Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТ
Пояснювальна записка

Рівень вищої освіти – другий (магістерський)

Дослідження моделей й алгоритмів фільтрації шуму на цифровому зображенні
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-1

Нірі М.Ю
(прізвище, ініціали)

Спеціальності 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-наукова програми
(тип програми)

Інженерія програмного забезпечення
(повна назва освітньої програми

Керівник: проф. Смеляков К.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ Дудар З.В.

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. Кафедри_____
(підпис)
« » 2020 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Нірі Михайлу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей й алгоритмів фільтрації шуму на цифровому зображенні.

Затверджена наказом університету від «___»_____20___р. №_____

2. Термін подання студентом роботи до екзаменаційної комісії

«___»_____20 р

3. Вихідні дані до роботи *алгоритми й моделі фільтрації шуму на цифровому зображенні пояснювальна записка. Використовувати ОС Windows, середовище об'єктно-орієнтованого проектування.*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи аналіз проблемної галузі і постановка задачі, огляд методів й алгоритмів фільтрації шуму на цифровому зображенні, порівняння різних алгоритмів й моделей й надання рекомендацій до використання цих фільтрів*.

5. Консультанти розділів роботи

Найменування розділу	Консультант	Позначка консультанта про виконання розділу

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	30.03.2020 – 05.04.2020	виконано
2	Огляд існуючих методів	06.04.2020- 10.04.2020	виконано
3	Алгоритми фільтрування шуму на цифровому зображенні	11.04.2020- 15.04.2020	виконано
4	Підготовка пояснювальної записки	16.04.2020- 30.04.2020	виконано
5	Підготовка презентації та доповіді	1.05.2020- 4.05.2020	виконано
6	Попередній захист	8.05.2020	виконано
7	Нормконтроль, рецензування	10.05.2020- 12.05.2020	виконано
8	Занесення диплома в електронний архів		
9	Допуск до захисту у зав. кафедри		

Дата видачі завдання «_____»_____20___р.

Студент _____ Нірі М.Ю.
(підпис)

Керівник роботи _____ проф. Смеляков К.С.
(підпис) (посада, прізвище, ініціали)

Реферат / Abstract

Атестаційна робота магістра містить: 66 сторінок, 32 рисунків, 14 формул, 13 джерел.

.NET, АЛГОРИТМ, ФІЛЬТР ШУМУ, ОБРОБКА ЗОБРАЖЕННЯ, ЯКІСТЬ ФІЛЬТРАЦІЇ, ПІКСЕЛЬ, VISUAL STUDIO, SOLID, KISS, DRY, RGB.

Метою роботи є моделі й алгоритми фільтрації шуму на цифровому зображенні, порівняння різних алгоритмів й пошук оптимальних для кожної специфічної задачі.

Методи розробки базуються на інструментах розробки десктопних застосувань на мові програмування C# та використовуючи головні принципи проектування: SOLID, DRY та KISS.

У результаті роботи було розглянуто й порівняно моделі й алгоритми фільтрації шуму на цифровому зображенні для різних специфічних задач. Знайдено негативні й позитивні сторони кожного з методів або алгоритмів. Зроблено висновки с рекомендаціями стосовно використання того чи іншого алгоритму та моделі фільтрації цифрового зображення від шуму.

.NET, ALGORITHM, NOISE FILTER, IMAGE PROCESSING, FILTER QUALITY, PIXEL, VISUAL STUDIO, SOLID, KISS, DRY, RGB.

The object of the study is the models and algorithms of noise filtering on a digital image, comparison of different algorithms and search of the optimal for each specific task.

The development methods are based on the C# and desktop application development tools using the basic design principles: SOLID, DRY and KISS.

As a result, models and algorithms for noise filtering on digital images were considered and compared for various specific tasks. The negative and positive sides of each of the methods or algorithms were found. The conclusions are made with recommendations in which situation to use a particular algorithm and model of filtering noise of digital image.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

RGB - red, green and blue colors. Відноситься до трьох відтінків світла, котрі можна змішати одне с одним, щоб отримати інші відтінки. Поєднання цих кольорів використовується для отримання кольорових зображень на фотоапаратах, екранах, проекторах та інше [1].

Фільтр зображення - це програмне забезпечення, котре маніпулює з зображенням, для покращення чи отримання якогось ефекту або для подальшого аналізу зображення.

RAW - це формат даних, котрий містить у собі необроблені дані, що дозволяє уникнути втрату інформації. У такому форматі файлу міститься повна інформація о сигналі. Дані у цьому файлі не є стиснутими, стиснутими без втрати або стиснутими з втратами.

Піксель - це елемент зображення чи найменший логічний двовимірний елемент цифрового зображення. Піксель це неділимий об’єкт прямокутної форми, характеризується певним кольором.

Шум зображення - це дефект зображення, внесений матрицею, тобто фото сенсором, внаслідок недосконалої технології та фотонної природи світла.

Датасет - це набір даних, файлів, зображень чи просто однотипних даних. Використовується для тестування, навчання нейронних сітей та інше.

Photoshop - графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в галузі комерційних засобів редагування растрових зображень і найвідомішим продуктом фірми Adobe. Часто цю програму називають просто Photoshop.

ISO – це один з трьох головних стовпів фотографії (інші два – витримка та діафрагма), і це має великий вплив на зображення. Це налаштування камери, котра дозволяє освітлити або затемнити фотографію. Це рівень сприйняття світла матрицею.

ВСТУП

Понад 90% інформації людина отримує за допомогою зору. Система зору люди обробляє приблизно зі швидкістю десять мільйонів біт у секунду, в той час як загальна швидкість обробки сенсорної інформації для людини становить близько одинадцяти мільйонів біт у секунду. Око людини має два типи світлочутливих клітин – фоторецепторів: високочутливі палички та менш чутливі колбочки. Палички використовуються в умовах недостатньої кількості світла, тобто це як механізм нічного зору, однак вони забезпечують лише нейтральне в колірному відношенні сприйняття дійсності. Колбочки працюють при більш високій кількості світла, ніж палички. Колбочки відповідають за механізм денного зору, тобто вони забезпечують колірний зір. Колбочки розділяються на три види які припадають на червоний, зелений і синій. Тобто модель RGB (red, green та blue) – це адитивна колірна модель, котра описує не тільки спосіб синтезу кольору люди, а й застосовується у техніці для отримання та відображення зображень різного характеру. У RGB моделі кількість градацій дорівнює 256, це дорівнює 256^3 = 16 777 216 відтінків кольору. Переваги RGB моделі:

– аналогічність з сканерами, проекторами, моніторами та другими пристроями відтворення чи отримання зображення;

– величезна колірна гама, схожа с зором людини;

– велика кількість інструментів для обробки зображення, тобто фільтрів;

– широкий спектр кольорів.

Більша частина процесів у галузі діяльності обробки зображень були присвячені обробці, аналізу та інтерпретації чорно-білих зображень. Застосунки комп’ютерного зору вимагають аналіз, обробку та інтерпретацію кольорового зображення. Кольорові зображення виражають як просторово змінюється спектральні розподіли випромінювання або відображення. Щоб ми могли

аналізувати данні кольорового зображення нам потрібно зіставити кольори на зображенні на кольоровий простір, тобто зробити систему представлення кольорів.

Обробка зображення – це метод виконання деяких операцій над зображенням, щоб отримати покращене зображення або отримати з нього необхідну інформацію. Наприклад на вході ми маємо зображення, а на виході ми можем отримати оброблене-покращене зображення чи властивості пов’язані з зображенням, тобто на виході ми отримуємо корисну інформацію. Зараз ця сфера обробки зображень стала дуже швидко зростати, тому що з’явилось багато сфер використання и багато потреб до обробки зображення, наприклад: комп’ютерний зір, цифрове зображення для професійних фотохудожників, аналіз зображень з телескопів, пошук мікро чи макро об’єктів [3].

Можна виділити три етапи обробки зображення:

– імпорт зображення;

– маніпулювання та аналіз;

– отримання результатів у виді обробленого зображення або звіту, заснованого на аналізі зображення.

За наявністю великою кількості сфер використання, великої кількості потреб обробки зображення та різноманітністю фільтрів котрі потрібно використовувати було вирішено дослідити цю тему більш детально.

Насамперед потрібно відзначити, що у ХНУРЕ на кафедрі Програмної Інженерії є досить багато наукових досліджень спрямованих на розробку математичних моделей механізмів людського зору, слуху, сприйняття та пізнавання. Не менш цікавий напрям, це ідентифікація об’єктів для обробки цифрової інформації не тільки статичного зображення, а й у реальному часі. Ці всі напрями тісно пов’язані з темою дипломної роботи – Дослідження моделей й алгоритмів фільтрації шуму на цифровому зображенні.

Метою дослідження є виявлення найкращих комбінацій потреб та алгоритму видалення шуму на цифровому зображенні, наприклад чорно-біле зображення та

алгоритм медіанного фільтру. Задача дати рекомендацію що до використання різних алгоритмів фільтрів при різних умов та надати порівняння їх між собою.

Об’єктом дослідження є огляд існуючих фільтрів для зменшення шуму на цифрових зображеннях, котрі необхідно покращити шляхом використання моделей й алгоритмів фільтрації шуму. Під «покращити» розуміється отримати якомога більше корисної інформації з зображення та отримати зображення якомога приємним для людського ока. На існуючих зображеннях з мого раннього архіву буде опрацьовано й налагоджено такі фільтри, як Median, Kuwahara, двосторонній та фільтр Вінера. Аналіз роботи цих фільтрів буде проходити на двох типах шуму: імпульсному та білому, й на двох різних за насиченістю й деталізацією зображеннях. У результаті дослідження буде надані висновки та рекомендації щодо використання окремого алгоритму на окремому виді цифрового зображення, позитивні й негативні сторони кожного з них, такі як втрата деталізації при видаленні шуму, велике розмиття зображення після роботи алгоритму та інша втрата корисної інформації, що може потягнути за собою погане сприйняття зображення чи некоректну роботу інших фільтрів основної обробки зображення.

1 СУЧАСНИЙ СТАН ПИТАННЯ ФІЛЬТРАЦІЇ ШУМУ НА ЗОБРАЖЕННІ

В комп’ютерних науках обробка зображень - це використання цифрового комп'ютера для обробки цифрових зображень за допомогою алгоритму. Будучи під категорією чи полем цифрової обробки сигналів, обробка цифрових зображень має багато переваг перед аналоговою обробкою зображення. Це дозволяє застосовувати набагато ширший спектр алгоритмів до вхідних даних і дозволяє уникнути таких проблем, як накопичення шуму та спотворення під час обробки. Оскільки зображення визначені у двох вимірах, обробка цифрових зображень може моделюватися у вигляді багатовимірних систем. На генерацію та розвиток цифрової обробки зображень в основному впливають три фактори: по-перше, розвиток комп'ютерів; по-друге, розвиток математики, особливо створення та вдосконалення дискретної теорії математики, по-третє, збільшився попит на широкий спектр застосувань у галузі навколишнього середовища, сільського господарства, військових, промисловості та медичної науки.

Багато методів цифрової обробки зображень були розроблені в 1960-х роках у Bell Laboratories, Лабораторії реактивного руху, Массачусетському технологічному інституті, Мерілендському університеті та кількох інших науково-дослідних установ, із застосуванням до супутникових знімків, перетворення стандартів фотографій, медичних зображень, розпізнавання символів та покращення фотографії. Метою ранньої обробки зображень було поліпшення якості зображення. Вона була спрямована на те, щоб зображення виглядало гарніше для людського ока. При обробці зображень вхід - це зображення низької якості, а вихід - зображення з покращеною якістю. Загальна обробка зображень включає в себе поліпшення, відновлення, кодування та стиснення зображення. Першою вдалою заявкою стала американська лабораторія реактивного руху. Вони використовували методи обробки зображень, такі як геометрична корекція, перетворення градацій, видалення шуму на тисячах фотографій, що були надіслані

назад космічним детектором Ranger 7 у 1964 році. Вплив успішного картографування поверхні Місяця за допомогою комп'ютера досяг величезного успіху. Пізніше була здійснена більш складна обробка зображень на майже 100 000 фотографій, відправлених назад космічним кораблем, завдяки чому були отримані топографічна карта, кольорова карта та панорамна мозаїка Місяця, які досягли надзвичайних результатів і заклали міцну основу для посадки людини на місяць.

Розуміння зображень, аналіз зображень та комп’ютерний зір мають на меті дублювати ефект зору людини шляхом цифрового сприйняття та розуміння зображення [4]. Інтерес до методів цифрової обробки зображень випливає з двох основних напрямків застосування: вдосконалення зображувальної інформації для інтерпретації людини та обробка даних фотографії для автономного машинного сприйняття. Наприклад, коли ми скануємо зображення за допомогою сканера, він не є таким, як насправді ми хочемо, і яку б задачу ми не збиралися виконати, використовуючи це зображення, ми не отримаємо його правильним, і тому обробка зображення входить у зображення.

Оскільки нам відомо, що супутник є більш корисним інструментом для отримання інформації про всесвіт, а також про землю, багато рішень приймаються нами за допомогою зображень, які надаються супутником, але зображення, зроблені супутником складаються у вигляді комбінації RGB, а значить, ми повинні перетворити її у відповідну колірну комбінацію. Супутники надсилають ці зображення або будь-які дані у вигляді цифрових сигналів, які обробляються комп'ютерами.

Однак обробка була досить важкою для обчислювальної техніки тієї епохи. Це змінилося в 1970-х, коли цифрова обробка зображень поширилася, коли стали доступні більш дешеві комп'ютери та спеціальне обладнання. Це спричинило обробку зображень у режимі реального часу для деяких завдань, таких як перетворення телевізійних стандартів. Оскільки комп'ютери загального призначення стали швидшими, вони почали брати на себе роль виділеного обладнання для всіх, крім найбільш спеціалізованих та комп'ютерно-інтенсивних

операцій. За допомогою швидких комп’ютерів та процесорів, які були доступні у 2000-х роках, цифрова обробка зображення стала найпоширенішою формою обробки зображень, оскільки це не тільки найбільш універсальний метод, але і найдешевший.

Використовується багато технологій для стиснення зображення. Для пояснення моделі стиснення зображень ми об'єднали всі ці методи для формування практичних систем стиснення зображень. Система стиснення складається з двох різних структурних блоків: кодера та декодера. Ми також можемо розглядати їх як компресор і декомпресор відповідно. Вхідне зображення подається в кодер або компресор, який створює набір символів із вхідних даних. Після передачі по каналу закодоване зображення подається в декодер або декомпресор, де генерується реконструйоване вихідне зображення. Якщо це так, система не містить помилок або зберігає інформацію; якщо ні, то деякий рівень спотворень присутній у відтвореному малюнку. І кодер, і декодер складаються з двох відносно незалежних функцій або субблоків. Кодер складається з кодера джерела, який видаляє надлишки вхідного сигналу і канального кодера, що збільшує імунітет від виходу кодера джерела. Як і слід було очікувати, декодер включає канальний декодер з подальшим декодером джерела. Якщо канал між кодером і декодером відсутній від шуму, канальний кодер і декодер опущені, а загальний кодер і декодер стають кодером джерела і декодером відповідно.

Методи комп'ютерного зору спричинили великий вплив на розвиток досліджень в галузі медицини. Більшість зображень піддається впливу різного роду шумів в процесі передачі їх по каналах зв'язку, а також на етапі формування. Тому, першим етапом обробки зображень є фільтрація. Наявність шумів на зображенні може причинити неточності та спотворення на етапі сегментації та розіпізнавання. Наприклад, система може сприйняти шуми за окремі об'єкти, що може негативно вплинути на подальші дослідження.

В результаті досліджень можна виділити такі основні види шумів: адитивний Гаусовий та імпульсний. Адитивний Гаусовий шум характеризується додаванням

до кожного пікселя зображення значень з відповідного нормального розподілу з нульовим середнім значенням. Такий шум з'являється в пристроях формування цифрових зображень. Імпульсний шум характеризується заміною частини пікселів значеннями фіксованої або випадкової величини. Такий шум пов'язаний з втратами при передачі зображень по каналах зв'язку. Зазвичай, в одному зображенні можна зустріти обидва види шумів, такі шуми називають комбінованими.

Всі види фільтрів можна розділити на такі класи: частотні, лінійні, нелінійні, комбіновані. У класі частотних фільтрів обробці підлягають коефіцієнти розкладання зашумленного сигналу по базису Фур'є або інших базисах перетворення Фур'є особливо важливе для лінійних фільтрів, оскільки множення в Фур'є-області для частотних методів - це операція згортки для вихідного зображення. У цифровій обробці сигналів широко використовуються методи лінійної фільтрації. Однак, лінійна фільтрація дає хороші результати лише в разі наявності гауссова адитивного шуму. У випадку імпульсного шуму ефективніше працюють моделі нелінійної фільтрації, зокрема медіанна та рангова фільтрація. У випадку комбінованого шуму можна послідовно застосовувати лінійні і нелінійні моделі або компонувати їх так, щоб посилити сильні сторони і послабити недоліки, це відбувається при використанні гібридних алгоритмів. Лінійні фільтри ще називають згладжуючими або усереднюючими, тому що відповідь лінійного фільтра усереднює значення пікселів, що містяться в апертурі, і таким чином згладжує зображення.

Фільтр Гауса усереднює по закону Гауса пікселі довкола точки. Властивість фільтру - згортка з самим собою:

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

Згортка з радіусом σ двічі надасть результат аналогічний згортці з радіусом $\sigma\sqrt{2}$. Ядро сепарабельне – розкладається в суму одномірних ядер. Приклад двовимірної згортки можна подивитися на рисунку 1, розкладання фільтру на 2 одновимірні згортки на рисунку 2, згортка по стрічках зображена на рисунку 3.

1	2	1	*	2	3	3
2	4	2		3	5	5
1	2	1		4	4	6

«Рисунок 1 – приклад 2D згортки»

1	2	1	=	1	*	1	2	1
2	4	2		2				
1	2	1		1				

«Рисунок 2 - Розкладання фільтру на 2 1D фільтри»

1	2	1	*	2	3	3	=		11	
				3	5	5			18	
				4	4	6			18	

«Рисунок 3 - Згортка по стрічках»

Найбільш відомим з фільтрів, основаних на порядкових статистиках, є медіанний фільтр. При обчисленні медіани значення в самій точці (тобто в центрі околиці) також ураховується. Широка популярність медіанних алгоритмів

обумовлена тим, що вони прекрасно пристосовані для зменшення деяких видів випадкових шумів, і при цьому приводять до меншого розмивання в порівнянні з лінійними алгоритмами, що згладжують, того ж розміру. Медіанні алгоритми особливо ефективні при наявності як біполярного, так й уніполярного імпульсного шуму. Насправді, застосування медіанних алгоритмів дає відмінні результати для зображень, які спотворені шумом цього типу. Дія цього фільтру полягає в заміні значення в точці зображення на медіану значень яскравості в околиці цієї точки:

$$\hat{f}(x,y) = \underset{(s,t) \in S_{xy}}{med}\{g(s,t)\} \tag{3}$$

Адаптивні медіанні алгоритми. Медіанні фільтри добре працюють доти, поки просторова щільність імпульсного шуму невелике. Адаптивна медіанна фільтрація допомагає впоратися з імпульсним шумом, імовірності якого перевищують вказані значення. Додаткова перевага адаптивного медіанного алгоритму полягає в тому, що такий алгоритм «намагається зберегти деталі» в областях, спотворених не імпульсним шумом. Звичайний медіанний фільтр такою властивістю не володіє. Подібно всім розглянутим дотепер алгоритмам, адаптивний медіанний алгоритм здійснює обробку в прямокутній околиці S_{xy}. Однак, на відміну від цих алгоритмів, адаптивний медіанний алгоритм змінює (збільшує) розміри околиці S_{xy}під час роботи відповідно до наведених нижче умов. Будемо пам'ятати про те, що відгук алгоритму являє собою однину, що заміщає значення елемента зображення в тій точці (x, y), що є центром околиці S_{xy}, у сучасний момент. Існують наступні позначення:

– Z_{min}- мінімальне значення яскравості в S_{xy};

– Z_{max} - максимальне значення яскравості в S_{xy};

– Z_{med} - медіана значень яскравості в S_{xy};

– Z_{xy} - значення яскравості в точці (x, y);

– S_{max} - максимальний припустимий розмір S_{xy}.

Для розуміння того, як працює цей алгоритм, необхідно пам'ятати, що його застосування переслідує три основні цілі: видалити біполярний імпульсний шум, забезпечити згладжування шумів інших типів, а також звести до мінімуму такі спотворення, як надмірне стоншення або стовщення границь об'єктів. Значення Z_{min} й Z_{max}сприймаються алгоритмом статистично як значення «імпульсних» складових шуму, навіть якщо вони не рівні найменшим і найбільшому можливим значенням яскравості на зображенні.

З урахуванням останнього зауваження ми бачимо, що область А алгоритму має на меті визначити, чи є медіана Z_{med}імпульсом («чорним» або «білим») чи ні. Якщо умова Z_{min}- <Z_{med} <Z_{max} виконана, то в силу зазначених у попередньому абзаці причин Z_{med} не може бути імпульсом. У цьому випадку ми переходимо до області Б и перевіряємо, чи є імпульсом значення Z_{xy} у тій точці, що відповідає центру околиці (нагадаємо, що ми будуємо відгук фільтра в цій точці). Якщо умови В1>0 й В2<0 виконані, то Z_{min}- <Z_{xy} <Z_{max}, і значення Z_{xy} не є імпульсним з тих же причин, що й вище. У цьому випадку алгоритм дає на виході незмінене значення Z_{xy}. Збереження значень у таких точках «проміжного рівня» яскравості мінімізує перекручування, внесені обробкою зображення. Якщо одне з умов В1> О і В2 < 0 порушено, то або Z_{xy} = Z_{min}-, або Z_{xy} = Z_{max}. В обох випадках значення є екстремальним, і алгоритм дає на виході значення медіани Z_{med}, що, як треба з результатів роботи області А, не є значенням імпульсного шуму. Остання операція відповідає дії звичайного медіанного фільтра. Відмінність полягає в тім, що звичайний медіанний алгоритм заміняє значення в кожній точці на значення медіани по відповідній області. Це приводить до зайвих спотворень деталей на зображенні.

Хоча медіанні алгоритми, безумовно, належать до числа найбільше часто використовуваних в обробці зображень фільтрів, основаних на порядкових статистиках, це аж ніяк не єдиний приклад таких фільтрів. Медіана являє собою 50

процентиль упорядкованого набору чисел, однак використання інших статистичних характеристик надає багато інших можливостей. Наприклад, використання 100-го процентиля призводить до фільтра, основаного на виборі максимального значення (або фільтру максимуму), що задається виразом.

$$\hat{f}(x,y) = \max_{(s,t) \in S_{xy}} \{g(s,t)\} \quad (4)$$

Такий фільтр корисний при виявленні найбільш яскравих точок на зображенні. Крім того, оскільки уніполярний «чорний» імпульсний шум приймає мінімальні значення, застосування цього фільтра призводить до зменшення такого шуму, тому що в процесі фільтрації з околиці S_{xy}вибирається максимальне значення. Використання 0-го процентиля призводить до використання фільтра, основаного на виборі мінімального значення (або фільтру мінімуму):

$$\hat{f}(x,y) = \min_{(s,t) \in S_{xy}} \{g(s,t)\} \quad (5)$$

Такий фільтр корисний при виявленні найбільш темних точок на зображенні. Крім того, застосування цього фільтра приводить до зменшення уніполярного «білого» імпульсного шуму внаслідок операції вибору мінімуму.

Для вивчення поточного стану питання фільтрації шуму на цифровому зображенні був обраний пошук науково-патентної літератури на українських і іноземних патентних базах

Патент – це джерело технічної інформації, технічних документів, котра містить інформацію про нові рішення у всіх галузях науки. Патентна інформації використовується для багатьох цілей, наприклад для уникнення зайвих витрат для

дослідження того, що вже відомо, моніторинг тенденцій розвитку техніки та науки, оскарження видачі патенту в тих випадках, коли вони знаходиться в колізії з вашим власним патентом. Патентна інформація необхідна науково-дослідним інститутам, державним установам, підприємствам або індивідуальним винахідникам [12]. Існує багато українських патентних баз, наприклад «Укрпатент», «Патентна бібліотека України» або «База патентів України».

Патентна документація – це документи, котрі містять у собі результати науково-технічних досліджень, відомості про власників патентів. Патентним документом вважається офіційно опублікований документ, що містить відомості про результати творчої діяльності людини. Юридичний статус патентних документів зумовлює випередження появи схожої інформації про винохід на час до 5 років.

Для пошуку патентної інформації потрібно ідентифікувати проблему, визначитися з критеріями інформації та обрати дату публікації. Останній пункт не для всіх ресурсів. Метою пошуку патенту є встановлення рівня техніки, визначення обсягу прав власника документу, та визначення умов реалізації прав власника.

Для покращення результатів дослідження, було обрано не тільки науково-патентні бази України, а й зарубіжні, наприклад британські та російські.

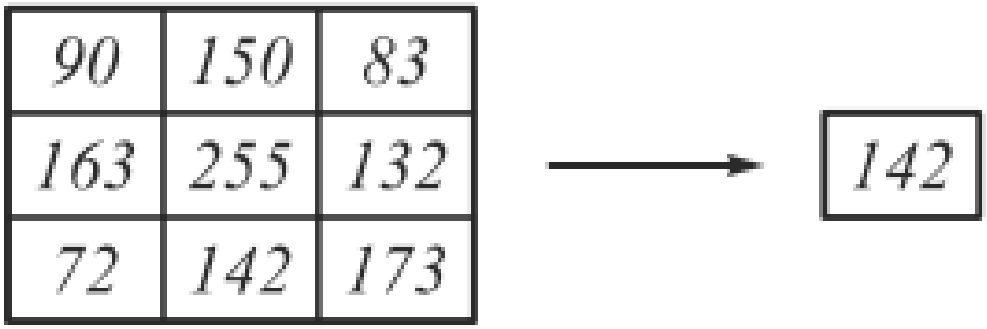
З російського джерела було використано статтю «Два метода адаптивной медианной фильтрации импульсного шума на изображении» [13].

У статті запропоновані два нових методи адаптивної медіанної фільтрації імпульсного шуму в зображеннях. Перший метод заснований на спільному застосуванні ітеративної обробки і перетворення результату медіанної фільтрації на основі розподілу Лоренца. Другий метод використовує альтернативні маски медіанного фільтра, розраховані з використанням метрики Евкліда.

Такий підхід дозволив зменшити розмір оброблюваної області без втрати якості обробки для шумів з низькою інтенсивністю. В експериментальній частині статті наведені результати порівняння якості роботи запропонованих методів з

відомими. Для моделювання були використані 3 різних зображення, спотворені імпульсним шумом з вірогідністю спотворення пікселів від 1% до 99% включно. Чисельна оцінка якості очищення зображень від шуму на основі пікового відношення сигналу до шуму (PSNR) і індексу структурного подібності (SSIM) показала, що запропоновані методи показують кращий результат обробки у всіх розглянутих випадках в порівнянні з відомими підходами.

Найбільш простим нелінійним фільтром для очищення зображень від імпульсного шуму є стандартний медіанний фільтр. Фільтрація на основі СМФ здійснюється за допомогою руху вікна (маски фільтра) уздовж послідовності дискретних відліків з привласненням медіани масиву значень всередині маски результуючому відліку сигналу. Щоб виконати медіанну фільтрацію для пікселя зображення, необхідно спочатку впорядкувати по зростанню значення пікселів всередині околиці, потім знайти значення медіани і, нарешті, привласнити отримане значення оброблюваного елементу. Так, для околиці 3×3 елементів медіаной буде п’яте значення за величиною, для околиці 5×5 - тринадцяте значення і так далі. На рисунку 4 наведено приклад отримання пікселя зображення при застосуванні медіанного фільтра з маскою 3×3.



«Рисунок 4 – Отримання пікселю зображення після вікна 3x3»

У другому методі у статті пропонується використовувати маску фільтра, що представляє собою безліч пікселів, координати яких віддалені від центрального пікселя на відстань, що не більше деякого наперед заданого значення R. На рисунку

5 показана схема розташування навколишніх пікселів з підписаними значеннями квадратів відстані R2 до центрального пікселя, який підписаний як 0.

					25					
			20	17	16	17	20			
		18	13	10	9	10	13	18		
	20	13	8	5	4	5	8	13	20	
	17	10	5	2	1	2	5	10	17	
25	16	9	4	1	0	1	4	9	16	25
	17	10	5	2	1	2	5	10	17	
	20	13	8	5	4	5	8	13	20	
		18	13	10	9	10	13	18		
			20	17	16	17	20			
					25					

«Рисунок 5 - Значення R2 для пікселів з околиці»

В роботі запропоновані два нових методи очищення зображень від імпульсного шуму, які дозволяють поліпшити якість обробки в порівнянні відомими підходами. Чисельна оцінка результатів моделювання на основі PSNR і SSIM дозволяє зробити висновок про те, що розроблені методи краще справляються як з очищенням від шумів з низькою інтенсивністю, так і з очищенням від екстремальних шумів з інтенсивністю 90 - 99%. Візуальне зіставлення результатів обробки дозволяє зробити висновок про те, що запропоновані методи не тільки добре справляються з відновленням перекручених пікселів, а й дозволяють зберегти деталі зображення. Отриманий результат дозволяє вирішувати проблему очищення від імпульсного шуму з більш високою ефективністю.

Отримані у статті результати можуть знайти широке практичне застосування в обробці супутникових і художніх зображень, геофізичних даних і інших додатках цифрової обробки зображень.

Більшість статей, котрі було розглянуто у рамках дослідження мають у собі опис тільки окремого фільтру у конкретній ситуації, наприклад медіанний фільтр на зображенні із супутника з імпульсним шумом. Ця інформація не дає повну картину, коли кожен з фільтрів потрібно використовувати, за яких умов фільтр надає бездоганний результат, а коли потрібно обрати інший.

Оцінка продуктивності методів зниження шуму потребує відповідних повноцінних показників, здатних вимірювати різну кількість залишкового шуму та спотворення фільтру. У цій роботі ми представили новий метод оцінки таких особливостей у кольорових зображеннях, відновлених від імпульсу та гауссового шуму. Підхід заснований на розкладанні на три компоненти, які, відповідно, вимірюють здатність фільтра видаляти шумові імпульси , зменшувати гауссовий шум та зберігати деталі зображення. Ці нові заходи зберігають специфічну перевагу, тобто оцінку помилок кольорів у сприйнятому рівномірному кольоровому просторі. З іншого боку, вони долають обмеження підошви, яка не може розрізнити кольорові помилки через спотворення фільтру та недостатнє зменшення шуму.

Результати комп’ютерного моделювання, що стосуються різних зображень, пошкоджених імпульсом та білим шумом, показали, що запропонований метод перевершує класичні та векторні метрики в літературі при оцінці різної кількості залишкового шуму та спотворень, заданих фільтрами.

У більшості статей описано різні методи середньої фільтрації та запропоновані нові методики, котрі дають найкращі результати серед усіх доступних методик. Результати показують, що середня методика фільтрації на основі прийняття рішень дає кращі показники для зображення, що позначає імпульсивний шум на основі PSNR та наочності. Для дуже високої щільності шуму

може бути використана серединна фільтрація на основі адаптивного рішення, щоб отримати кращу чіткість зору та значення PSNR, ніж середня методика фільтрації на основі прийняття рішень. Звичайна адаптивна серединна фільтрація також дає хорошу чіткість зору, позначаючи імпульсивний шум для густини шуму до 50%. Однак, порівняно з цими методами, прогресивне перемикання та адаптивне прогресивне перемикання демонструють хороші показники для низької та середньої щільності шуму, але займають більше часу на обчислення, а значить, не застосовуються для додатків у режимі реального часу. Таким чином, в цілому середня техніка фільтрації на основі прийняття рішень та методика серединної фільтрації на основі адаптивного рішення є кращими методами, ніж будь-які інші описані вище методи фільтрації імпульсних шумів, і час обчислення цих методів значно менше, що робить їх ідеальною технікою для використання в реальному час застосування. Однак для більшої щільності шуму пропонована методика перевищує техніку, що базується на рішеннях, на основі продуктивності. Але якщо щільність шуму значно вища, то нові алгоритми повинні бути розроблені.

2 ОПИС ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1 Опис шуму

Як правило зменшення шуму служить для поліпшення візуального сприйняття, але цей процес використовується для специфічних потреб, наприклад у медицині для покращення зображення на знімках з мікроскопу чи рентгенівських знімках для предобробки для подальшого розпізнавання [2]. Слід зауважити, що зменшення шуму грає важливу роль при стисненні зображень, адже зараз використовується багато сервісів для зберігання й передачі цифрового зображення. Наприклад Instagram – це соціальна мережа для обміну фотокартками, де можна ділитися й переглядати, коментувати та ставити «подобається» фотокарткам інших користувачів цього сервісу. Алгоритми цього сервісу не є досконалими, тому при завантаженні фотографії сервіс перевіряє зображення на вагу та кількість пікселей за висотою та ширині. Якщо кількість пікселей більш ніж 1080 або розмір зображення більш ніж 1.5 мегабайтів, то сервіс прийме рішення стиснути його. Адже у сервіс щодня завантажується понад вісімдесят мільйонів зображення й на це потрібно виділяти достатньо багато пам’яті. Наприклад, якщо взяти середньо статистичну фотокамеру Nikon D5100 та фотографувати у режимі RAW, тобто без втрати кольорової інформації, то розмір зображення буде близько 15-20 мегабайтів при роздільної здатністю зображення 4928 х 3264 пікселей, то потребується у 13 разів більше простору на сервері для збереження зображень без стиснення. З’являється необхідність в стисненні зображення, що за собою тягне спотворення у вигляді зашумлення та пікселизації. Порівняємо два зображення до та після завантаження у Instagram, дивиться рисунок 6.

«Рисунок 6 – Порівняння зображень до та після»

Джерела шуму можуть бути різними, наприклад:

– недосконале обладнання для захоплення зображення (фотокамера, сканер та інше);

– погані умови зйомки, нічна фотозйомка чи зйомка при недостатньому світлі на неповно розмірну матрицю фотокамери;

– перешкоди при передачі по аналоговим каналам – електромагнітні поля, власні шуми активних компонентів (підсилювачі) лінії передачі.

Слід зауважити, що шуми також бувають різних видів [3]. На практиці найчастіше використовують модель адитивної Гауссового та імпульсного шуму. Адитивний Гаусів шум характеризується додаванням до кожного пікселя зображення значень з відповідного нормального розподілу з нульовим середнім значенням. Такий шум зазвичай вводиться на етапі формування цифрових зображень. Імпульсний шум характеризується заміною частини пікселів на зображенні значеннями фіксованою або випадкової величини. Така модель шуму пов’язана з помилками при передачі зображень.

Алгоритми зменшення шуму зазвичай спеціалізуються на придушенні якогось конкретного виду шуму. На зараз не існує фільтрів, котрі можуть зменшити всі види шумів. Однак найбільш поширений вид шуму – це білий Гауссовий. Приклади шуму дивиться на рисунку 7, 8 та 9, де рисунок 7 – це оригінальне зображення без додавання шуму [2].

«Рисунок 7 – Оригінальне зображення»

Оригінальне зображення зроблено взимку, при рівномірному розподілу світла у кадрі, немає перегріву світлочутливої матриці фотоапарату, що сприяє гарному результату для цифрового зображення без шуму.

Завдяки фото редактору Photoshop додано до зображення білий гаусівский шум (рисунок 3).

«Рисунок 8 – Зображення з додаванням Гаусівського білого шуму»

«Рисунок 9 – Зображення з додаванням медіанного шуму»

Розмір датчика камери відіграє велику роль у кінцевій якості зображення, включаючи рівень шуму на фотографії. Датчик вашої камери містить мільйони пікселів або чутливих до світла крапок, які використовуються для збору та запису інформації, що надходить через об’єктив камери. Більший датчик має можливість збирати більше інформації. Тому чим більший датчик вашої камери, тим краща якість зображення. Різниця у розмірах датчиків є однією з головних причин, чому камери з кропнутими датчиками створюють зображення з більшим рівнем шуму при підвищених ISO, ніж повнокадрові камери. У ідеальному світі ви будете працювати із зображенням з відмінним балансом білого, низьким ISO та хорошою експозицією. Якщо цього не відбувається, розуміння того, що викликає шум на зображенні, допоможе вам протидіяти його афектам, а також уникне ненавмисного введення шуму в зображення, або розуміння природи появлення шуму допоможе обрати коректний фільтр, котрий зменшить кількість цього шуму на зображенні. Також слід пам’ятати, що для деяких зображень шум можна насправді додати до загального відчуття та настрою зображення, будь то заради текстури або для створення ефекту старовинного фільму.

Отже шум зображення це випадкова яскравість або кольорова інформація в зображеннях і, як правило, з аспектом електронного шуму. Він може вироблятися датчиком зображення та схемою сканера або цифровою камерою. Шум зображення також може виникати у зерні плівки та у неминучому шумі ідеального фотонного детектора. Шум зображення - небажаний побічний продукт зйомки зображення, що затьмарює бажану інформацію. Також ми розглянули причини виникнення шуму, найчастіше це недосконале обладнання, чи погана обробка зображення, котрі приводять до появлення білого або імпульсного шуму, у разі проблеми з обладнанням.

2.2 Види фільтрів

Найпоширеніші методи видалення шуму:

– згладжуючі фільтри;

– фільтр Вінера;

– медіанний фільтр;

– kuwahara;

– двосторонній фільтр.

Для зменшення Гауссова шуму використовуються як лінійні так и нелінійні алгоритми фільтрації. Лінійні алгоритми визначаються функцією заданої на растрі. Фільтрація проводиться за допомогою операції дискретної згортки. При лінійної фільтрації значення інтенсивності у кожній точці усередняється за деякою згладжувальною маскою. Приклад масок:

$$A_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, A_2 = \frac{1}{10}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}, A_3 = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \quad (6)$$

У першому випадку значенням інтенсивності в центральній точці присвоюється середнє значення інтенсивності сусідніх значень. В інших прикладах використовується середнє зважене з коефіцієнтами.

2.2.1 Фільтр Вінера

Гарні результати обробки цифрового зображення досягаються при використанні алгоритму Вінера. Застосування цього алгоритму пов’язане з припущенням про стаціонарність зображення. Технічно алгоритм Вінера реалізується за допомогою дискретного перетворення Фур’є в частотній області.

З метою збільшення кількості задач, котрі можна вирішити засобами цифрової обробки зображень, в даний час активно впроваджується методи нелінійної цифрової фільтрації. Розглянемо фільтрацію зображень, пошкоджених сигналом, залежним від нульового середнього білого шуму. Проблему можна моделювати як:

$$y(i,j) = x(i,j) + n(i,j) \tag{7}$$

де $y(i,j)$ – рівень шуму, $x(i,j)$ – безшумне зображення, $n(i,j)$ – адитивний шум Гауса. Мета полягає в тому, щоб видалити шум, або «позначити» у (i, j), і отримати лінійну оцінку $x(i,j)$х $x(i,j)$яка мінімізує середню квадратичну помилку - MSE.

$$MSE(\hat{x}) = \frac{1}{N}\sum_{i,j=1}^{N}(\hat{x}(i,j) - x(i,j))^2 \tag{8}$$

де N – кількість елементів у $x(i,j)$.

Залежно від характеру програми деякі практичні системи зменшення шуму вимагають дуже якісного зображення, але можуть переносити певну кількість залишкового шуму, тоді як інші системи вимагають, щоб зображення було

максимально чистим, але може дозволити певну ступінь спотворення зображення. Тому необхідно мати певну схему управління для контролю компромісу між зменшенням шуму та спотворенням зображення в контексті фільтрації Вінера. Існує три підходи. Перший підхід призводить до оптимального фільтра, де вводиться параметр для управління компромісом між спотворенням зображення і зниженням шуму. Другий підхід призводить до відомої методики зменшення шуму на основі параметричних моделей, де модель AR використовується для досягнення зниження шуму, зберігаючи низький рівень спотворення зображення. Третій підхід стосується багатоканального підходу, де використовуються просторово-часові фільтрувальні методи для отримання шумового зменшення з меншим або навіть відсутнім спотворенням зображення.

Алгоритм Вінера дійсно працює. він має властиву залежність між зменшенням шуму та спотворенням зображення. Починаючи з оцінки шуму, використовуючи теорію Вінера, ввели індекс спотворення зображення та два коефіцієнти зменшення шуму і показали, що для одно алгоритму Вінера кількість ослаблення шуму взагалі пропорційна моменту деградація зображення, тобто зменшення рівня шуму спричиняє більше спотворень зображення.

2.2.2 Медіанний фільтр

Медіанний фільтр на відміну від фільтра, що згладжує є нелінійною процедурою зменшення кількості шумів на зображенні. Цей алгоритм являє собою ковзне по зображенню так зване вікно. Центральний відлік замінюється медіаною всіх елементів зображення, що потрапили у вікно.

Медіанний фільтр використовується для зменшення адитивного та імпульсного шумів на цифровому зображенні. Особливістю медіанного фільтра є

збереження перепадів яскравості, тобто контурів. Максимально ефективно можна використовувати у випадку імпульсного шуму.

Приклад використання медіанного фільтру на монохромного зображенні з імпульсним шумом можна подивитись на рисунку 10, де зверху – оригінальне зображення, а знизу – після використання фільтру [5].

«Рисунок 10 – Медіанний фільтр»

На рисунку 10, дуже добре відпрацював медіанний алгоритм, деталізація після котрого залишилась на високому рівні, у секторах перепаду зі світла на тінь імпульсний шум було зменшено. Зображення було отримано на «Nikon D80», з об’єктивом «Nikkor 18-135mm», котрий є не дуже світлочутливим та у ситуації з недостатньою освітленістю потрібно використовувати підвищення ISO до значення 1250 та збільшену витримку, близько до 15-20 секунд. Що у сукупності з не повно кадровою матрицею дає результат, як на оригінальному зображенні.

Стосовно імпульсного шуму, наприклад медіанний алгоритм з «вікном» 3x3, повністю зменшує поодинокі та групи з двох, трьох чи чотирьох імпульсних випадків шуму. У загалі для зменшення групи імпульсного шуму розмір «вікна» повинен бути щонайменше вдвічі більше розмірів групи імпульсного шуму.

Узагальнення розглянутого медіанного алгоритму є зважений медіанний алгоритм. У простому медіанному алгоритмі всі елементи зображення в межах «вікна» однаково впливають на результат знаходження медіани. У деяких випадках бажано надати більшої ваги точкам, що розташовано ближче до центру «вікна».

Ідея виваженого медіанного алгоритму полягає в зміні числа елементів, що потрапили у «вікно», шляхом повторення кожного елемента заданого числа та знаходження медіани - виходить розтягнута послідовність. Приклади медіанних алгоритмів з «вікном» 3х3:

$$W_1 = \begin{matrix} 0 & 1 & 0 \\ 1 & 3 & 1, \\ 0 & 1 & 0 \end{matrix} \quad W_2 = \begin{matrix} 1 & 1 & 1 \\ 1 & 3 & 1, \\ 1 & 1 & 1 \end{matrix} \quad W_3 = \begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1, \\ 0 & 1 & 0 \end{matrix} \quad W_4 = \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \tag{9}$$

Прості медіанні алгоритми в загальному випадку зменшують імпульсні шуму більшої контрастності, ніж зважені медіанні фільтри. Якщо на зображенні присутні дрібні деталі розміром 3х3 і менш, вони можуть бути зменшені простими медіанними алгоритмами. Слід зауважити, що зважені медіанні алгоритми менше змінюють форму об’єктів, не деформує крайні межі.

Можливо ітеративне використання медіанного алгоритму на відміну від алгоритму, що згладжує результат ітерацій не обов’язково збігається до зображення з постійною яскравістю. Іноді доцільно використовувати наступний метод: ітеративно застосувати медіанний алгоритм поки не вийде зображення, інваріантне до даного медіанному алгоритму. Вихідне зображення можна називати стабільною точкою медіанного алгоритму.

Метод усереднення пікселів по сусідах:

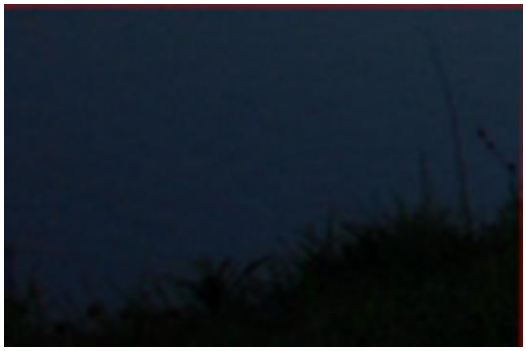
$$f(x,y) = \frac{1}{m \times n} \sum_{(s,t) \in S_{xy}} g(S,t), \tag{10}$$

де $f(x,y)$ – точка обробленого зображення, S_{xy} – прямокутне «вікно» розміром m×n з центром у точці (x,y), $g(x,y)$ – точка оригінального зображення.

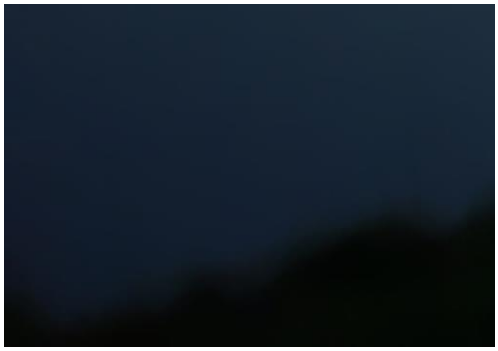
Приклад використання моделі усереднення пікселів по сусідах можна подивитися на рисунках 11, 12 та 13, де 11 – виділений кут оригінального зображення, а 8 – виділений кут після методу усереднення пікселів по сусідах.

Оригінальне зображення з виділеним куточком для візуалізації роботи методу усереднення дивиться на рисунку 11.

«Рисунок 11 – Оригінальне зображення»



«Рисунок 12 – Виділений кут оригінального зображення»



«Рисунок 13 – Зображення після моделі усереднення»

2.2.3 Фільтр Kuwahara

Фільтр Kuwahara – це нелінійний фільтр для згладжування, котрий використовують для обробки зображень для зменшення шуму. Більшість фільтрів, котрі використовують для згладжування зображень також згладжують краї, але цей фільтр здатний зменшити шум, зберігаючи краї. Він діє аналогічно розмиттю «вікна», за винятком того, зо ми беремо це «вікно» навколо центрального пікселя – ядра, і ділимо його на чотири менші «віконця». Ці менші «віконця» обчислюються таким же чином, як и розмиття «вікна» для середнього значення. Поки ми робимо це, ми паралельно робимо пошук варіації для «вікна». Нам потрібно знайти, у котрому «віконці» буде щонайменша кількість дисперсії. У метод передається само зображення та розмір діафрагми, котрий нам потрібен. У свою чергу це дає нам згладжене зображення. Однією з найбільших недоліків середнього фільтра є розмивання шуму та значна модифікація правильних пікселів. Припустимо, що ми побудуємо фільтр, що працює на вікні N×N. У цьому випадку отримане значення пікселя є середнім значенням усіх пікселів, що містяться у поточному вікні аналізу вихідного зображення.

$$m_\theta \frac{1}{N \times N} \times \sum_{(x,y)\in\theta} \vartheta(f(x,y)) \tag{11}$$

де f – функція, $f(x,y)$ – значення пікселя за координатами (x,y), ϑ – функція, що обчислює значення певного пікселя. Ця функція може приймати різні значення залежно від кольорового простору, формату або глибини зображення, $N \times N$ – кількість пікселів у поточному вікні, θ – представляє колекцію пікселів у поточному вікні. Наприклад:

$$m_k = \frac{1}{(N+1)\times(N-1)} \times \sum_{(x,y)\in\theta_k}\left[\vartheta\left(f(x,y)\right) - m_k\right]^2 \quad (12)$$

де $k \in \{0,1,2,3\}$, f – функція вихідного зображення, $f(x,y)$ – значення пікселя за координатами (x,y), ϑ – функція, що обчислює значення певного пікселя, $\frac{1}{(N+1)\times(N-1)}$ – кількість пікселів у поточній області, n – значення, отримане безпосередньо з розміру вікна фільтра.

2.2.4 Двосторонній фільтр

Двосторонній фільтр – це різновид нелінійних алгоритмів, котрі використовуються не тільки для згладжування з зберіганням країв, але й для зменшення кількості шуму на цифровому зображенні. Цей алгоритм замінює інтенсивність кожного пікселя середньозваженими значеннями інтенсивності від пікселів, котрі розташовані поблизу центру. Слід зауважити, що вага залежить не тільки від евклідової відстані пікселів, а й від інтенсивності кольору, глибини відстані та інше. Після використання цього методу на зображенні зберігаються гострі краї. Приклад використання алгоритму зображено на рисунках 14 та 15.

«Рисунок 14 – Оригінальне зображення до двостороннього фільтру»

Після використання двостороннього алгоритму на зображенні з рисунку 14, ми отримуємо на виході зображення, на котрому можна явно побачити зменшену кількість шумів, але слід зауважити, що зображення отримало деяку акварельну рису, тобто занадто м’які контури перепаду освітленості, приклад можна побачити на рисунку 15.

«Рисунок 15 – Зображення після накладання двостороннього фільтру»

Щоб визначитися з розміром «вікна» для кожного з методів застосовуються дві найголовніші ідеї:

– перша ідея використовує наближенні зміщення і дисперсії оцінки сигналу з відповідним обчисленням оптимізованим розміром «вікна». Але для цього потрібно знати похідну даного сигналу. Отже, стає необхідним оцінювати і ці похідні за допомогою другорядних «вікон». Такий підхід є досить складним для реалізації;

– друга ідея не використовує оцінку зміщення. Група методів заснована на статистиці якості фільтрації, котра використовується для прямої оптимізації точності.

Ціль алгоритму полягає у знаходженні околиці точки оцінки. Оцінка обчислюється для декількох різних параметрів і порівнюється. Адаптивний параметр визначається як найбільший, для котрого оцінка не значно відрізняється

від оцінок для менших параметрів. Розглянемо зашумлені спостереження z^r такого виду:

$$z(x) = y(x) + \varepsilon(x), x\epsilon X, \tag{13}$$

де $x(s) = (x_1(s), x_2(s))$ – двовимірний вектор з компонентами $x_1(s)$ и $x_2(s)$, де параметр s відповідає s –тому пікселю на зображенні.ε – нелазежні однаково розподілені випадкові гаусові помилки, їх розподіл має вигляд $N(0, \sigma^2)$, а X – таблиця, $y: X \to R$ – функція інтенсивності вихідного чорно-білого зображення.

Передбачається, що функція $y(x)$ – частково безперервна і належить класу F^r:

$$F_r = \left|D^r_{r_1,r_2}\, y(x)\right| \le L_r(x) \le L_r, \forall r_1 + r_2 = r, \tag{14}$$

де $D^{k_1+k_2}_{r_1,r_2} = \frac{\delta^{k_1+k_2}}{\delta^{r_1}x_1\delta^{r_2}x_2}$- оператор диференціювання, а L_r – константа.

Відомо, що вибір розміру «вікна» - ключовий момент ефективності локальної оцінки. Коли h - мало, тоді цей метод дає гарний наближення $y(x)$. Але тоді використовується менше даних і оцінка біль чутлива до шуму на зображенні. Отже, якщо h велике, то даних використовується багато і зображення буде занадто розмите. Наведені алгоритми дозволяють ефективно видаляти гаусів шум з зображення. Але якщо застосувати його до зображення, котре має імпульсний шум, то воно залишиться беззмінним. При цьому метод дозволяє для кожного пікселю знайти околицю, в котрій піксель не відрізняється від центру «вікна». Імпульсний шум на цифровому зображенні, котрий виникає у разі битих пікселів, вони виникають у результаті пошкодження світлочутливих поверхонь, таки як матриця цифрового фотоапарату, сканер або фотоплівка. Якщо битих пікселів на зображені дуже багато, то сусідство таких пікселів дуже ймовірно і тоді околиця буде більше.

У разі коли алгоритм повинен бути застосований для різних видів шумів, буде некоректно видаляти гаусів шум з цифрового зображення, котрий має тільки імпульсний шум. Тоді потрібно вбудувати до алгоритму елементи адаптивності високого рівня, котрі прискорюють його роботу. При обчислюванні дисперсії шуму наявність битих пікселів впливає на цю величину. На випадок якщо гаусів шум на цифровому зображенні буде відсутній, то дисперсія буде відмінна від нуля. При видаленні гаусівського шуму підраховується дисперсія зменшеного шуму, якщо вона виявляється значно менше ніж обчислена раніше, то дисперсія коригується, у такому разі, після коригування, алгоритм починає працювати більш точно.

У другому розділи ми ознайомились з видами шуму, з такими як гаусівський шум та імпульсний шум. Причинами виникнення того чи іншого шуму, у разі небездоганного обладнання, перешкод передачі даних чи пошкодження світлочутливої матриці фотоапарату. Також ми ознайомились з видами алгоритмів: Вінера, медіанний. Кувахара та двосторонній.

Неможливо використовувати один алгоритм, як універсальний під кожну ситуацію, тому потрібно уважно підійти до вибору алгоритму зменшення шуму. Для кольорового зображення з білим шумом рекомендовано використовувати медіанний фільтр, котрий дасть гарний результат видалення білого шуму без великого рівня розмиття зображення. Найпоганіший результат дав двосторонній алгоритм. У разі з видаленням шуму він впорався непогано, але на виході ми отримали дуже розмите зображення. У дослідженні алгоритмів на зображенні з імпульсним шумом було виявлено, що двосторонній алгоритм впорався з завданням фільтрації шуму бездоганно, він видалив шум повністю, на зображенні з менш кольоровим насиченням, у свою чергу медіанний алгоритм не видалив шум з зображення навіть на 50 відсотків і дав велике розмиття по краям зображення.

Тому, для насиченим кольорами зображення з білим шумом рекомендовано використовувати медіанний алгоритм, а для зображення з менш насиченим кольорами та з імпульсним шумом слід використовувати двосторонній алгоритм.

3 ОПИС ТЕХНОЛОГІЇ. ВІЗУАЛІЗАЦІЇ МЕТОДІВ Й АЛГОРИТМІВ

3.1 Алгоритми та методи

Для візуалізації роботи моделей й алгоритмів фільтрації шуму на цифровому зображенні було обрано бібліотеку з відкритим вихідним кодом $OpenCV$ [7].

$OpenCV$ - це бібліотека, інструменти котрої використовуються для роботи з комп'ютерним зором. Окрім комп'ютерного зору, цю бібліотеку можна використовувати для обробки зображень, завдяки універсально реалізованим алгоритмам, у нашому випадку для фільтрації шуму на цифровому зображенні. $OpenCV$ першу релізну версію було випущено у дві тисячі шостому році компанією $Intel$. Ядро бібліотеки реалізовано на $C++$. Так як бібліотека має відкритий вихідний код, її можна використовувати в академічних й комерційних цілях.

Наприклад, для реалізації двостороннього методу фільтрації цифрового зображення від шуму використовується метод
void bilateralFilter(InputArray src, OutputArray dst, int d, double sigmaColor, double sigmaSpace,
int borderType = BORDER_DEFAULT),

де src - це джерело, восьми бітне, одно канальне або трьох канальне зображення;

dst - це зображення того ж розміри й типу, що й src;

d - це діаметр кожного піксельної околиці, котра використовується під час фільтрації;

$sigmaColor$ - це фільтр сигми в кольоровому просторі. Більш велике значення параметра означає, що дальші кольори в межах піксельного сусідства будуть змішуватися разом, у результаті чого більші площі мають піврівні кольору;

$sigmaSpace$ - це фільтр сигми у просторі координат. Більше значення цього параметра означає, що дальші пікселі будуть впливати один на одного, поки їх

кольори будуть досить близькими. Коли $d > 0$, він визначає розмір околиці незалежно від $sigmaSpace$. У інших випадках d пропорційний $sigmaSpace$.

Для візуалізації моделей й алгоритмів з бібліотеки $OpenCV$ було використано $IDE\ Visual\ studio$. $Visual\ studio$ – це середовище розробки програм, цей продукт дозволяє робити розробку не тільки консольні програми, а й програмні продукти з використанням графічного інтерфейсу, такі як веб-застосування, виндовс застосування. У нашому випадку було використано Emgu CV – як крос-платформне $.Net$ доповнення до бібліотеки $OpenCV$ для обробки цифрового зображення за допомогою мов програмування: $C\#,\ VB,\ VC\ ++,\ IronPython$ [6].

```
public static void FastNlMeansDenoisingColored(
        IInputArray src,
        IOutputArray dst,
        float h = 3f,
        float hColor = 3f,
        int templateWindowSize = 7,
        int searchWindowSize = 21
)
```

Це приклад методу для зменшення білого гаусівського шуму на кольоровому цифровому зображенні з бібліотеки Emgu CV.

Наступний фрагмент коду надає визначення вихідного методу $MedianFilter$ можна знайти на рисунку 16, котрий використовується для медіанного фільтру. Зверніть увагу на визначення трьох окремих байтових масивів, кожен з яких повинен представляти значення пікселів сусідства пікселів, пов’язані з певним кольоровим каналом. Кожен масив байтів кольорового каналу сусідства повинен бути відсортований за значенням. Значення, розташоване в індексі масиву рівно на півдорозі від початку та кінця масиву, представляє медіанне значення. Коли визначено медіанне значення, потрібно встановити піксель буфера результатів, пов’язаний з піксельним буфером джерел, з точки зору X та Y.

```
private static byte[] MedianFilter(this byte[] pixelBuffer,
                                   int imageWidth,
                                   int imageHeight,
                                   int filterSize)
{
    byte[] resultBuffer = new byte[pixelBuffer.Length];

    int filterOffset = (filterSize - 1) / 2;
    int calcOffset = 0;
    int stride = imageWidth * pixelByteCount;

    int byteOffset = 0;
    var neighbourCount = filterSize * filterSize;
    int medianIndex = neighbourCount / 2;

    var blueNeighbours = new byte[neighbourCount];
    var greenNeighbours = new byte[neighbourCount];
    var redNeighbours = new byte[neighbourCount];

    for (int offsetY = filterOffset; offsetY <
        imageHeight - filterOffset; offsetY++)
    {
        for (int offsetX = filterOffset; offsetX <
            imageWidth - filterOffset; offsetX++)
        {
            byteOffset = offsetY *
                         stride +
                         offsetX * pixelByteCount;

            for (int filterY = -filterOffset, neighbour = 0;
                filterY <= filterOffset; filterY++)
            {
                for (int filterX = -filterOffset;
                    filterX <= filterOffset; filterX++, neighbour++)
                {
                    calcOffset = byteOffset +
                                 (filterX * pixelByteCount) +
                                 (filterY * stride);

                    blueNeighbours[neighbour] = pixelBuffer[calcOffset];
                    greenNeighbours[neighbour] = pixelBuffer[calcOffset + greenOffset];
                    redNeighbours[neighbour] = pixelBuffer[calcOffset + redOffset];
                }
            }

            Array.Sort(blueNeighbours);
            Array.Sort(greenNeighbours);
            Array.Sort(redNeighbours);

            resultBuffer[byteOffset] = blueNeighbours[medianIndex];
            resultBuffer[byteOffset + greenOffset] = greenNeighbours[medianIndex];
            resultBuffer[byteOffset + redOffset] = redNeighbours[medianIndex];
            resultBuffer[byteOffset + alphaOffset] = maxByteValue;
        }
    }
    return resultBuffer;
}
```

«Рисунок 16 – Метод "*MedianFilter*"»

Приклад алгоритму коду, котрий використовується у алгоритмі *Kuwahara* дивиться рисунок 17.

```
public static Bitmap Kuwahara(Bitmap image, int size)
{     System.Drawing.Bitmap TempBitmap = image;
    System.Drawing.Bitmap NewBitmap = new System.Drawing.Bitmap(TempBitmap.Width, TempBitmap.Height);
    System.Drawing.Graphics NewGraphics = System.Drawing.Graphics.Fromimage(NewBitmap);
    NewGraphics.Drawimage(TempBitmap, new System.Drawing.Rectangle(0, 0, TempBitmap.Width, TempBitmap.Height),
    new System.Drawing.Rectangle(0, 0, TempBitmap.Width, TempBitmap.Height), System.Drawing.GraphicsUnit.Pixel);
    NewGraphics.Dispose(); Random TempRandom = new Random();
    int[] ApetureMinX = { -(Size / 2), 0, -(Size / 2), 0 };     int[] ApetureMaxX = { 0, (Size / 2), 0, (Size / 2) };
    int[] ApetureMinY = { -(Size / 2), -(Size / 2), 0, 0 };     int[] ApetureMaxY = { 0, 0, (Size / 2), (Size / 2) };
    for (int x = 0; x < NewBitmap.Width; ++x)
    {
        for (int y = 0; y < NewBitmap.Height; ++y)
        {
            int[] RValues = { 0, 0, 0, 0 };             int[] GValues = { 0, 0, 0, 0 };
            int[] BValues = { 0, 0, 0, 0 };             int[] NumPixels = { 0, 0, 0, 0 };
            int[] MaxRValue = { 0, 0, 0, 0 };             int[] MaxGValue = { 0, 0, 0, 0 };
            int[] MaxBValue = { 0, 0, 0, 0 };             int[] MinRValue = { 255, 255, 255, 255 };
            int[] MinGValue = { 255, 255, 255, 255 };             int[] MinBValue = { 255, 255, 255, 255 };
            for (int i = 0; i < 4; ++i)
            {
                for (int x2 = ApetureMinX[i]; x2 < ApetureMaxX[i]; ++x2)
                {
                    int TempX = x + x2;
                    if (TempX >= 0 && TempX < NewBitmap.Width)
                    {
                        for (int y2 = ApetureMinY[i]; y2 < ApetureMaxY[i]; ++y2)
                        {
                            int TempY = y + y2;
                            if (TempY >= 0 && TempY < NewBitmap.Height)
                            {
                                Color TempColor = TempBitmap.GetPixel(TempX, TempY);
                                RValues[i] += TempColor.R;                                  GValues[i] += TempColor.G;
                                BValues[i] += TempColor.B;
                                if (TempColor.R > MaxRValue[i])
                                {
                                    MaxRValue[i] = TempColor.R;
                                }
                                else if (TempColor.R < MinRValue[i])
                                {
                                    MinRValue[i] = TempColor.R;
                                }
                                ++NumPixels[i];
                            }
                        }
                    }
                }
            }
            int j = 0; int MinDifference = 10000;
            for (int i = 0; i < 4; ++i)
            {
                int CurrentDifference = (MaxRValue[i] - MinRValue[i]) + (MaxGValue[i] - MinGValue[i]) + (MaxBValue[i] - MinBValue[i]);
                if (CurrentDifference < MinDifference && NumPixels[i] > 0)
                {
                    j = i;                    MinDifference = CurrentDifference;
                }
            }
            Color MeanPixel = Color.FromArgb(RValues[j] / NumPixels[j], GValues[j] / NumPixels[j], BValues[j] / NumPixels[j]);
            NewBitmap.SetPixel(x, y, MeanPixel);
        }
    }
    return NewBitmap;
}
```

«Рисунок 17 – Приклад алгоритму *Kuwahara*»

Для згладжування та зменшення шуму на цифровому зображенні було реалізовано алгоритм *Median*. Він широко використовується у алгоритмах пошуку країв чи кутів, тому що при деяких умовах воно дає можливість зберегти кути, при цьому дуже ефективно видалив шум.

Приклад алгоритму для $Median$ фільтру можна знайти на рисунку 18.

```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui_c.h>

using namespace cv;

int main(int argc, char** argv)
{
   namedWindow("Before" , CV_WINDOW_AUTOSIZE);

   // Load the source image
   Mat src = imread( "../dataset/mniriTest.jpg", 1);

   // Create a destination Mat object
   Mat dst;

   // display the source image
   imshow("Before", src);

   for (int i=1; i<51; i=i+2)
   {
      medianBlur( src, dst, i );

      imshow( "Median filter", dst );

      waitKey(5000);
   }
}
```

«Рисунок 18 - $Median$ алгоритм»

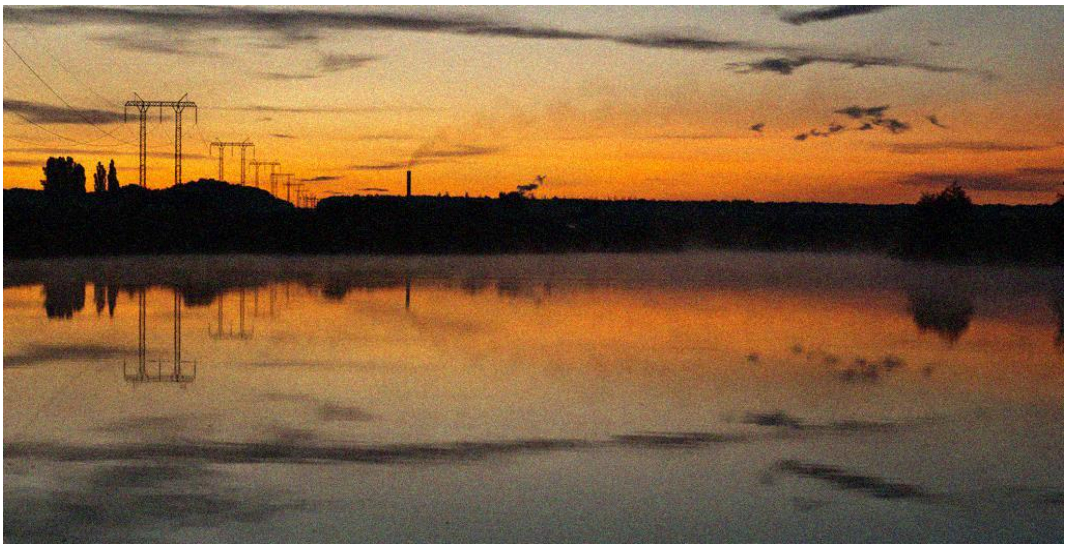
3.2 Порівняння алгоритмів на білому гаусівському шуму

Для порівняння усіх наведених алгоритмів було взято цифрове зображення пейзажу з гарною деталізацією та з належною якістю, приклад цього зображення дивиться на рисунку 19. Як ми можемо бачити, на даному вихідному зображенні немає шумів, гарна деталізація, дуже багато контрастних зон й була використана

маленька діафрагма, що забезпечую чіткість й деталізацію на усій фокусній відстані.

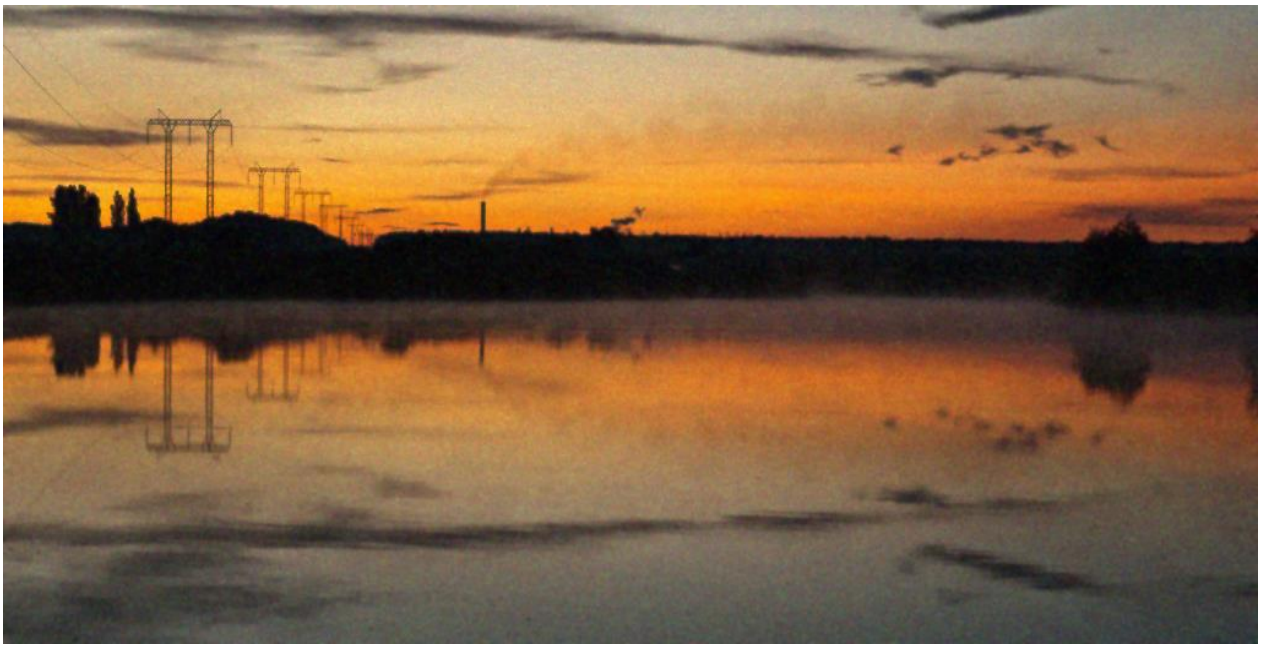
«Рисунок 19 – Зображення для порівняння алгоритмів»

Далі до оригінального зображення було навмисно додано гаусівський білий шум за допомогою фото редактору Photoshop. На рисунку 20 ми отримали досить спотворене зображення, де втратили велику кількість інформації, такі як малюнки на поверхні води та детальність на горизонті й у далі.



«Рисунок 20 – Тестове зображення з додаванням шуму»

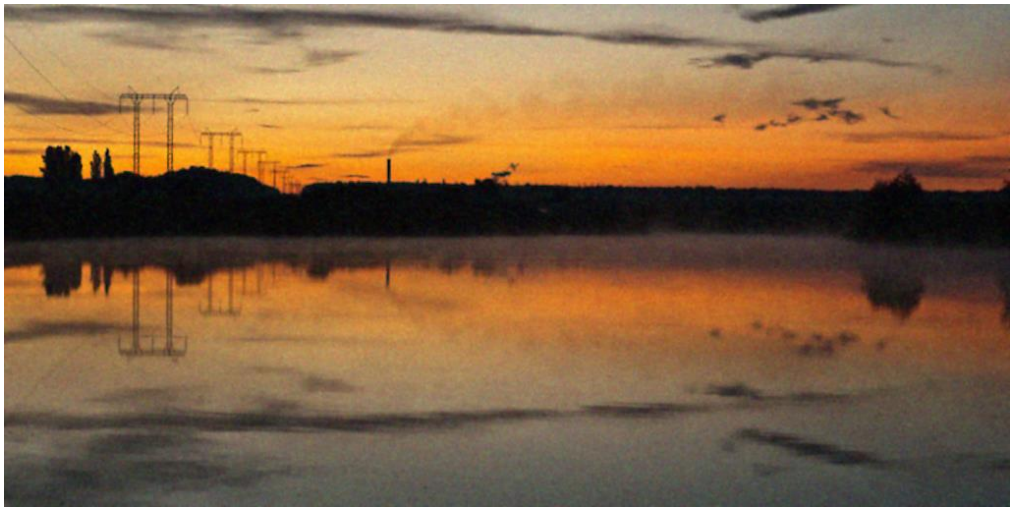
Першим алгоритмом для дослідження було обрано алгоритм $Median$, котрий добре зберігає межі елементів й володіє швидкодією. Але дає велике розмиття на кольоровому зображенні, приклад зображено на рисунку 21.



«Рисунок 21 – Тестове зображення з використанням алгоритму $Median$»

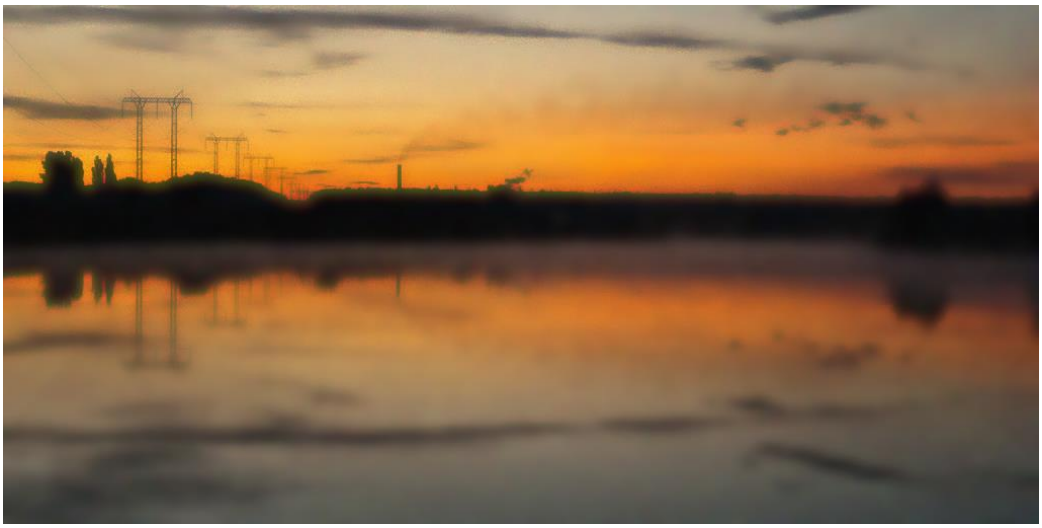
Наступним алгоритмом для дослідження розглянемо алгоритм Kuwahara. Цей алгоритм дозволяє більш ефективно зменшити шум без розмивання країв та підготовки зображення до сегментації та подальшого аналізу операцій. Один за найбільших обмежень цього алгоритму є необхідність декларативного визначення вікна фільтра, розмір або кількість повторень, якими повинна бути операція повторена. У випадку запропонованого рішення ми маємо справу з автоматичною адаптацією алгоритму до локального оточення кожного пікселя в оброблюваному зображенні. Так як контур відіграє дуже важливу роль у процесі аналізу зображень, це дуже важливо стежити, щоб розгладження зображення не впливало на різкість контурів. Це може спричинити серйозні проблеми пізніше, під час сегментації.

Цей алгоритм може бути побудований для будь-якого розміру вікна. Для оптимального результату було використано алгоритм з розміром «вікна» 3x3, приклад зображення після обробки алгоритмом Kuwahara зображено на рисунку 22.



«Рисунок 22 – Зображення після використання алгоритму Kuwahara»

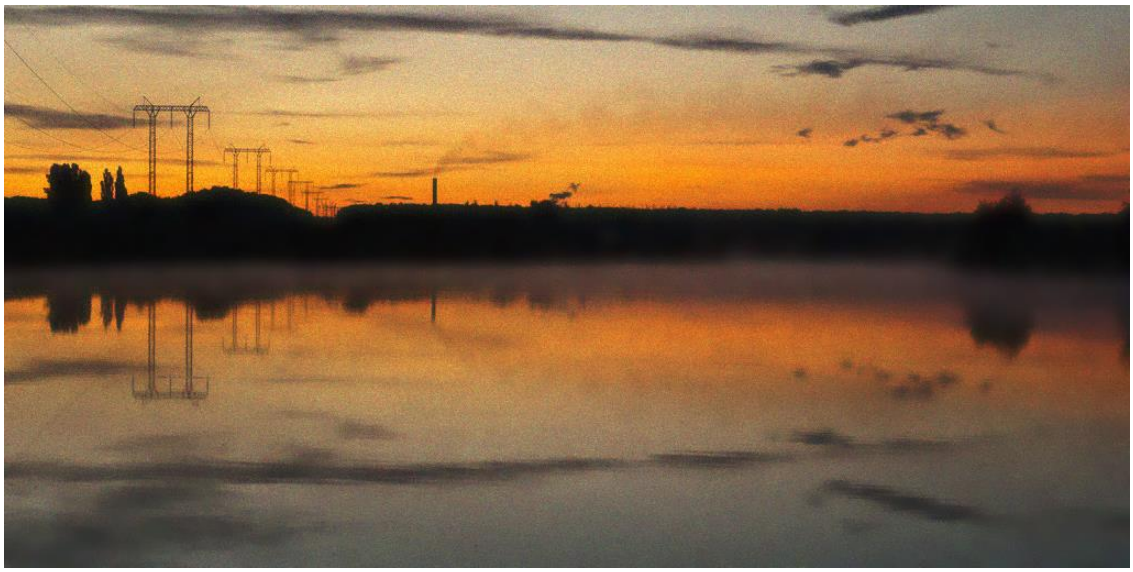
Наступний алгоритм – це двосторонній алгоритм – це фільтр котрий згладжує та зберігає краї та зменшує кількість шуму на зображенні. Приклад зображення можна знайти на рисунку 23.



«Рисунок 23 – Зображення з використанням двостороннього алгоритму»

Як можна побачити з рисунку 23, шуми були повністю видалено, але втрачено значну кількість корисної інформації, втрачено деталізацію та отримали дуже розмите зображення. Цей алгоритм краще застосовувати до чорно-білих цифрових зображень.

Алгоритм Вінера досягає зниження шуму з певною втратою цілісності деталізації зображення. Але якщо балансувати між зниженням шуму та втратою деталізації можна досягнути непоганих результатів. Результати можна подивитись на рисунку 24.



«Рисунок 24 – Зображення з використанням алгоритму Вінера»

Було перевірено різні режими використання цього алгоритму, найгарніший результат було отримано у режимі середньої фільтрації шуму, де ми отримали найбільш деталізовано зображення, з якомога меншим розмиттям перепадів яскравості. Найбільш походить до оригінального зображення.

Для більш детального порівняння було зроблено колаж, зліва на право: оригінальне зображення, зображення з додаванням білого шуму, використання Median, Kuwahara, Двостороннього та Вінера алгоритмів.

Колаж з усіма алгоритмами можна подивитись на рисунку 25. Можна зробити висновок, що найбільш схожим с оригінальним зображенням є зображення з використанням алгоритмів Median та Вінера. Де алгоритм Median дав гарну деталізацію, а алгоритм Вінера майже повністю видалив білий шум з зашумленного зображення.

«Рисунок 25 – Колаж з використанням усіх алгоритмів»

3.3 Порівняння алгоритмів на імпульсному шуму

Для надання більш точної оцінки й коректної рекомендації потрібно порівняти оглянуті алгоритми використовуючи їх на імпульсному шумі. Для візуалізації їх роботи було вирішено обрати інше зображення. Приклад котрого, можна знайти на рисунку 26. Зображення було зроблено на фотокамеру Nikon D80,

об’єктив Nikkor18-135 за поганих погодних умов, недостатньої кількості світла та великому рівні вологості повітря.

«Рисунок 26 – Зображення без додавання імпульсного шуму»

Завдяки фото редактору Photoshop було надано імпульсний шум на оригінальне зображення. Для демонстрації роботи алгоритмів будемо використовувати їх по черзі до зашумленного зображення, як робили для білого шуму. Приклад зображення з додаванням імпульсного шуму можна подивитись на рисунку 27.

«Рисунок 27 – Зображення з додаванням імпульсного шуму»

Першим фільтром будемо використовувати фільтр $Median$. Приклад на рисунку 28. Цей алгоритм дав непоганий результат на білому шумі та з більш насиченим кольорами зображенні, але з великими розмиттям переходів.

«Рисунок 28 – Зображення після роботи алгоритму $Median$»

Як можна побачити з рисунку 28, зображення не отримало бажану деталізацію, на більш нейтральному до кольорів зображенні також було погано видалено імпульсний шум на цифровому зображенні та спотворено розмиттям.

Наступним алгоритмом буде алгоритм Kuwahara, використаємо його на зашумленому зображенні з рисунку 27, да подивимся результат його роботи на нейтральному до кольорів зображенні. Візуальний приклад зображено на рисунку 29.

«Рисунок 29 – Зображення з додаванням алгоритму Kuwahara»

Алгоритм Kuwahara дав гарний результат на даному типі зображення, деталізація переходів яскравості, глибина чіткості залишилися майже на рівні оригінального зображення, кількість шумів знизилась до приємної для ока якості. Можна зробити висновок, що цей алгоритм краще підходить для видалення імпульсного шуму з менш насиченого кольором зображенні.

Наступний алгоритм котрий ми будемо використовувати – це двосторонній. На кольоровому зображенні й з білим гаусівським шумом цей алгоритм показав гарний результат видалення шуму, але це було супроводжене розмиттям. Візуальний приклад можна побачити на рисунку 30.

«Рисунок 30 – Зображення з додаванням двостороннього алгоритму»

Двосторонній алгоритм на менш насиченому кольорами зображенні й з імпульсним шумом дав бездоганний результат, котрий відсотків на 95 схожий з оригінальним зображенням без додавання шуму. Але на зображенні з білим гаусівським шумом дало велике розмиття на всій поверхні цифрового зображення.

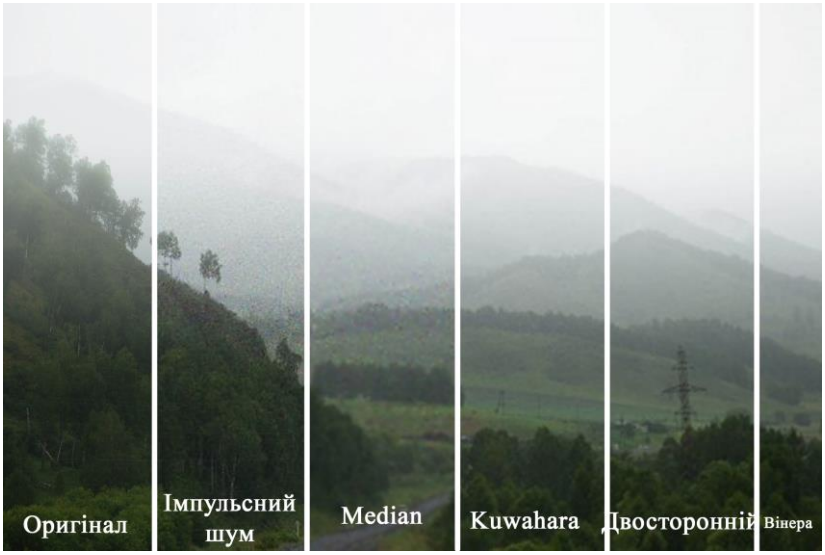
Останній алгоритм, котрий ми розглянемо на даному типі фотографії з імпульсним шумом – це алгоритм Вінера. Це лінійний алгоритм для адаптивної локальної обробки цифрового зображення. Цей підхід найчастіше більше ефективний, ніж звичайна лінійна фільтрація. Переваги цього алгоритму у тому, що він зберігає краї й інші високочастотні частини об’єктів зображення. Однак цей алгоритм потребує порівняно більше часу на обробку зображення. Візуальний приклад роботи алгоритму Вінера зображено на рисунку 31.

У результаті бачимо гарний результат, наближений до двостороннього алгоритму та оригінального зображення.

Перепади світла, контрастність, деталізація на високому рівні. Більш-менш коректне розмиття не перешкоджає перегляду зображення.

«Рисунок 31 – Зображення з додаванням алгоритму Вінера»

Колаж з усіма алгоритмами можна подивитись на рисунку 32.



«Рисунок 32 – Колаж з алгоритмами»

Найбільш схожим с оригінальним зображенням є зображення з використанням двостороннього алгоритму. Він дав гарну деталізацію та бездоганно видалив імпульсний шум.

Завдяки використанню бібліотеки EmguCV, ми можемо візуально оцінити якість роботи, а також отримати кількість часу роботи кожного з алгоритмів. Реалізований медіанний алгоритм на кольоровому зображенні з білим гаусівским шумом впорався з задачею бездоганно, витратив 0.111701 секунди. Що є гарним результатом, якщо нам потрібно оброблювати зображення в умовах потоку, наприклад відео-потік, тобто мінімальна прийнятна кількість кадрів за секунду для людського ока є 30 кадрів за секунду, далі «fps». Тобто у разі потреби обробки відео потоку цей алгоритм дасть приблизно 500-530 fps, що є гарним показником. У свою чергу алгоритм Кувахара обробляє зображення за 3.0089847 секунди. У разі відео потоку буде дорівнювати трохи менш ніж 20 fps, що не допускає можливість використовувати цей алгоритм до відео потоку.

Отримані візуальні й часові результати слід використати для надання рекомендації. Отже, якщо є потреба обробляти насичене кольорами зображення з білим шумом якомога швидко слід використовувати медіанний алгоритм. Якщо маємо діло з фільтрацією шуму з менш насиченим кольорами зображенням з імпульсним шумом і слід також якомога швидше відфільтрувати зображення – то потрібно використовувати двосторонній.

ВИСНОВКИ

Більшість маніпулювань з цифровим зображенням робиться не тільки з метою зменшення різних видів шуму, але першим завданням у будь-якому процесі над зображенням повинно починатися з зменшення шуму, якщо це потрібно. На основі значення параметра метод застосовує серединний алгоритм до буфера вихідного зображення. Під час ітерації сусідства пікселів проводиться порівняння між поточно ітераційним значенням сусіднього пікселя та попередньо визначеними мінімальними та максимальними значеннями. Наприклад, розглянемо необхідність пошуку обличь на цифровому зображенні [6]. Для пошуку обличь на зображенні використовується багато різних методів, але кожен з них використовує патерни, такі як очі, овал обличчя, рот чи ніс. У разі, коли в нас датасет має у собі дуже зашумлені зображення, що може негативно вплинути на пошук обличчя та надати недостовірно позитивні або недостовірно негативні результати, то перед кожним скануванням зображенням нам потрібно спочатку зробити попередню обробку, таку як зменшення шуму, підвищення деталізації, підвищення контрастності – це робиться для отримання більшої кількості інформації з цифрового зображення, що у свою чергу дасть більш точний результат.

Під час праці було досліджено природу появи шуму на цифровому зображенні. Появі шуму сприяє погана матриця фотоапарату, високе значення ISO, погані погодні умови, нічна зйомка, перегрів матриці, перешкоди при передачі сигналу, недосконалі сканери, чи у результаті роботи інших фільтрів. Але не завжди шуми це погано, іноді у художній фотографії використовують шум, як елемент художнього прийому, наприклад на портретному зображенні, щоб надати цифровому зображення ефект старої фотографії чи для акцентування погляду на обличчі.

У результаті праці за темою «Дослідження моделей й алгоритмів фільтрації шуму на цифровому зображенні» було виявлено найкращі умови для кожного алгоритму. Так як немає універсального алгоритму для кожного типу зображень, чорно-біла фотографія чи кольорова, стара кропнута матриця чи повно-кадрова, пейзаж чи портрет, висока роздільна здатність чи фото після стиснення.

Для більш правильного тестування алгоритмів було обрано цифрову фотографію належної якості, щоб було з чим порівняти результати роботи алгоритмів. До оригінальної фотографії було додано у першому порівнянні гаусівський білий шум, а у другому імпульсний. На кожному етапі до зашумленного імпульсним або білим шумом було додано фільтри: Median, Kuwahara, двосторонній та Вінера. У результаті ми отримали порівняння у вигляді двох колажів з шести секцій, оригінальне зображення, зображення з додаванням шуму, та усі розглянуті алгоритми.

Для тесту над насиченим кольором зображенні з білим шумом, з великими контрастними дільницями й безліччю маленьких деталей було обрано пейзаж на світанку. Додано завдяки фото редактору Photoshop білий шум, й накладено фільтри. Найбільш якісно відпрацював алгоритм Median, він надав гарну деталізацію, мінімальне розмиття переходів та наближено до оригінального зображення без білого шуму. На другому місці алгоритм Вінера, дав непоганий результат, але є деяка акварельність до зображення та розмиття. Двосторонній алгоритм та алгоритм Kuwahara спотворив зображення, перший не видалив шум, а другий дав занадто розмите цифрове зображення відповідно.

Для другого тесту було використано менш насичене кольорами зображення, з менш контрастними дільницями, але теж з великою кількістю деталей. Завдяки фото редактору Photoshop було додано імпульсний шум до оригінального зображення й по черзі до оригінального зображення було накладено фільтри. Двосторонній алгоритм на цьому типі фотографії з імпульсним шумом дав бездоганний результат, котрий наближений до оригінального цифрового зображення без додавання шуму. Але у свою чергу цей алгоритм на зображенні з

білим шумом дало велике розмиття на всій поверхні зображення. Наступний алгоритм котрий добре підходить до менш кольорового зображення з імпульсним шумом – це алгоритм Вінера. Переваги алгоритм Вінера – це те, що він зберігає краї й інші високочастотні частини об’єктів зображення. Однак з недоліків цього алгоритм є більший час на обробку зображення, що є критичним у разі потреби обробки потокового відео у реальному часі. Але це ідеально підходить до роботи з цифровим зображенням. Алгоритми Median та Kuwahara спотворили зображення, перший не якісно зменшив імпульсний шум, а другий дав досить велике розмиття, що дає втрату корисної інформації.

Слід зауважити, що для конкретної ситуації, імпульсний шум чи білий, високо насичені кольором зображення чи чорно-білі, зображення з текстом, або портрет з навмисним додаванням шуму – для кожної з таких ситуацій потрібно обирати коректний алгоритм. Для подальшої роботи з цифровим зображенням потрібно робити попередню обробку на видалення чи зменшення шуму. Також що до рекомендації, доброю практикою є використання комбінації з двох або більше алгоритм, з неагресивною дією кожного. Приклад цього використання можна знайти у пресетах до Adobe Lightroom, де можна створити набір фільтрів й зберегти їх до пресету. Бездоганна робота одного алгоритм у одній ситуації, не обов’язково дасть бездоганний результат у другій. Як бачимо з дослідження, двосторонній алгоритм якісно відпрацював на менш насиченому кольорами зображенні, але у той же час спотворим зображення на насиченому кольорами. Так и навпаки, на кольоровому цифровому зображенні, Median дав гарний результат, але у менш кольоровому спотворив його. Тому потрібно досить акуратно підходити до вибору фільтру, виходячи з потреби, ситуації й результату.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Arsenov A. Evolution of Convolutional Neural Network Architecture in Image Classification Problems / A. Arsenov, I. Ruban, K. Smelyakov, A. Chupryna // Selected Papers of the XVIII International Scientific and Practical Conference on Information Technologies and Security (ITS 2018). – CEUR Workshop Processing. – Kyiv, Ukraine, November 27, 2018. – Pp. 35-45. R. Gonzalez, R. Woods Digital Image Processing. Second Edition. – Prentice Hall, 2002. – 793p.

2. Б. Яне Цифровая обработка изображений. – М.: Техносфера, 2007. – 584 с.

3. M. Sonka, V. Hlavak, R. Boyle Image processing, analysis, and machine vision. – California (USA): Cole Publishing Company, 1999. – 770p.

4. Smelyakov K., Smelyakov S., Chupryna A. Advances in Spatio-Temporal Segmentation of Visual Data. Chapter 1. Adaptive Edge Detection Models and Algorithms. Series Studies in Computational Intelligence (SCI), Vol. 876. – Publisher Springer, Cham, 2020. – P. 1-51.

5. Л. Шапиро, Дж. Стокман Компьютерное зрение; пер. с англ. – М.: БИНОМ, 2006. – 752 с.

6. OpenCV [Електронний ресурс] // opencv.org. – 2006. – Режим доступу до ресурсу: https://opencv.org/.

7. Smelyakov K., Datsenko A., Skrypka V., Akhundov A., Chupryna A. Efficiency of Image Reduction Algorithms with Small-Sized and Linear Details // 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 8-11 Oct. 2019, Kyiv, Ukraine. – P. 745-750.

8. Smelyakov K., Chupryna A., Hvozdiev M., Sandrkin D. Gradational Correction Models Efficiency Analysis of Low-Light Digital Image // 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), 25-25 April 2019, Vilnius, Lithuania. – P. 34-39.Гонасален Р. Цифровая обработка изображений / Р. Гонасален, Р. Вудс., 2002. – 1104 с. – (3-е).

9. Цифрове патентне джерело [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: http://iii.ua/uk/cifrova-patentna-biblioteka.

10. Два метода адаптивной медианной фильтрации импульсного шума на изображении [Електронний ресурс] // 2018 – Режим доступу до ресурсу: http://www.mathnet.ru/links/1ecadcb69503355fc6fb6865784d5f94/co548.pdf.

11. IMAGE DENOISING USING NEW ADAPTIVE BASED MEDIAN FILTER [Електронний ресурс] // Signal & Image Processing : An International Journal. – 2014. – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1410/1410.2175.pdf.