів фільтрації шумів;
- оцінити вплив шумів на процес розпізнавання зображень та спробувати зменшити цей вплив;
- порівняти ефективність застосування різних методів фільтрації шумів для вирішення практичних задач видалення шуму з зображення;
- зробити висновок про важливість та ефективність застосування методів фільтрації шумів у різних практичних галузях.

Результати цього дослідження мають практичне застосування в таких галузях, як комп'ютерний зір, медична діагностика, аерофотозйомка та інші.

### 1.3 Гістограма розподілу яскравості пікселів

Гістограма розподілу яскравості пікселів – це графічне представлення розподілу яскравості пікселів на зображенні.

Гістограма складається з графіка, на якому по осі $X$ відображається значення яскравості пікселів, а по осі $Y$ кількість пікселів з певним значення яскравості.

Гістограму розподілу яскравості пікселів можна використовувати для аналізу та визначення характеристик яскравості зображення: Гістограма може допомогти зрозуміти, які діапазони яскравості присутні на зображенні та наскільки різноманітні ці діапазони. На основі чого можна зробити допущення що до визначення методів фільтрації зображень які необхідно використовувати для покращення зображення. Наприклад, якщо гістограма має вузький діапазон яскравості, то можуть бути застосовані методи контрастування, щоб підвищити контрастність зображення. Якщо ж гістограма має широкий діапазон яскравості, то можуть бути застосовані методи зменшення шуму або динамічного діапазону для поліпшення зображення.

На рисунку 1.2 наведено тестове зображення та гістограму розподілу яскравості для нього.

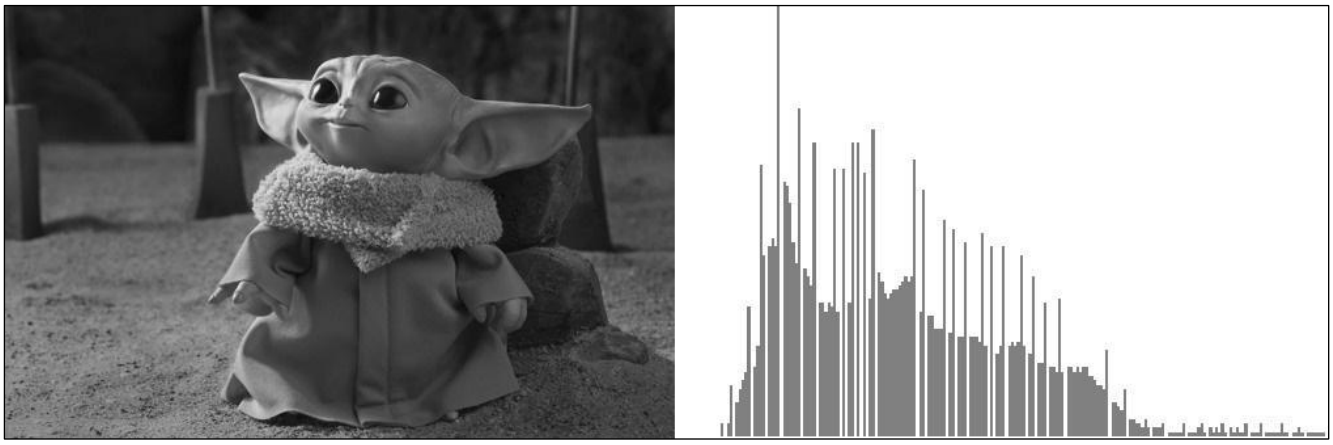

Рисунок 1.2 – Тестове зображення та гістограму розподілу яскравості

Зображення було обране для тестування, оскільки воно містить плавні зміни яскравості, гострі контрасти та регулярні структури.

## 1.4 Гаусівський шум

Гаусівський шум – характеризується нормальним розподілом значень яскравості, при якому значення збігаються з середнім значенням з ймовірністю, яка залежить від дисперсії шуму. Цей тип шуму зазвичай виникає через електронний шум в матриці зображення або через недостатнє освітлення, і описується такою функцією щільності розподілу амплітуд (формула 1.2):

$$p(z) = \frac{1}{\sqrt{\pi D}} e^{-\frac{(z-\eta)^2}{2D}} \tag{1.2}$$

де $z$ являє собою значення випадкової величини (в разі зображення яскравість),

$\mu$ – середнє значення випадкової величини,

$D$ – дисперсія випадкової величини.

Середнє значення гаусового розподілу відповідає його центру. У контексті додавання шуму до зображення середнє значення визначає середній рівень шуму,

який буде додано до кожного пікселя. Наприклад, якщо середнє значення дорівнює 0, шум буде мати середнє значення 0, і він не буде мати систематичного зміщення.

Дисперсія гаусового розподілу визначає його "ширину" або розкид значень навколо середнього значення. Чим більше стандартне відхилення, тим більший розкид значень шуму буде додано до пікселів зображення, і тим сильніше буде видимий шум на зображенні [2].

На рисунку 1.3 наведено приклад зашумленого зображення гаусівським шумом з $\mu=0, D=125,$ та гістограма яскравості пікселей.

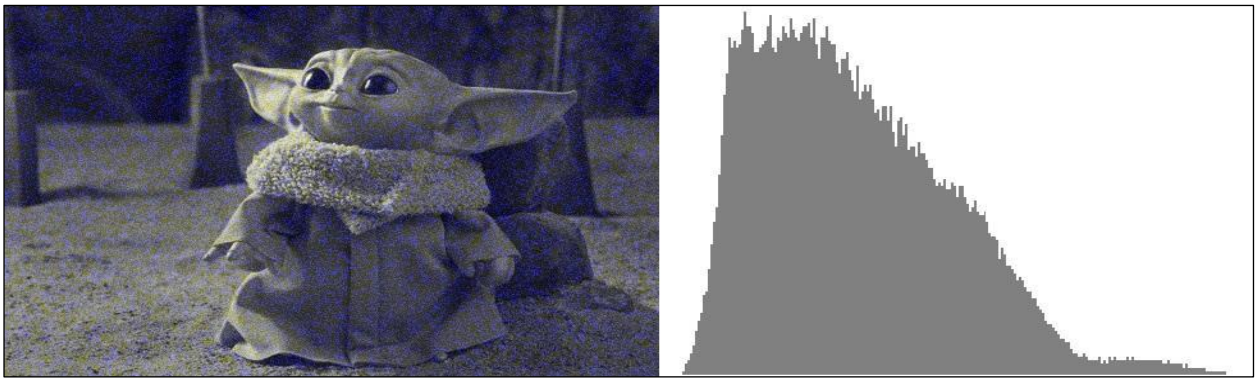

Рисунок 1.3 – Гаусівський шум та гістограма

Якщо гістограма зображення відображає рівномірний розподіл точок по всьому діапазону яскравості, то це може свідчити про наявність гаусівського шуму. Однак, щоб бути впевненим в наявності гаусівського шуму, потрібно провести додаткові тести та аналізи зображення. На рисунку 1.4 наведено порівняння гістограм вихідного зображення та зображення зашумленого гаусівським шумом.

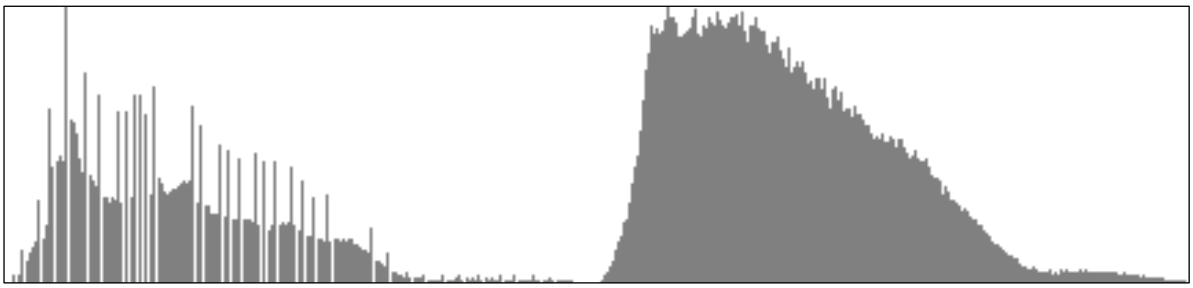

Рисунок 1.4 – Гістограма вихідного зображення та зашумленого гаусівським шумом.

Для фільтрації гаусівського шуму можна використовувати різні фільтри. Одним з найпоширеніших фільтрів є фільтр Гауса, який згладжує зображення, зменшуючи шум і зберігаючи різність між яскравими та темними областями зображення. Інші фільтри, які можна використовувати для фільтрації гаусівського шуму, включають фільтри медіанного рівняння та фільтри зі зменшенням шуму на основі вейвлет–перетворення.

## 1.5 Рівномірний шум

Рівномірний шум – характеризується рівномірним розподілом значень яскравості, коли кожне значення має однакову ймовірність випадати від мінімального до максимального значення. Цей тип шуму може виникати при високій чутливості матриці зображення. На зображенні 1.5 наведено приклад рівномірного шуму з амплітудою ±50, та його гістограму.

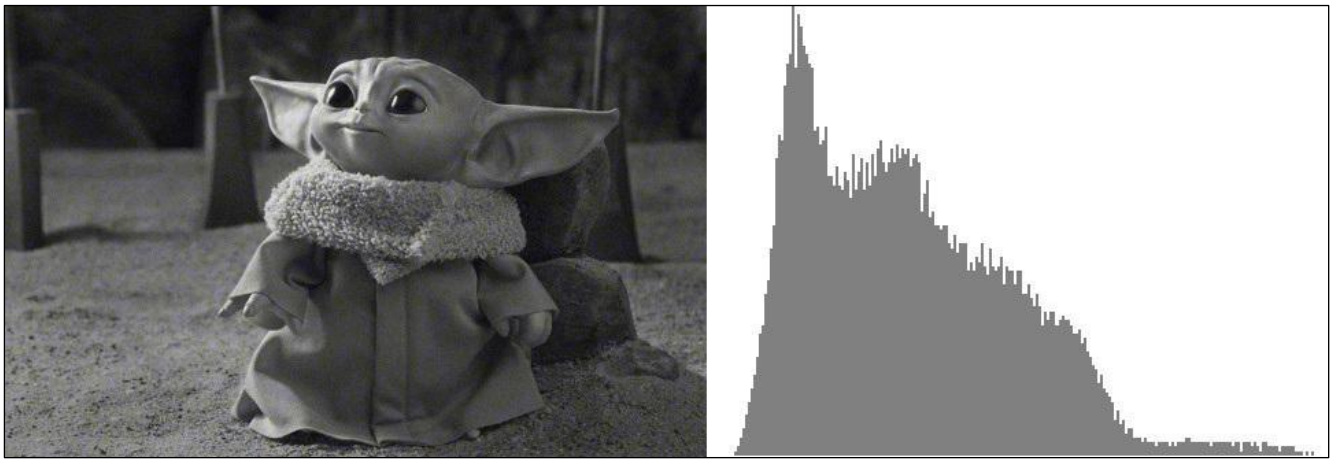

Рисунок 1.5 – Рівномірний шум з амплітудою ±50

Якщо додати рівномірний шум до зображення, гістограма яскравості покаже рівномірне збільшення кількості пікселів на всіх рівнях яскравості в межах діапазону шуму. Зауважте, що гістограма зображення з рівномірним шумом не буде ідеально рівномірною, оскільки вихідне зображення має власний розподіл яскравості.

## 1.6 Імпульсний шум

Імпульсний шум (сіль і перець) – це тип шуму, який виникає в результаті дефектів матриці зображення або проблем зі зберіганням файлу і відображається на зображенні як випадкові білі і чорні пікселі. На рисунку 1.6 зображено приклад імпульсного шуму "сіль і перець" з частотою 10%. Функція розподілення для такого шуму, для 8–бітового зображення, виглядає наступним чином (формула 1.3):

$$f(x) = a, \text{якщо } x \neq b, \qquad (1.3)$$
$$f(x) = x, \text{якщо } x \neq b$$

де $a$ – значення яскравості білого (255) або чорного,

$b$ – ймовірність того, що даний піксель буде білим або чорним.

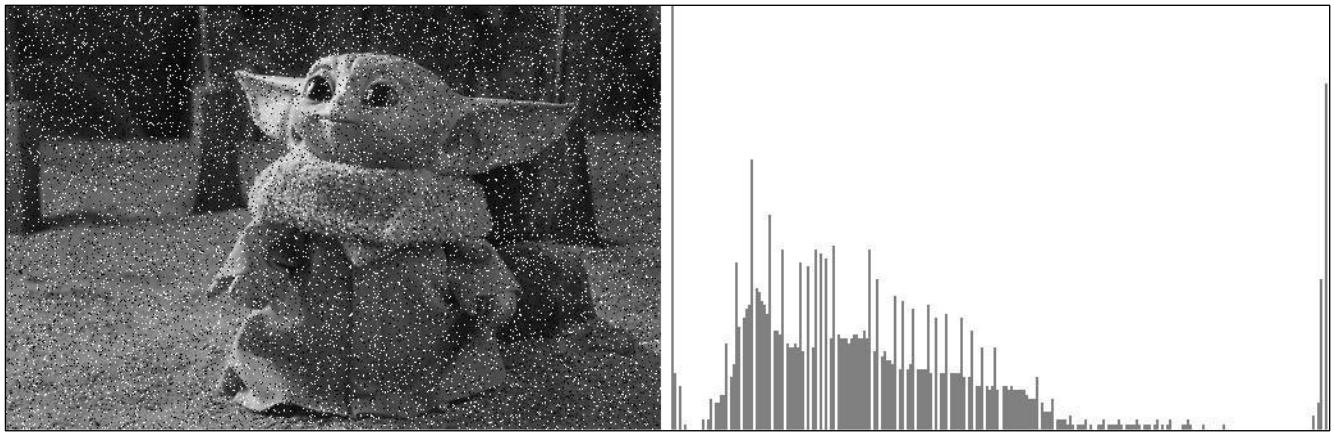

Рисунок 1.6 – Приклад імпульсного шуму "сіль і перець" з частотою 10%

Коли шум "сіль і перець" додається до зображення, на гістограмі яскравості можна помітити збільшення висоти стовпців, що відповідають чорному та білому кольорам. Це зумовлено випадковим розподілом білих та чорних пікселів на зображенні.

Окрім наведених вище шумів існує безліч різновидів.

## 1.7 Огляд фільтрів для зниження рівня шуму

У контексті розпізнавання образів, фільтри для фільтрації шумів мають на меті зменшити шум та збереження  корисної інформацію для подальшої обробки та аналізу. Для цього можуть використовуватися наступні типи фільтрів:

- фільтри для зниження шумів;
- фільтри для збільшення контрасту;
- фільтри для збільшення різкості зображення;
- фільтри для виділення контурів зображень.

Критеріями оцінки роботи використання фільтру або декількох фільтрів можуть бути наступні:

- ефективність;
- якість зображення;
- час виконання;
- простота реалізації;
- ступінь фільтрації шуму.

У магістерській роботі проведені експериментальні дослідження, які виявили, що незважаючи на незначну різницю в сукупній яскравості, зображення, спотворені гаусовим або рівномірним шумом, візуально майже не відрізняються. Хоча відмінності в їх гістограмах досить великі. Натомість, зображення, спотворені імпульсним шумом, відчутно відрізняються, що дає змогу визначити тип шуму, який спричинив спотворення.

### 1.7.1 Фільтр середнього значення

Фільтр середнього значення є одним з найпростіших методів фільтрації шумів, який широко використовується для поліпшення якості зображень. Його основна ідея полягає в обчисленні середнього значення інтенсивності пікселів у вікні певного розміру, яке переміщується по всьому зображенню. Значення інтенсивності кожного пікселя замінюється на середнє значення його околиці, рисунок 1.7.

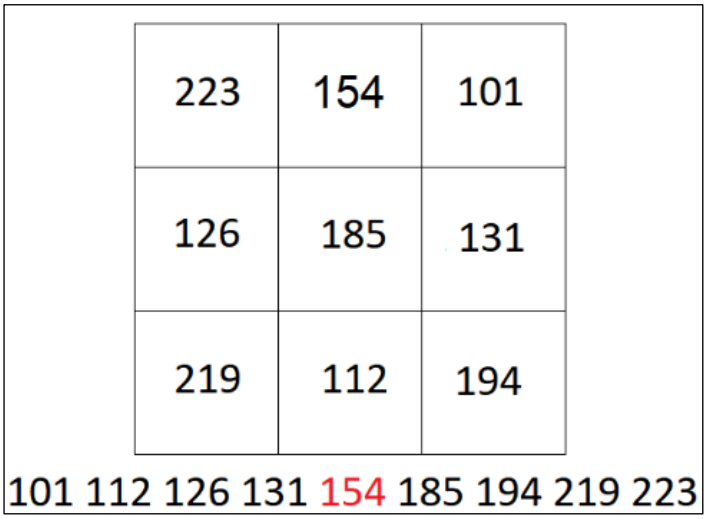


Рисунок 1.7 – Розрахунок медіани для вікна 3x3

Цей метод працює досить добре для випадків, коли шум є адитивним з гаусовою структурою. Однак, для зображень з іншими типами шумів, такими як сіль і перець або дрібний шум, фільтр середнього значення може дати незадовільний результат, тому що він може призвести до втрати деталей об'єктів на зображенні.

Фільтр середнього значення можна описати так (формула 1.4):

$$G(x,y) = \frac{1}{mn} * \Sigma i = 0..M-1 \, \Sigma j = 0..N-1 \, f(x+i, y+j)) \tag{1.4}$$

де G(x,y) – нове значення яскравості пікселя на позиції (x,y),

$f(x+i,y+j)$ – значення яскравості пікселя у околі пікселя (x,y),

$M$ та $N$ – ширина та висота вікна, що використовують для вирахування середнього значення.

Фільтр середнього значення є досить простим та ефективним методом для фільтрації шумів у зображеннях, але його ефективність залежить від типу шуму та параметрів фільтра. Тому, для досягнення оптимального результату, потрібно проводити додаткові дослідження та порівнювати результати різних методів фільтрації шумів.

### 1.7.2 Фільтр Гауса

Фільтр Гауса – це лінійний фільтр згладжування, який використовує гаусівську функцію як його імпульсну характеристику. Фільтр Гауса часто використовується для зменшення шуму та згладжування зображень у комп'ютерному зорі та обробці зображень. Він ефективно зберігає краї та деталі зображення, але видаляє шум.

Суть фільтра Гауса полягає у пакунку зображення з ядром Гаусса. Ядро Гаусса – це двовимірна функція, яка використовується для обчислення ваги, що застосовуються до кожного пікселя в процесі згортки [3].

Ядро Гауса визначається такою формулою 1.5:

$$G(x, y) = (1 / (2 * \pi * \sigma\text{^}2)) * e\text{^}(- (x\text{^}2 + y\text{^}2) / (2 * \sigma\text{^}2)) \quad (1.5)$$

де $x$ і $y$ – відстані від центру ядра,

σ – параметр сигма, який визначає ступінь розмиття.

Для застосування фільтра Гауса до зображення ядро Гауса застосовується шляхом проходу по кожному пікселю зображення та обчислення нового значення для кожного пікселя на основі його сусідів та ваг, визначених ядром.

В результаті застосування фільтра Гауса зображення стає більш розмитим і шуми на ньому видаляються. Однак, при надто великому значенні параметра сигма зображення може стати занадто розмитим і втратити деякі деталі.

### 1.7.3 Білатеральний фільтр

Цей фільтр використовує дві функції схожості – одну для інтенсивності пікселів та іншу для геометричної схожості пікселів – для зменшення шумів і збереження різкості меж об'єктів на зображенні.

Білатеральний фільтр зберігає структуру зображення, що дозволяє підвищити якість розпізнавання об'єктів на зображенні. Він заснований на локальній обробці зображення, що означає, що він може бути застосований до різних типів зображень та шумів.

Принцип роботи білатерального фільтру полягає в тому, що він розглядає околи кожного пікселя та враховує його інтенсивність та геометричну схожість з іншими пікселями. Інтенсивність пікселів використовується для обчислення вагового коефіцієнту, який визначає вплив кожного пікселя на результат фільтрації. Геометрична схожість використовується для визначення радіусу околу кожного пікселя, що використовується для обчислення вагових коефіцієнтів.

У результаті застосування білатерального фільтру до зображення, шуми зменшуються, а різкість меж об'єктів зберігається.

Математичний опис білатерального фільтру включає дві гаусівські функції: просторову гаусівську функцію (для урахування просторового розташування пікселів) та доменну гаусівську функцію (для урахування різниці інтенсивності пікселів).

Просторова гаусівська функція має такий вигляд (формула 1.6):

$$Gs(x, y) = e\text{^}(-(x\text{^}2 + y\text{^}2) / (2 * \sigma s\text{^}2)) \quad (1.6)$$

де Gs(x, y) – просторова гаусівська функція,

*x* та *y* – координати пікселя,

σs – стандартне відхилення для просторової гаусівської функції.

Доменна гаусівська функція має наступний вигляд (формула 1.7):

$$Gd(I1, I2) = e\text{^}(-((I1 - I2)\text{^}2) / (2 * \sigma d\text{^}2)) \quad (1.7)$$

де *Gd(I1, I2)* – доменна гаусівська функція,

*I1* та *I2* – інтенсивності сусідніх пікселів,

*σd* – стандартне відхилення для доменної гаусівської функції.

Білатеральний фільтр можна описати наступним математичним виразом (формула 1.8):

$$BF(I)(x, y) = (1 / W) * \Sigma(I(x', y') * Gs(x - x', y - y') * Gd(I(x, y), I(x', y'))) \quad (1.8)$$

де *BF(I)(x, y)* – результат білатерального фільтра для пікселя (x, y),

*W* – нормалізуючий коефіцієнт.

### 1.8 Фільтр Собеля для виділення контурів

Фільтр Собеля – це лінійний фільтр, який використовується для виявлення границь в зображеннях. Він працює шляхом обчислення приблизного значення градієнта інтенсивності зображення в кожному пікселі. Фільтр Собеля використовує два 3x3 ядра, одне для обчислення вертикальних границь (Sx) та інше для обчислення горизонтальних границь (Sy).

Ядра Собеля виглядають наступним чином (формула 1.9):

$$Sx = [[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], \quad (1.9)$$
$$Sy = [[-1, -2, -1], [0, 0, 0], [1, 2, 1]]$$

де *Sx* використовується для виявлення вертикальних границь (зміни інтенсивності зліва направо),

*Sy* використовується для виявлення горизонтальних границь (зміни інтенсивності зверху вниз).

Обидва ядра застосовуються до зображення шляхом операції згортки, що дозволяє отримати два нових зображення: одне для вертикальних границь та інше для горизонтальних границь. Значення градієнта в кожному пікселі можна обчислити як корінь квадратний з суми квадратів значень вертикального та горизонтального градієнтів (формула 1.10):

$$G = sqrt((Sx * I)\text{^}2 + (Sy * I)\text{^}2) \quad (1.10)$$

де *G* – значення градієнта,

*I* – зображення,

*"*"* вказує на операцію згортки.

Напрямок градієнта також можна обчислити як арктангенс відношення горизонтального градієнта до вертикального (формула 1.11):

$$\theta = atan2((Sy * I), (Sx * I)) \quad (1.11)$$

де $\theta$ – напрямок градієнта,

*atan2* – функція арктангенса з двох аргументів.

Фільтр Собеля широко використовується в обробці зображень та комп'ютерному зорі для виявлення країв зображень та інших завдань, таких як сегментація, аналіз текстур та виявлення особливостей [4].

Одним з головних переваг фільтра Собеля є його простота та ефективність. Оскільки він використовує лише маленькі ядра, обчислення можуть бути здійснені досить швидко, що робить цей фільтр підходящим для реального часу та великих зображень.

# 2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

## 2.1 Фільтрація шуму усередненими фільтрами

Для деяких усереднюючих фільтрів можна використовувати операцію згортки, щоб застосувати їх до оброблюваного зображення. У цьому випадку, фільтр можна представити у вигляді матриці або маски, де кожен елемент має вагу. При застосуванні фільтра до зображення, ця маска згортається з кожним пікселем зображення, щоб отримати нове значення пікселя. Ваги елементів маски допомагають змінювати значення пікселя залежно від його оточення, що може змінювати його яскравість або контрастність, зменшувати шум або підсилювати деякі особливості зображення. У результаті, зображення стає плавнішим або виразнішим, в залежності від того, який фільтр був застосований (формула 2.1.

$$z'(x,y) = \frac{1}{mn}\sum_{(s,t)\in S_{xy}} h(s,t)z(s,t) \quad (2.1)$$

де $z(s,t)$ та $z'(x,y)$ яскравості пікселів з координатами $(s,t)$ та (x, y) відповідно спотвореного і відновленого зображень,

$h(s,t)$ – вагові коефіцієнти матриці ядра фільтра,

$S_{xy}$ – область завдання ядра фільтра або його апертура (центр апертури $S_{xy}$ співпадає з пікселем $z'(x,y)$ спотвореного зображення) [5].

Так, у більшості випадків апертури мають симетричну форму, таку як квадрат або круг, що допомагає забезпечити однакову обробку кожного пікселя зображення. Часто, ваги елементів апертури мають цілочисленні значення, що дозволяє більш ефективно виконувати операції згортки на практиці.

При застосуванні фільтрів до зображення, результат згортки може бути дуже великим числом, що не відповідає діапазону значень яскравості пікселів на зображенні. Щоб уникнути цього, результат згортки при необхідності множаться на нормуючий множник, який дозволяє зберігати без змін області з постійною яскравістю на зображенні. Нормуючий множник може залежати від розміру апертури та ваг елементів апертури [6].

Так, середньоарифметичний фільтр є одним з найпростіших серед усереднюючих фільтрів. Він використовується для зменшення шуму на зображенні шляхом усереднення яскравості пікселів в околі кожного пікселя.

На рисунку 2.1 наведено приклад фільтрації гаусівського шуму фільтром середнього значення.

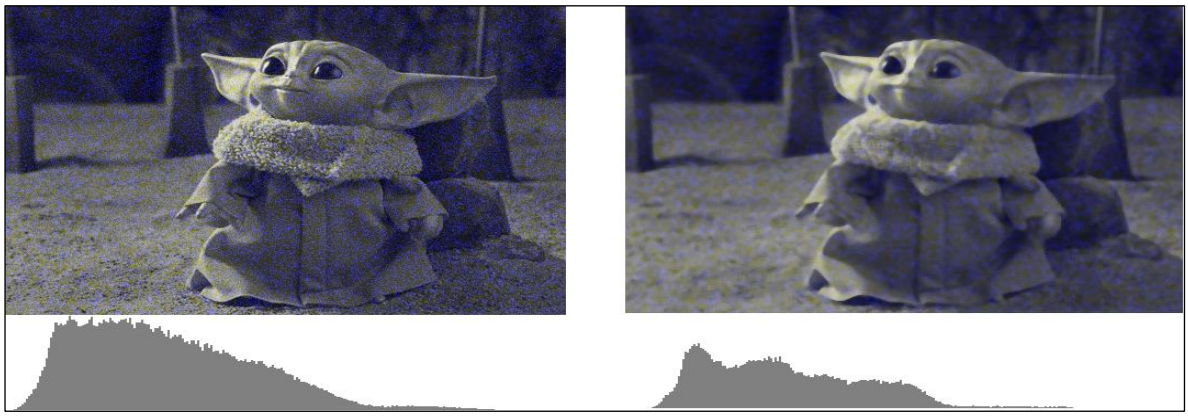

Рисунок 2.1 – Зображення з гаусівським шумом до фільтрації фільтром середнього значення та після.

Візуально можна побачити що зображення кількість шуму стала менша але зображення стало трохи розмитим. На гістограмі яскравості помітно що розподіл став більш рівномірний.

На рисунку 2.2 наведено результат роботи фільтра середнього значення з імпульсним шумом.

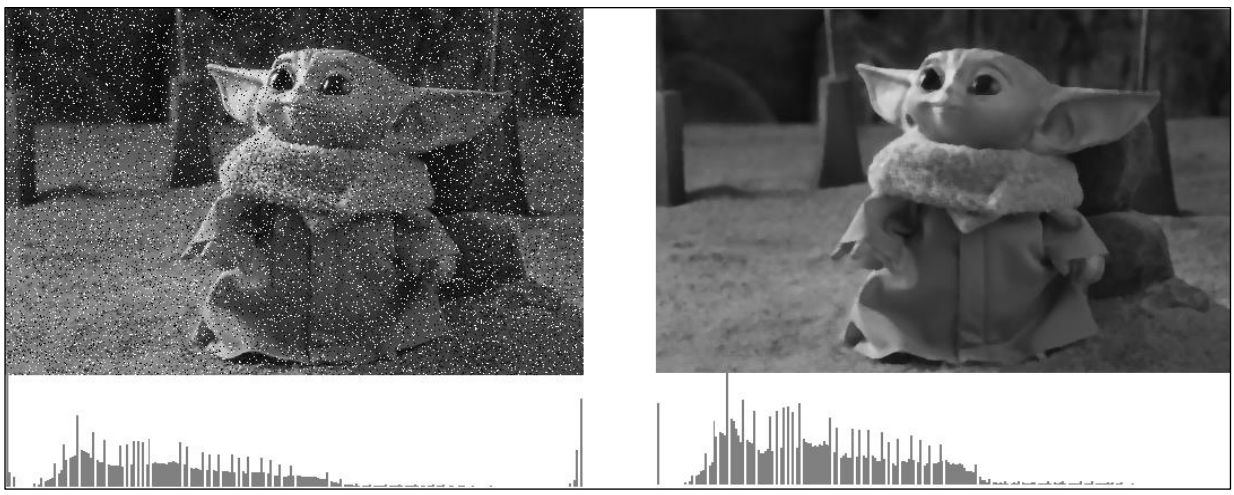

Рисунок 2.2 – Зображення з шумом сіль та перець до фільтрації фільтром середнього значення та після

При візуальному порівнянні зашумленого і відфільтрованого зображення можна побачити, що фільтр середнього значення добре працює з імпульсним шумом, шум повністю зник з зображення, контури зображення залишилися досить чіткими.

На гістограмі яскравості помітно шо піки по краях зникли, графік більше походить на графік вихідного не зашумленого зображення.

На рисунку 2.3 результат роботи фільтра середнього значення з рівномірним шумом.

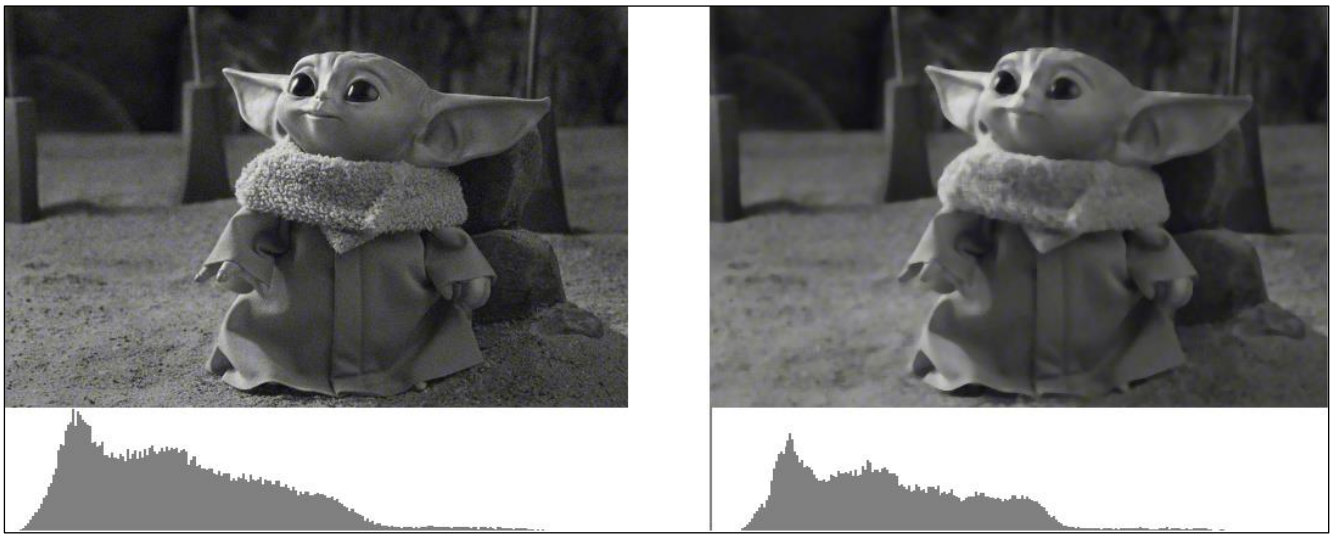

Рисунок 2.3 – Зображення з рівномірним шумом до фільтрації фільтром середнього значення та після

Хоча візуально різниця між відфільтрованим зображення не дуже помітна, але на гістограмі можна побачити що розподіл став більш плавим.

До усіх зображень застосовувався фільтр з розміром матриці 5x5. Виходячи з візуальних спостережень можна зробити висновок що даний фільтр найкраще підходить для роботи з імпульсним шумом. Результат роботи відображається на гістограмі.

На рисунку 2.4 наведено вихідне зображення та результати застосування фільтра середнього значення, до різних видів шумів (вихідне зображення, гаусівський, рівномірний, імпульсний).

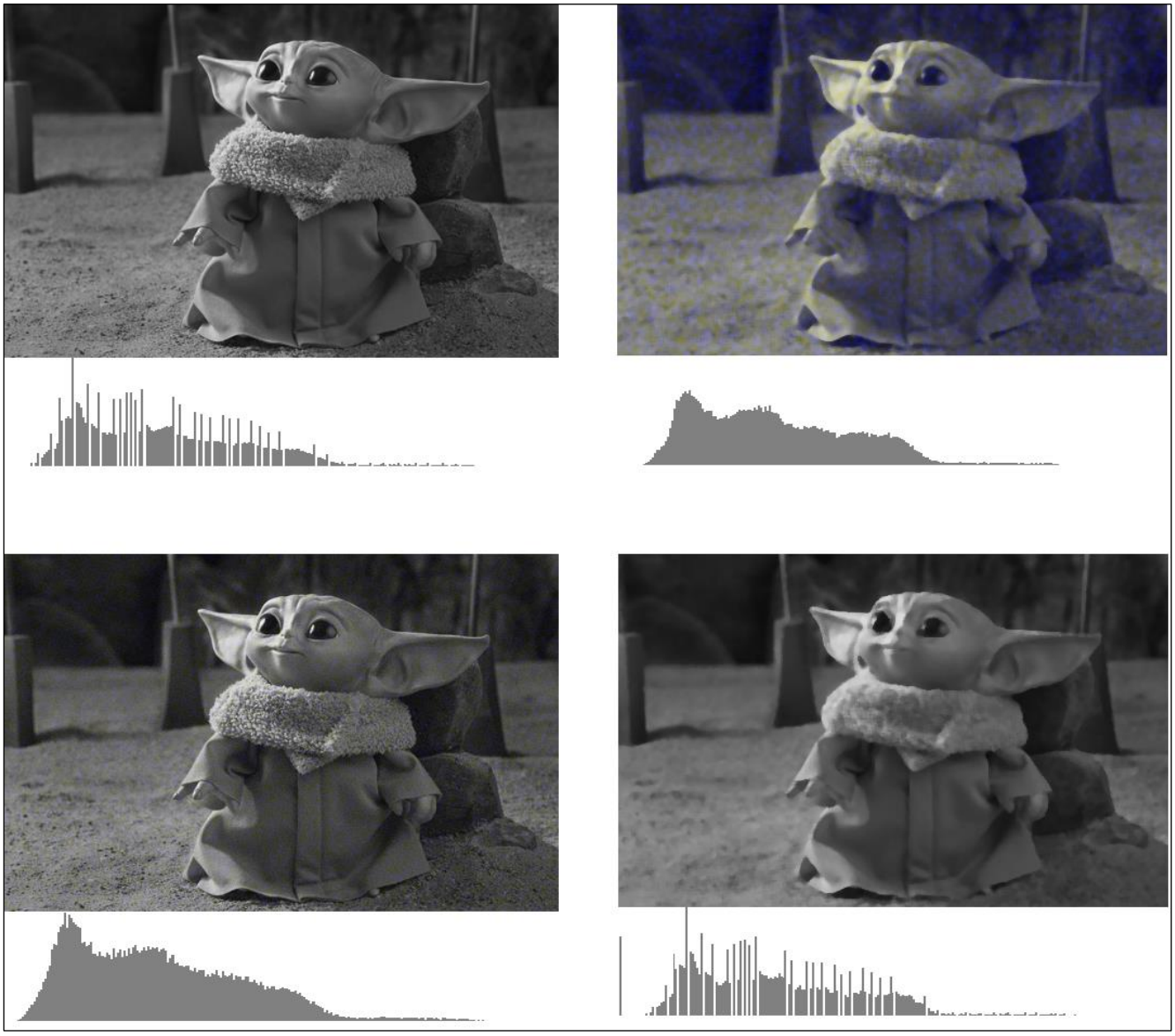

Рисунок 2.4 – Порівняння вихідного зображення та відфільтрованих зображень ,за допомогою фільтром середнього значення, з гаусівським, рівномірним та імпульсним шумах відповідно.

Длі наведено приклад реалізації фільтра середнього значення, на мові програмування c#, метод на вході отримує зображення та розмір вікна самого фільтра. Далі алгоритм буде виглядати наступним чином:

- розділити вхідне зображення на три окремі канали кольору: червоний, зелений та синій;
- для кожного каналу застосувати фільтр середнього значення обчисливши медіану значень пікселів у вікні фільтрації;

– об'єднати відфільтровані канали в одне кольорове зображення.

```
public static Bitmap MedianFilter(Bitmap image, int matrixSize)
        {
            Bitmap result = new Bitmap(image.Width, image.Height);
            int radius = matrixSize / 2;

            for (int i = radius; i < image.Width – radius; i++)
            {
                for (int j = radius; j < image.Height – radius; j++)
                {
                    List<int> rList = new List<int>();
                    List<int> gList = new List<int>();
                    List<int> bList = new List<int>();

                    for (int k = –radius; k <= radius; k++)
                    {
                        for (int l = –radius; l <= radius; l++)
                        {
                        Color pixel = image.GetPixel(i + k, j + l);
                            rList.Add(pixel.R);
                            gList.Add(pixel.G);
                            bList.Add(pixel.B);
                        }
                    }

                    rList.Sort();
                    gList.Sort();
                    bList.Sort();

                    int medianIndex = (rList.Count – 1) / 2;

                    Color                    newPixel                    =
Color.FromArgb(rList[medianIndex], gList[medianIndex], bList[medianIndex]);
                    result.SetPixel(i, j, newPixel);
                }
            }

            return result;
    }
```

Таким чином, для кожного пікселя зображення значення кожного каналу замінюється на медіанне значення з вибраних пікселів в околиці.

## 2.2 Фільтрація шуму фільтром Гауса

На рисунку 2.5 наведено приклад фільтрації гфусівського шуму фільтром Гауса.

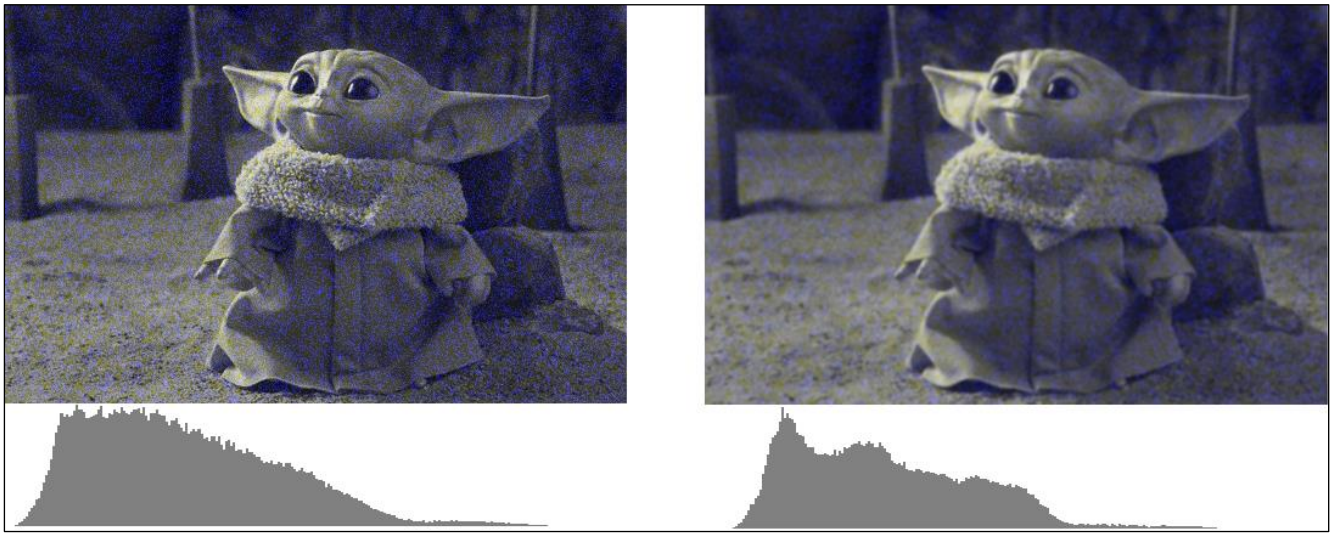

Рисунок 2.5 – Зображення з гаусівським шумом до фільтрації фільтром Гауса та після.

При візуальному порівнянні зашумленого і відфільтрованого зображення можна побачити, що фільтр Гауса добре  працює з імпульсним шумом, шум став менш помітним, хоча зображення стало трохи розмитим.

На рисунку 2.6 наведено результат роботи фільтра Гауса з імпульсним шумом та гістограми розподілу яскравості пікселів на зашумленому та відфільтрованому зображеннях відповідно.

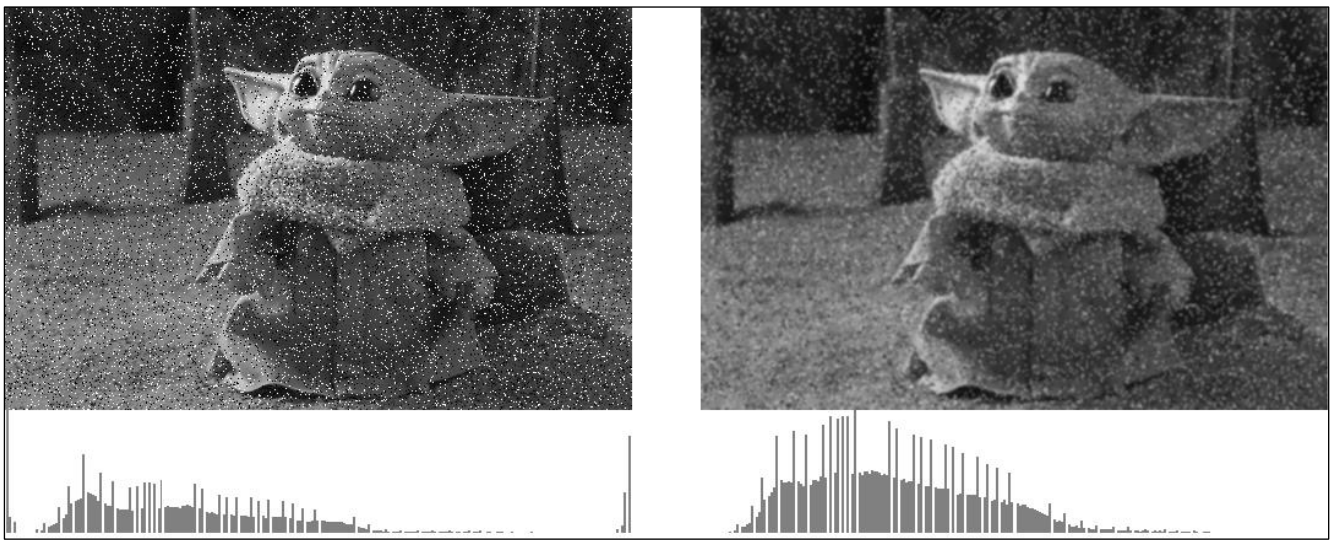

Рисунок 2.6 – Зображення з імпульсним шумом до фільтрації, фільтром Гауса та після.

При візуальному порівнянні зашумленого і відфільтрованого зображення можна побачити, що фільтр Гауса не дуже добре справився з імпульсним шумом, шум все ще яскраво виражений. На гістограмі можна побачити що розподіл став менш рівномірним.

На рисунку 2.7 наведено результат роботи фільтра Гауса з рівномірним шумом.

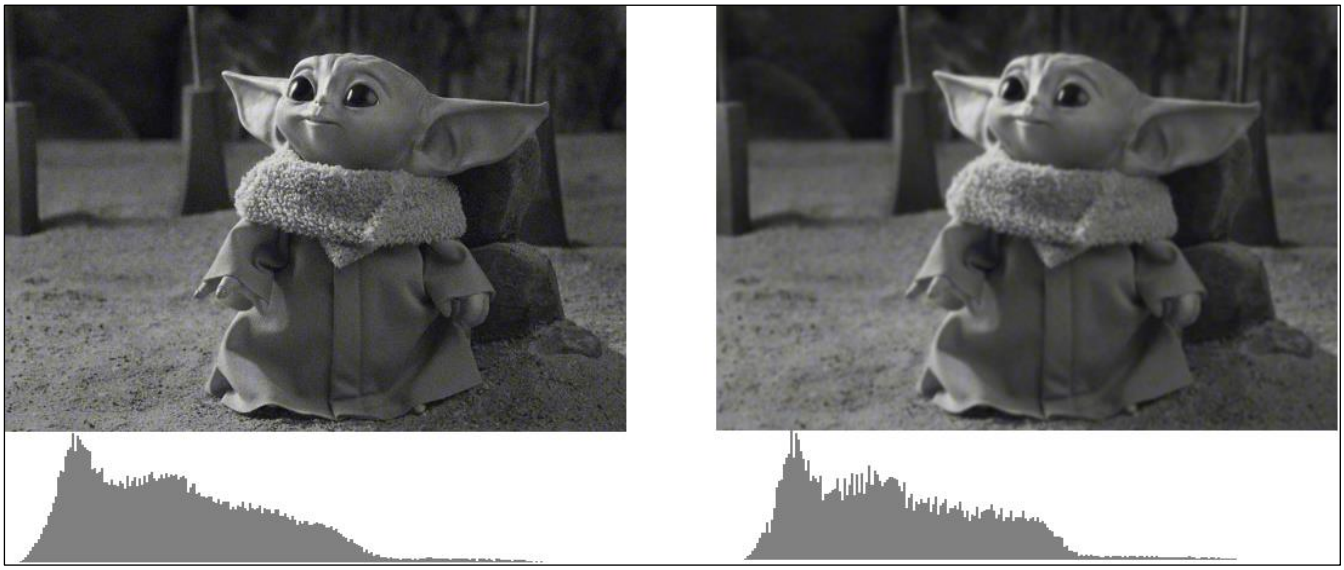

Рисунок 2.7 – Зображення з рівномірним шумом до фільтрації, фільтром Гауса та після.

На рисунку 2.7 можна побачити, що відфільтроване зображення і вихідне майже не відрізняються. Гістограми схожі.

До зображень застосовувався фільтр Гауса з розміром ядра 5x5 на коефіцієнтом розмиття, сігма 0.

На рисунку 2.8 наведено вихідне зображення та результати застосування фільтра Гауса, до різних видів шумів (вихідне зображення, гаусівський, рівномірний, імпульсний), також наведено гістограми розподілу яскравості для кожного з них.

Далі наведено приклад реалізації фільтра Гауса, на мові програмування c#, метод на вході отримує зображення, розмір вікна самого фільтра та сігму який відповідає за коефіцієнт розмиття.

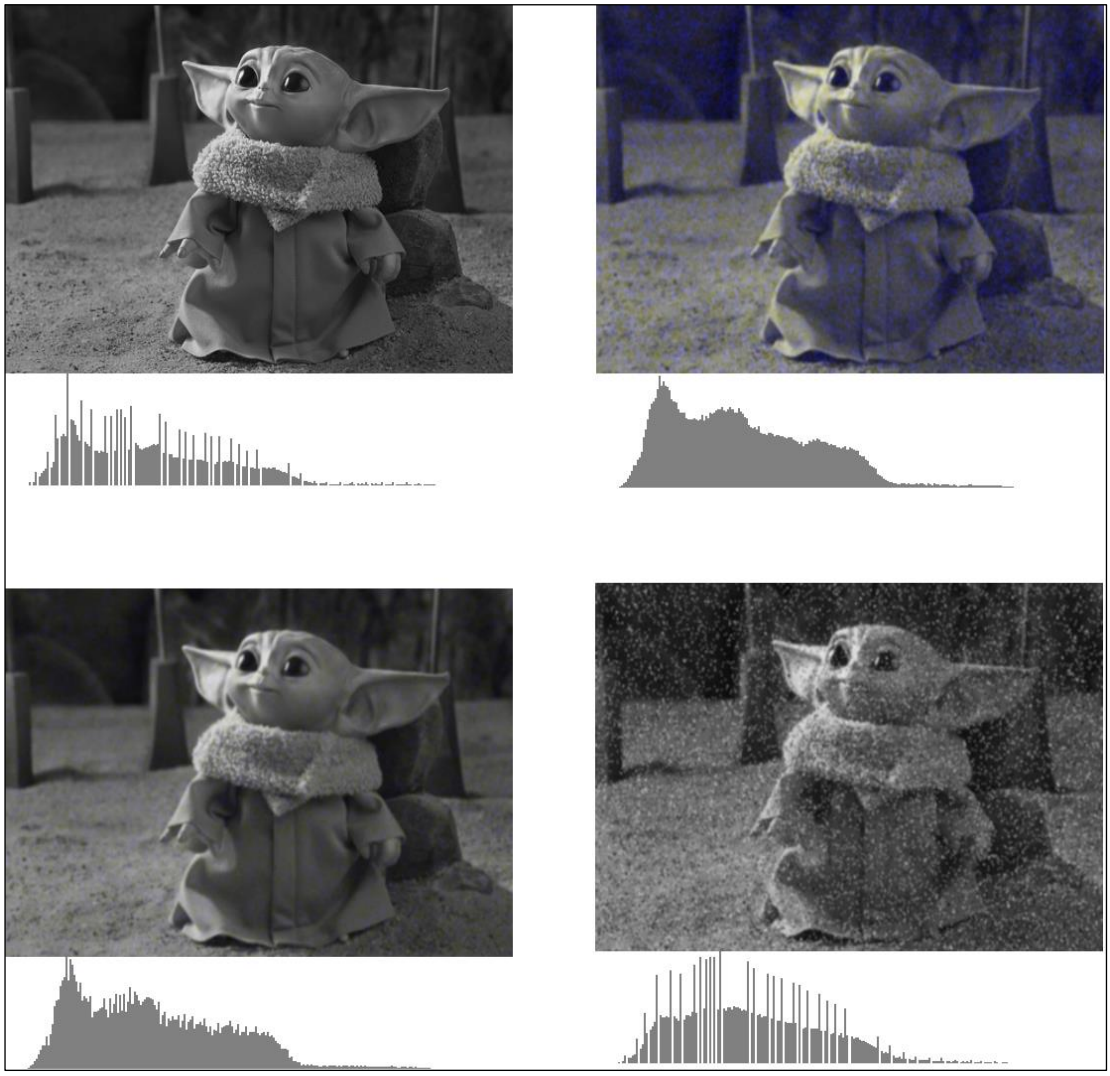

Рисунок 2.8 – Порівняння вихідного зображення та відфільтрованих зображень, за допомогою фільтра Гауса, з гаусівським, рівномірним та імпульсним шумах відповідно.

Далі алгоритм буде виглядати наступним чином:

- створити ядро Гауса, яке використовуватиметься для фільтрації. Розмір ядра залежить від рівня розмиття;
- обчислити вагу кожного пікселя на околиці, використовуючи значення ядра Гауса. Для кожного пікселя на околиці помножити його значення на відповідну вагу з ядра Гауса;
- скласти зважені значення пікселів на околиці, щоб отримати нове значення для поточного пікселя;
- продовжити процес для всіх пікселів у зображенні;

– повторіть процедуру для всіх каналів;

– об'єднати відфільтровані канали в одне кольорове зображення.

```
    public static Bitmap GaussianFilter(Bitmap image, int kernelSize, double sigma)
            {
                int width = image.Width;
                int height = image.Height;
                double[,] kernel = GaussianKernel(kernelSize, sigma);

                Bitmap output = new Bitmap(width, height, PixelFormat.Format24bppRgb);

                BitmapData srcData = image.LockBits(new Rectangle(0, 0, width, height),
                                                    ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
                BitmapData dstData = output.LockBits(new Rectangle(0, 0, width, height),
                                                    ImageLockMode.WriteOnly, PixelFormat.Format24bppRgb);

                int stride = srcData.Stride;
                int bytesPerPixel = 3;
                int kernelRadius = kernelSize / 2;

                unsafe
                {
                    byte* srcPtr = (byte*)srcData.Scan0;
                    byte* dstPtr = (byte*)dstData.Scan0;

                    for (int y = 0; y < height; y++)
                    {
                        for (int x = 0; x < width; x++)
                        {
                            double bSum = 0, gSum = 0, rSum = 0, weightSum = 0;

                            for (int ky = -kernelRadius; ky <= kernelRadius; ky++)
                            {
                                for (int kx = -kernelRadius; kx <= kernelRadius; kx++)
                                {
                                    int px = x + kx;
                                    int py = y + ky;

                                    if (px >= 0 && px < width && py >= 0 && py < height)
                                    {
                                        byte* pixelPtr = srcPtr + py * stride + px * bytesPerPixel;
                                        double weight = kernel[kx + kernelRadius, ky + kernelRadius];
```

```
                                    bSum += pixelPtr[0] * weight;
                                    gSum += pixelPtr[1] * weight;
                                    rSum += pixelPtr[2] * weight;

                                    weightSum += weight;
                                }
                            }
                        }

                        byte* outputPtr = dstPtr + y * stride + x *
bytesPerPixel;

                        outputPtr[0] = (byte)(bSum / weightSum);
                        outputPtr[1] = (byte)(gSum / weightSum);
                        outputPtr[2] = (byte)(rSum / weightSum);
                    }
                }
            }

            image.UnlockBits(srcData);
            output.UnlockBits(dstData);

            return output;
        }
```

В результаті застосування фільтра Гауса зображення буде розмите, що може згладити шуми та покращити якість зображення. Однак, сильне розмиття може призвести до втрати деталей, тому необхідно підбирати оптимальний розмір ядра та ступінь розмиття.

## 2.3 Фільтрація шуму білатеральним фільтром

На рисунку 2.9 наведено приклад фільтрації гфусівського шуму білатеральним фільтром та гістограми розподілу яскравості пікселів на зашумленому та відфільтрованому зображеннях відповідно.

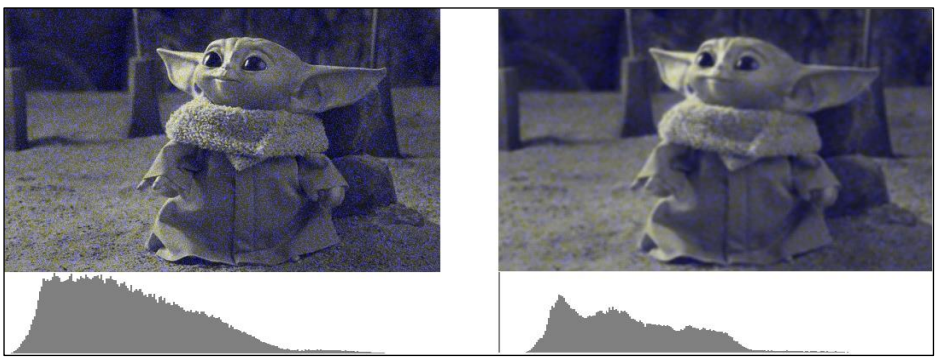

Рисунок 2.9 – Зображення з гаусівським шумом до фільтрації білатеральним фільтром та після.

На рисунку 2.9 можна побачити, що білатеральний фільтр видалив більшу частину шуму, але зробив зображення більш розмитим. Розподіл яскравості став більш плавним.

На рисунку 2.10 наведено результат роботи білатерального фільтра з імпульсним шумом та гістограми розподілу яскравості пікселів на зашумленому та відфільтрованому зображеннях відповідно.

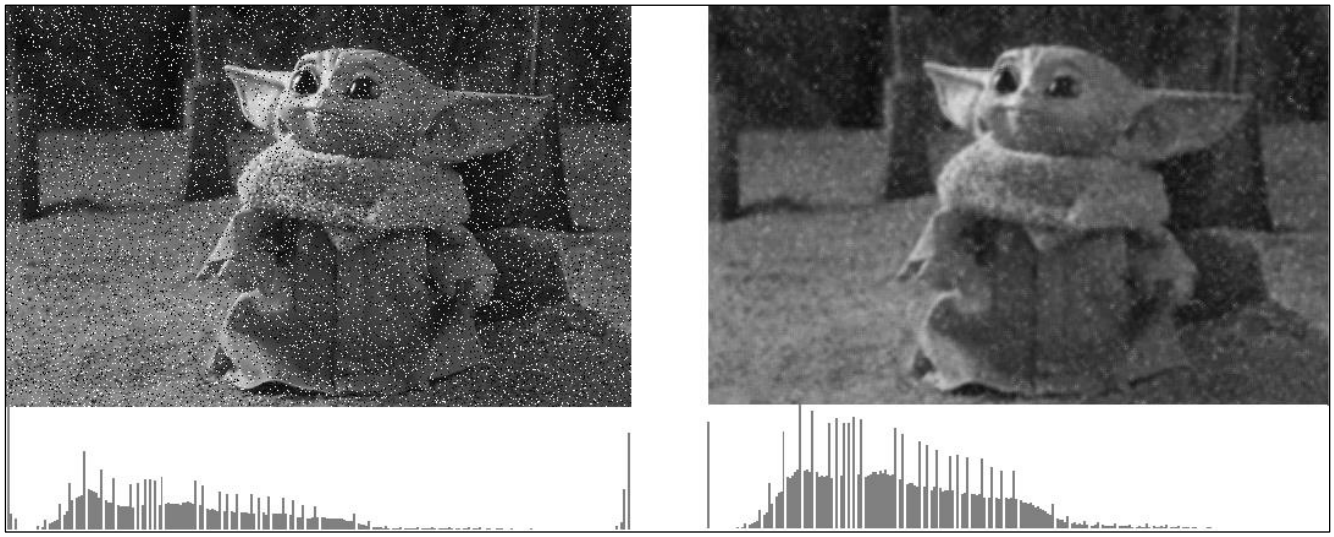

Рисунок 2.10 – Зображення з імпульсним шумом до фільтрації, білатеральним фільтром та після.

На рисунку 2.11 наведено результат роботи білатерального фільтра з рівномірним шумом.

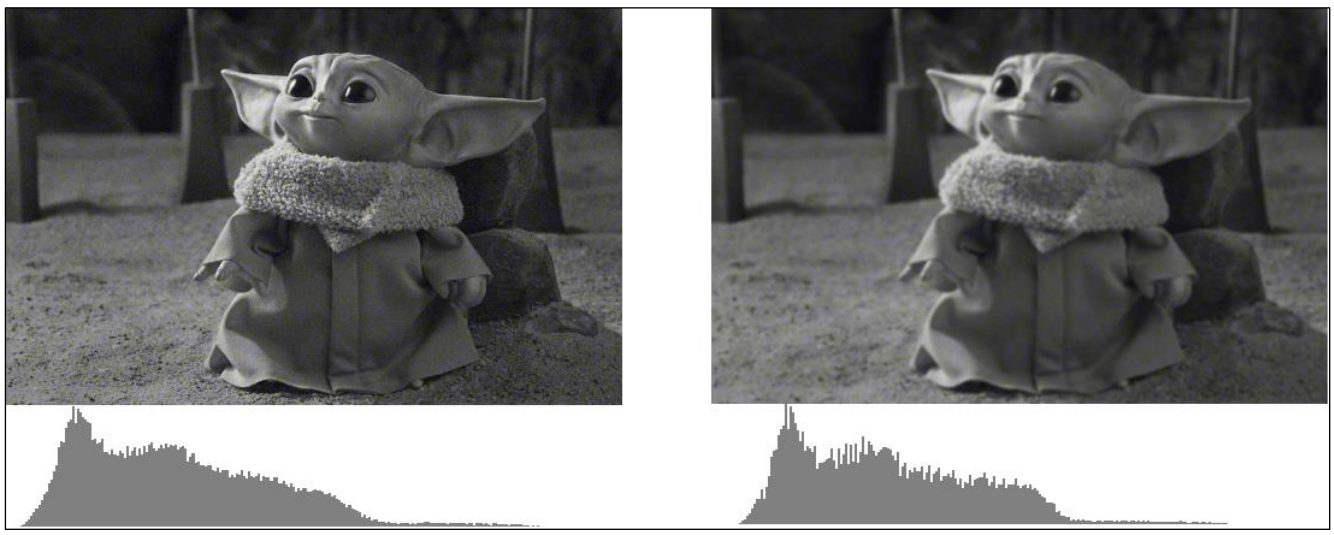

Рисунок 2.11 – Зображення з рівномірним шумом до фільтрації, білатеральним фільтром та після.

На рисунку 2.12 наведено вихідне зображення та результати застосування білатерального фільтра, до різних видів шумів (вихідне зображення, гаусівський, рівномірний, імпульсний)

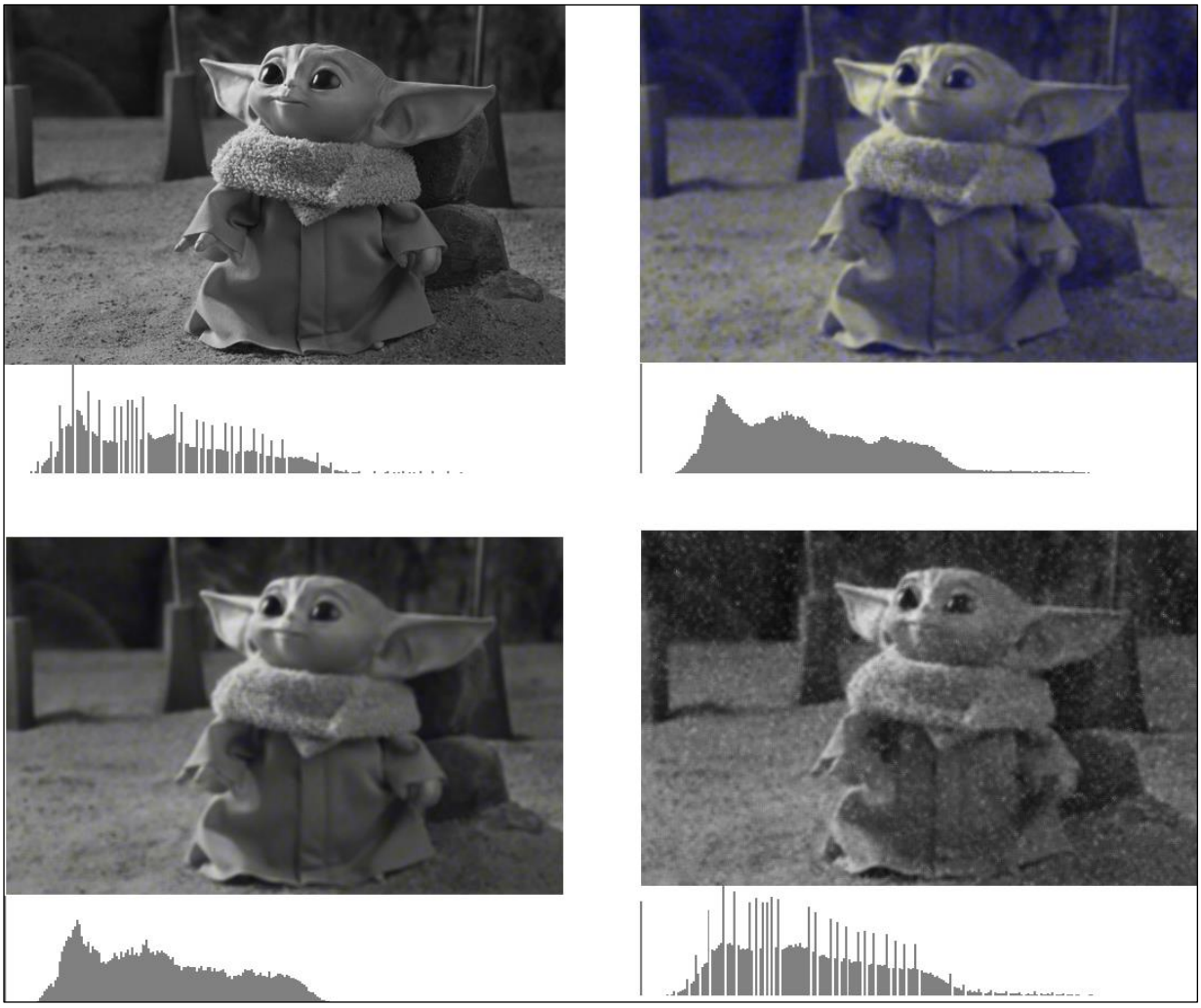

Рисунок 2.12 – Порівняння вихідного зображення та відфільтрованих зображень ,за допомогою білатерального фільтра, з гаусівським, рівномірним та імпульсним шумах відповідно.

Далі наведено приклад реалізації білатерального фільтра, на мові програмування c#, метод на вході отримує зображення, розмір вікна самого фільтра, sigmaColor – параметр, який контролює гладкість фільтрації залежно від різниці в яскравості між сусідніми пікселями. Чим більше значення sigmaColor, тим сильнішими будуть розмиті межі між об'єктами на зображенні, sigmaSpace – параметр, який контролює гладкість фільтрації залежно від відстані між пікселями у просторі зображення. Чим більше значення sigmaSpace, тим сильніше

будуть розмиті деталі зображення, що знаходяться на великій відстані один від одного. Далі алгоритм буде виглядати наступним чином:

- вибираємо розмір ядра та значення стандартного відхилення (sigma) для просторової складової та інтенсивності кольору;
- проходимо по кожному пікселю зображення та для кожного пікселя вибираємо сусідів відповідно до розміру ядра;
- обчислюємо просторову вагу для кожного пікселя;
- обчислюємо інтенсивність ваги для кожного пікселя;
- обчислюємо вагу для кожного пікселя як добуток просторової та інтенсивної ваги;
- обчислюємо нове значення кольору для кожного пікселя як середнє значення кольорів його сусідів.

```
    public static Bitmap BilateralFilter(Bitmap input, int diameter, double sigmaColor, double sigmaSpace)
            {
                Bitmap output = new Bitmap(input.Width, input.Height);
                int radius = diameter / 2;

                double[,] spatialWeights = new double[diameter, diameter];
                double weightSum = 0;
                for (int y = -radius; y <= radius; y++)
                {
                    for (int x = -radius; x <= radius; x++)
                    {
                        double weight = Math.Exp(-(x * x + y * y) / (2 * sigmaSpace * sigmaSpace));
                        spatialWeights[y + radius, x + radius] = weight;
                        weightSum += weight;
                    }
                }

                for (int y = radius; y < input.Height - radius; y++)
                {
                    for (int x = radius; x < input.Width - radius; x++)
                    {
                        Color centerColor = input.GetPixel(x, y);
                        double sumR = 0;
                        double sumG = 0;
                        double sumB = 0;
                        double sumWeights = 0;
                        for (int i = -radius; i <= radius; i++)
                        {
                            for (int j = -radius; j <= radius; j++)
```

```
                            {
                                Color neighborColor = input.GetPixel(x +
j, y + i);
                                double colorDistance = Math.Sqrt(
                                    Math.Pow(centerColor.R               –
neighborColor.R, 2) +
                                    Math.Pow(centerColor.G               –
neighborColor.G, 2) +
                                    Math.Pow(centerColor.B               –
neighborColor.B, 2)
                                );
                                double    colorWeight    =    Math.Exp(–
colorDistance / (2 * sigmaColor * sigmaColor));
                                double spatialWeight = spatialWeights[i +
radius, j + radius];
                                double    weight    =    colorWeight    *
spatialWeight;
                                sumR += neighborColor.R * weight;
                                sumG += neighborColor.G * weight;
                                sumB += neighborColor.B * weight;
                                sumWeights += weight;
                            }
                        }
                        int red = (int)Math.Round(sumR / sumWeights);
                        int green = (int)Math.Round(sumG / sumWeights);
                        int blue = (int)Math.Round(sumB / sumWeights);
                        red = Math.Max(Math.Min(red, 255), 0);
                        green = Math.Max(Math.Min(green, 255), 0);
                        blue = Math.Max(Math.Min(blue, 255), 0);
                        output.SetPixel(x, y, Color.FromArgb(red, green,
blue));
                    }
                }
                return output;
            }
```

# 3 СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ПІДГОТОВКИ ДО ЕКСПЕРИМЕНТУ

Для проведення практичної частини дослідження потрібно створити комп'ютерну програму, яка буде реалізовувати наступні функції:

– додавання гаусівського шуму до фото;
– додавання рівномірного шуму до фото;
– додавання імпульсного шуму до фото;
– побудова гістограми розподілу яскравості пікселів;
– фільтрація за допомогою фільтру Гауса;
– фільтрація за допомогою фільтру середнього значення;
– фільтрація за допомогою білатерального фільтру;
– виділення контурів за допомогою фільтру Собеля;
– обчислення середньоквадратичної помилки для кількісної оцінки шуму.

Для реалізації вище перерахованих функцій було обрано платформу .Net і мову програмування c#. Частина функцій була реалізована за допомогою бібліотеки Emgu.CV [7]. Emgu.CV є .NET обгорткою навколо OpenCV – це бібліотека програмного забезпечення, що використовується для обробки зображень і комп'ютерного зору. Emgu.CV була розроблена, щоб дозволити програмістам на .NET (включаючи C#) використовувати функціональність OpenCV.

Програму було реалізовано як консольний додаток. Приклад інтерфейсу наведено на рисунку 3.1.


Рисунок 3.1 – Приклад консольного інтерфейсу розробленої програми.

Приклад коду додавання рівномірного шуму до фото за допомогою Emgu.CV бібліотеки:

```
public static Mat AddUniformNoise(Mat image, int minValue, int maxValue)
        {
            Mat noise = new Mat(image.Size, DepthType.Cv32S,
image.NumberOfChannels);
            CvInvoke.Randu(noise, new MCvScalar(minValue), new
MCvScalar(maxValue));

            Mat result = new Mat();
            image.ConvertTo(result, DepthType.Cv32S);
            CvInvoke.Add(result, noise, result);

            result.ConvertTo(result, DepthType.Cv8U);
            return result;
        }
```

# 4 ДОСЛІДЖЕННЯ

## 4.1 Виділення контурів на зображенні

Виділення контурів на зображенні є важливим етапом в багатьох завданнях розпізнавання образів. Контури можуть надати важливу інформацію про форму та структуру об'єктів на зображенні, які можуть бути важкими для виявлення при прямому аналізі самих пікселів.

Контурні фільтри, такі як фільтр Собеля, використовуються для виділення границь об'єктів на зображеннях. Ці границі можуть бути використані для сегментації зображень, виявлення особливостей, визначення форми об'єктів та багатьох інших завдань [7].

На рисунку 4.1 зображено приклад роботи фільтра Собеля для вихідного зображення з розміром ядра 5x5.

Рисунок 4.1 – Виділення контурів за допомогою фільтра Собеля з ядром 5x5

Також важливо враховувати, що шум на зображенні може вплинути на якість виділення контурів, тому може бути необхідно застосувати фільтрацію шуму перед обробкою контурів.

Хоча вихідні зображення з шумом можуть виглядати візуально прийнятно, шум може суттєво вплинути на якість виявлення контурів. Специфічно, шум може призвести до виникнення небажаних деталей та розмиття або

невизначеності контурів. Це підкреслює важливість застосування адекватних методів фільтрації шуму перед обробкою зображень для задач розпізнавання образів [8].

На рисунку 4.2 зображено результат застосування фільтра Собеля до зображень зашумлених різними видами шумів – гаусівським, рівномірним та імпульсним, відповідно.

Рисунок 4.2 – Результат виділення контурів на зашумлених зображеннях.

Далі розглянемо вплив деяких фільтрів, призначених для зниження рівня шуму, на процес виділення контурів на зображенні.

На рисунку 4.3 зображено результат застосування фільтра Собеля до зображень зашумлених різними видами шумів – гаусівським, рівномірним та імпульсним, відповідно, в першому рядку та відфільтрованих за допомогою фільтра Гауса у другому рядку.

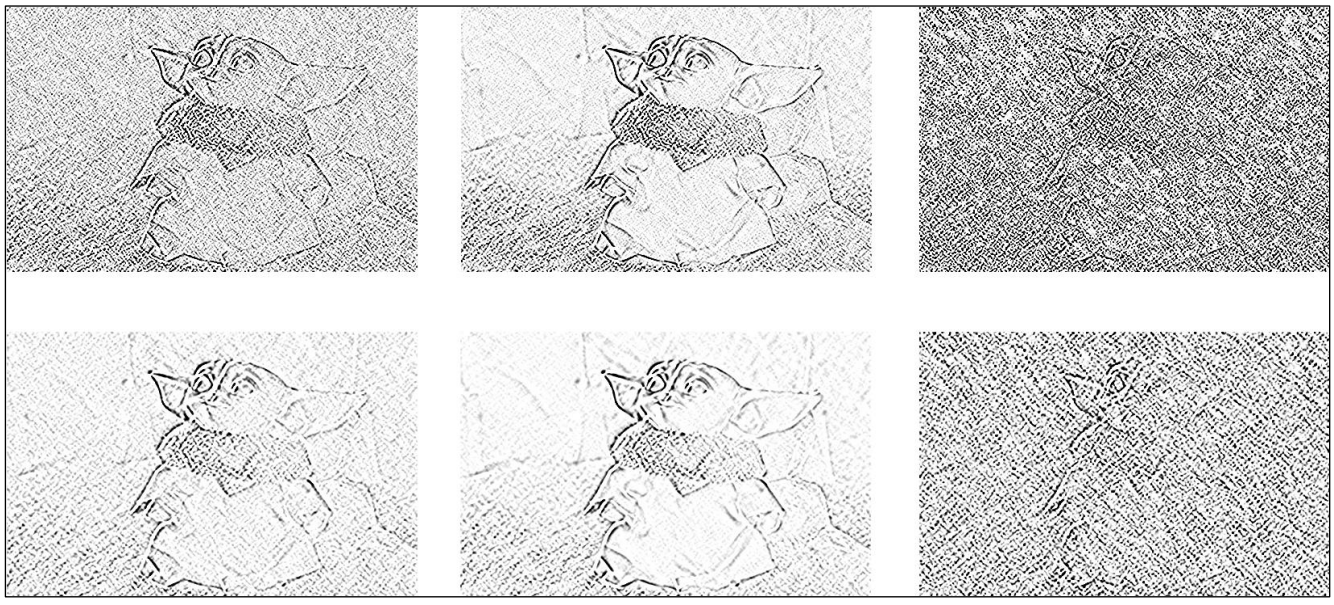

Рисунок 4.3 – Порівняння роботи фільтра Собеля на зашумлених зображення, та тих же зображеннях після фільтрації фільтром Гауса.

Ретельно аналізуючи представлені зображення, ми можемо спостерігати, як фільтр Гауса впливає на процес виділення контурів при наявності різних типів шуму. Для зображень, що мають гаусівський та рівномірний шуми, застосування фільтру Гауса призвело до виразного покращення якості виділення контурів.

Специфічно, небажані деталі, що були внаслідок шуму, ефективно усунені, а зернистість зображення зменшилася. Це призвело до збільшення чіткості границь, що зробило контури об'єктів більш виразними та відчутними.

Однак, для імпульсного шуму вплив фільтру Гауса менш помітний. Зокрема, не спостерігається зазначеного вище значного покращення якості виділення контурів. Це підкреслює важливість вибору відповідного фільтра для подавлення шуму, який відповідає специфіці шуму, присутнього на зображенні [9].

На рисунку 4.4 зображено результат застосування фільтра Собеля до зображень зашумлених різними видами шумів – гаусівським, рівномірним та імпульсним, відповідно, в першому рядку та відфільтрованих за допомогою фільтра середнього значення у другому рядку.

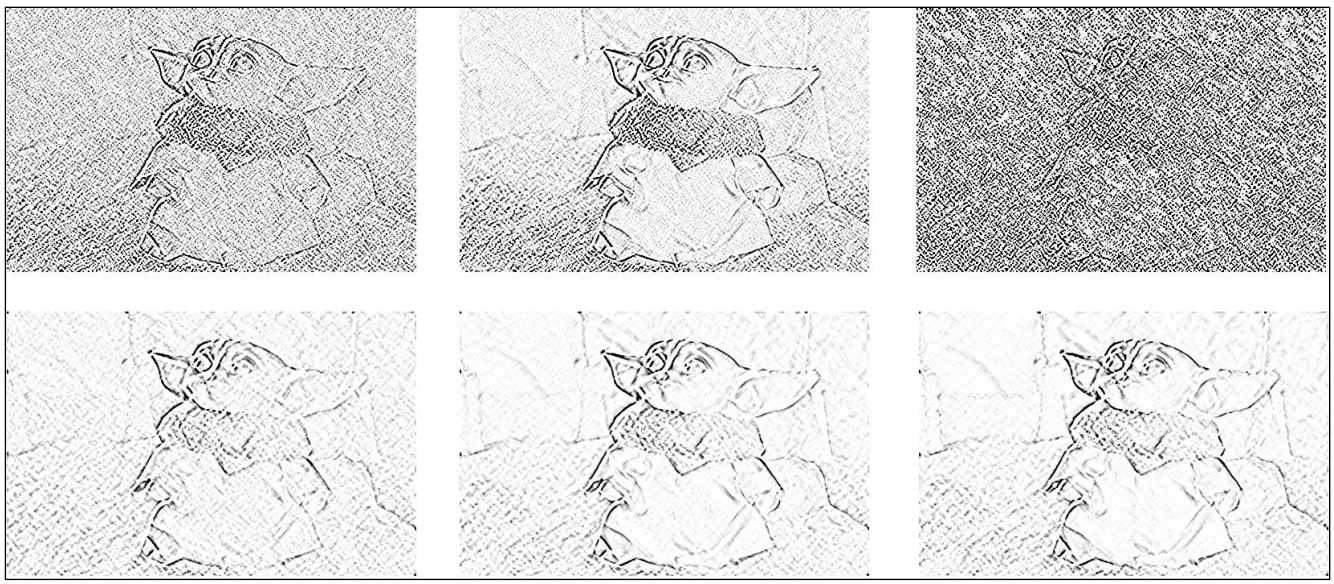

Рисунок 4.4 – Порівняння роботи фільтра Собеля на зашумлених зображення, та тих же зображеннях після фільтрації фільтром середнього значення.

При ретельному аналізі результатів застосування фільтру середнього значення, ми можемо спостерігати, що цей фільтр показав себе ефективно при

боротьбі з імпульсним та рівномірним шумами. Він успішно зменшив зернистість на зображенні, при цьому зберігаючи контури досить чіткими.

Однак, при обробці зображень з гаусівським шумом, фільтр середнього значення виявився менш ефективним. Він призвів до деякого розмиття контурів, що знизило якість виділення контурів.

На рисунку 4.5 зображено результат застосування фільтра Собеля до зображень зашумлених різними видами шумів – гаусівським, рівномірним та імпульсним, відповідно, в першому рядку та відфільтрованих за допомогою білатерального фільтра у другому рядку.

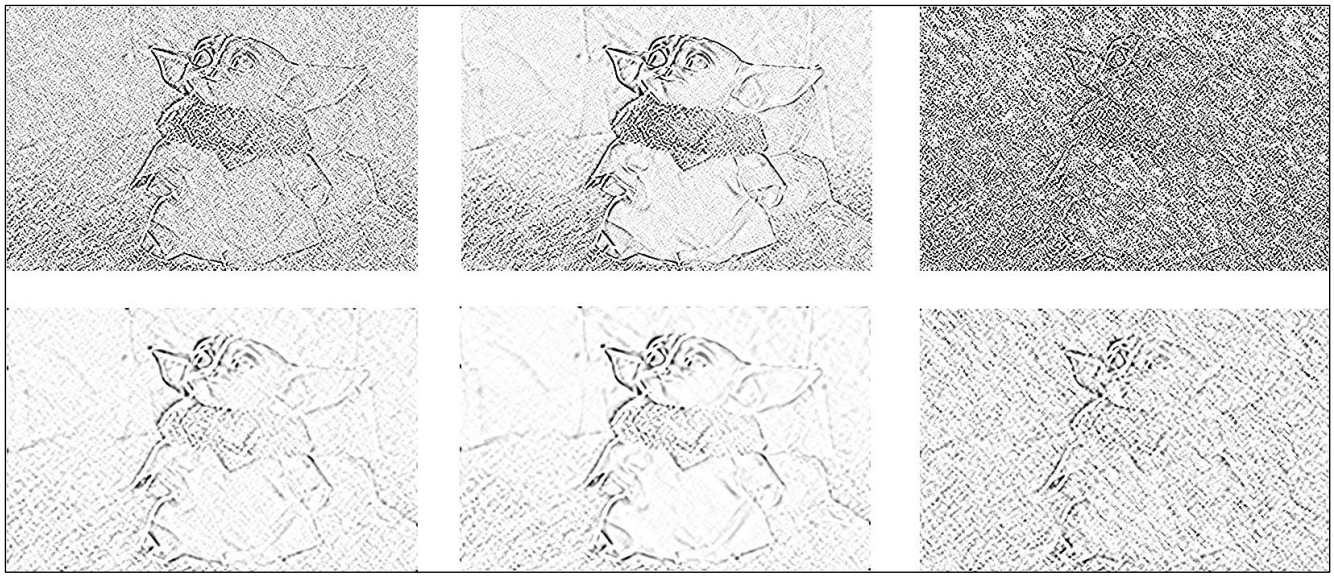

Рисунок 4.5 – Порівняння роботи фільтра Собеля на зашумлених зображення, та тих же зображеннях після фільтрації білатеральним фільтром значення.

Аналізуючи результати використання білатерального фільтру, можна зробити висновок, що він приносить лише незначне покращення при обробці імпульсного шуму. Контурні лінії залишаються досить невизначеними, а кількість небажаних деталей, викликаних шумом, залишається високою.

Однак, при обробці зображень з рівномірним та гаусівським шумами, білатеральний фільтр демонструє виразне покращення. Незважаючи на те, що виникає деяка розмитість, загалом якість зображення покращується.

На основі проведених порівнянь між впливом трьох різних фільтрів (Гауса, середнього значення та білатерального) на зображення з різними типами шуму, ми можемо зробити декілька ключових висновків.

Фільтр Гауса виявився особливо ефективним у видаленні гаусівського та рівномірного шумів, зберігаючи при цьому чіткість контурів. Однак, він показав меншу ефективність у боротьбі з імпульсним шумом.

Фільтр середнього значення добре справився з імпульсним та рівномірним шумами, але був менш ефективним для гаусівського шуму, призводячи до розмиття контурів.

Білатеральний фільтр показав покращення для рівномірного та гаусівського шумів, але при цьому не справився з імпульсним шумом, залишаючи контури невизначеними.

Ці висновки можуть бути обумовлені характеристиками кожного з фільтрів. Наприклад, фільтр Гауса відомий своєю здатністю зменшувати високочастотний шум, що пояснює його ефективність при видаленні гаусівського та рівномірного шумів. З іншого боку, білатеральний фільтр зберігає краї, що може бути корисним для боротьби з рівномірним та гаусівським шумами, але менш ефективним для імпульсного шуму.

Ці результати підкреслюють важливість вибору відповідного методу фільтрації, заснованого на характеристиках конкретного типу шуму, що присутній на зображенні. Кожен тип шуму має свої особливості та впливає на зображення по-різному, тому важливо враховувати особливості кожного з них і використовувати різні методи фільтрації.

## 4.2 Кількісна оцінка шуму

Кількісна оцінка шуму – це процес визначення ступеня шуму на зображенні, який вимірюється за допомогою числових параметрів. Цей показник дозволяє об'єктивно порівнювати рівень шуму на різних зображеннях та використовувати його для підтримки рішень при обробці зображень. Існують різні методи для кількісної оцінки шуму одна з них наведені нижче.

RMSE, або Root Mean Square Error (середньоквадратична помилка), це статистична міра, що вказує на різницю між значеннями, спостережуваними та прогнозованими, або між оригінальним зображенням і зашумленим зображенням у контексті обробки зображень.

RMSE використовується для квантової оцінки якості зображення після обробки і вимірюється в одиницях, що використовуються для вимірювання даних (наприклад, значень пікселів).

Математична формула для обчислення RMSE виглядає так (формула 4.1):

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \quad (4.1)$$

де n – кількість точок даних (або пікселів у зображенні);

$y_i$ – дійсне значення в точці і (значення пікселя в оригінальному зображенні);

$\hat{y}_i$– прогнозоване або спостережуване значення в точці і (значення пікселя в зашумленому зображенні).

RMSE враховує квадратичну різницю між оригінальними та спостережуваними значеннями, що означає, що більші помилки матимуть більшу вагу, ніж менші помилки. Результат потім береться корінь, щоб повернути масштаб до початкових одиниць вимірювання [8].

Таблиця 1 – Квадратичну різницю між оригінальними та спостережуваними значеннями

| | Зображення з шумом | Фільтр середнього значення | Фільтр Гауса | Білатеральний фільтр |
|---|---|---|---|---|
| Гаусівський | 9.3149 | 14.137 | 6.5125 | 14.155 |
| Рівномірний | 3.5067 | 13.8835 | 6.1080 | 14.0889 |
| Імпульсний | 45.1244 | 13.9617 | 15.2329 | 16.7102 |

Оцінюючи середньоквадратичну помилку при фільтрації шумів на зображеннях, можна зробити такі висновки: фільтр середнього значення ефективно працює з імпульсним шумом, але погіршує якість зображення при використанні гаусівського або рівномірного шуму. Гаусівський фільтр добре працює з гаусівським та рівномірним шумом, але не дуже ефективний при роботі з імпульсним шумом. Аналіз результатів показує, що білатеральний фільтр має найбільшу середньоквадратичну помилку серед усіх розглянутих фільтрів. Однак, ці дані не повністю відображають фізичні спостереження результатів роботи фільтрів, іноді фільтри можуть виявляти більшу ефективність в реальних умовах, ніж їхня середньоквадратична помилка.

Нижче наведено код на мові програмування c# для розрахунку RMSE

```
        public static double CalculateRMSE(Bitmap original, Bitmap noisy)
        {
            if (original.Size != noisy.Size)
                throw new ArgumentException("Images must have the same
dimensions.");

            double errorSum = 0;
            for (int i = 0; i < original.Width; i++)
            {
                for (int j = 0; j < original.Height; j++)
                {
                    Color originalPixel = original.GetPixel(i, j);
                    Color noisyPixel = noisy.GetPixel(i, j);

                    int diff = originalPixel.R – noisyPixel.R;
                    errorSum += diff * diff;
                }
            }

            double meanError = errorSum / (original.Width *
original.Height);
            return Math.Sqrt(meanError);
        }
```

# ВИСНОВКИ

В ході виконання дослідження проведено аналіз предметної галузі, систематичний огляд методів фільтрації шумів, експерименти щодо визначення оптимальних параметрів фільтрації шумів, порівняння ефективності застосування методів для вирішення практичних задач.

Мета дослідження – аналіз існуючих методів оптимальної фільтрації шумів для підвищення якості розпізнавання, оцінка їх ефективності, переваг та недоліків в процесі фільтрації шумів на зображенні, зменшення впливу шумів в процесі розпізнавання, визначення оптимальних параметрів фільтрації шумів – виконана у повному обсязі.

Процес дослідження включав детальний аналіз декількох найбільш поширених фільтрів, які використовуються в обробці зображень:

- фільтр Гауса, який використовується для розмиття зображення і зменшення деталей;
- фільтр Собеля, який використовується для підсилення країв на зображенні;
- білатеральний фільтр, який дозволяє зменшити шум, зберігаючи при цьому краї;
- фільтр середнього значення, який використовується для розмиття зображення.

Кожен з цих фільтрів було реалізовано з використанням мови програмування C#. Це включало написання коду для кожного з фільтрів, а також створення тестових сценаріїв для перевірки їх ефективності.

Результати виконаної роботи показали, що використання фільтрів для зменшення шуму та покращення якості зображення – це ефективна стратегія в обробці цифрових зображень. Водночас було виявлено, що вибір певного фільтру залежить від характеру шуму та специфічної мети розпізнавання. Розуміння принципів роботи різних фільтрів та їх оптимізація значно покращує ефективність та якість обробки зображень. Цей процес включає в себе вивчення специфіки

шумів, їх впливу на зображення, параметрів фільтрів. Це дає можливість для гнучкого використання методів та пошуку оптимальних параметрів фільтрації.

В цілому кваліфікаційна робота підтверджує значення та ефективність застосування методів фільтрації шумів для вирішення практичних задач, наприклад, в таких галузях, як комп'ютерний зір, медична діагностика, аерофотозйомка та інші.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки “Основи розпізнавання образів” / Упорядн.: **А.В. Работягов.** – Saarbrücken: LAP Lambert Academic Publishing (Deutschland), 2014. – 64 с. ISBN: 978-3-659-61240-4.

2. Методичні вказівки до практичних занять з дисципліни “Основи штучного інтелекту”; для студентів усіх форм навчання напряму підготовки 6.050103 – Програмна інженерія / Упоряд.: **А.В. Работягов.** – Харків: ХНУРЕ, 2013. – 62 с.

3. **Работягов А.В.**, Ляшенко В.В., Кобылин О.А. Сегментация сложных изображений цитологических препаратов // Радиотехника. – Харьков: ХНУРЭ, 2016. – № 185. – С. 87-95.

4. What Is Image Recognition? 3 things you need to know [Електронний ресурс] - Режим доступу до ресурсу: https://www.mathworks.com/discovery/image-recognition-matlab.html (дата звернення 20.04.2023)

5. Image recognition [Електронний ресурс] - Режим доступу до ресурсу: https://www.techtarget.com/searchenterpriseai/definition/image-recognition (дата звернення 03.05.2023)

6. Image Recognition it nutshell [Електронний ресурс] - Режим доступу до ресурсу: https://www.sciencedirect.com/topics/engineering/image-recognition (дата звернення 23.04.2023)

7. EMGU Wiki Main page. [Електронний ресурс] - Режим доступу до ресурсу: https://emgu.com/wiki/index.php/Main_Page (дата звернення 01.05.2023)

8. Root Mean Square Error (RMSE). [Електронний ресурс] - Режим доступу до ресурсу: https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error (дата звернення 11.04.2023)

9. Обробка зображень, цифрове оброблення сигналів, розпізнавання образів [Електронний ресурс]. – Режим доступу: http://www.sati.archaeology.nsc.com/gr/texts/imageprocess/index.html. (дата звернення 26.04.2023)

10. Грузман, І.С. Цифрова обробка зображень в інформаційних системах/І.С. Грузман, В.С. Киричук, В.П. Косих, Г.І. Перетягін, А.А. Спектр // Науковий посібник. - Київ: НДТУ, 2002. - С. 125-139.

11. Гонсалес, Р. Цифрова обробка зображень/Р. Гонсалес, Р. Вудс. - М.: Техносфера, 2005. - С. 148-414.