МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система виявлення клонування на цифровому зображенні»

Завідувач
випускаючої кафедри **Довбиш А.С.**
Керівник роботи **Шелехов І.В.**
Студент гр. КБ–61 **Панченко А.А.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую _____

Зав. кафедрою Довбиш А.С.

“_____” _____2020 р.

Завдання

до випускної роботи

Студента четвертого курсу, групи КБ-61 спеціальності “Кібербезпека” денної форми навчання Панченка Аліни Анатолівні.

Тема: “**Інформаційна система виявлення клонування на цифровому зображенні**”

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд сучасних методів виявлення фальсифікації цифрових зображень;2) постановка задачі дослідження; 3) дослідження алгоритмів блокового співставлення, та автокореляційного алгоритму; 3) розробка інформаційного та програмного забезпечення інтелектуальної системи виявлення клонування на цифровому зображенні.

Дата видачі завдання “_____”_____2020 г.

Керівник випускної роботи _____Шелехов І.В.

Завдання прийняв до виконання _____ Панченко А.А.

РЕФЕРАТ

Записка: 48 стор., 29 рис., 1 таблиця, 1 додаток, 12 джерел.

Об’єкт дослідження — процес виявлення клонування на цифровому зображенні.

Мета роботи —порівняння та розробка алгоритмів виявлення клонування на цифровому зображені

Результати — розроблено алгоритм та програмне забезпечення системи виявлення клонування на цифровому зображенні. При цьому виконано модифікацію базових блочних алгоритмів. Програмна реалізація виконана в середовищі для наукових та інженерних розрахунків MATLAB

ФАЛЬСИФІКАЦІЯ ЦИФРОВИХ ЗОБРАЖЕНЬ, КЛОНУВАННЯ, БЛОЧНЕ СПІВСТАВЛЕННЯ, АВТОКОРЕЛЯЦІЙНИЙ АЛГОРИТМ

ЗМІСТ

ВСТУП

Протягом останнього десятиліття невідповідні маніпуляції із зображенням стали серйозним занепокоєнням у різних галузях суспільства, таких як новини, у політиці чи у розважальному секторі. Програми цифрового редагування зображень в наш час дуже потужні та постійно розвиваються. В наш час цілісність зображення відіграють дуже важливу роль і через багато випадків шахрайства маніпулювання зображеннями привертає все більше уваги, для пошуку та розробки методів розпізнавання маніпуляцій.

Метою даної роботи є дослідження розвитку й актуального положення фальсифікації цифрових зображень.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Підробка цифрових зображень

У сучасному світі легко маніпулювати зображенням, додаючи або видаляючи з нього деякі елементи, що призводить до великої кількості підроблених зображень. Цифрові зображення пропонують безліч атрибутів, таких як яскравість або колір окремих пікселів як для маніпуляцій, так і для алгоритмів виявлення несанкціонованих дій.

Зображення, в основному являється загальноприйнятий доказ виникнення будь-якої зображеної події. Через наявність недорогих програм легко маніпулювати такими зображеннями. В результаті ми втратили автентичність, та достовірність зображення. Зміна або модифікація зображення характеризується як редагування, тобто "додавання або стирання" деяких життєвих моментів із зображення, не залишаючи слідів втручання.

Існує два типи методів фальсифікації зображень: активні та пасивні підходи [1]. Ці типи мають свої специфічні типи, які показані на рисунку 1.1. Зважаючи на методи, що застосовуються для зміни зображень, існує три типи пасивної підробки цифрових зображень: сплайсинг зображення або композитні зображення, підробка та ретуш зображення, копіювання-переміщення або дублювання регіонів

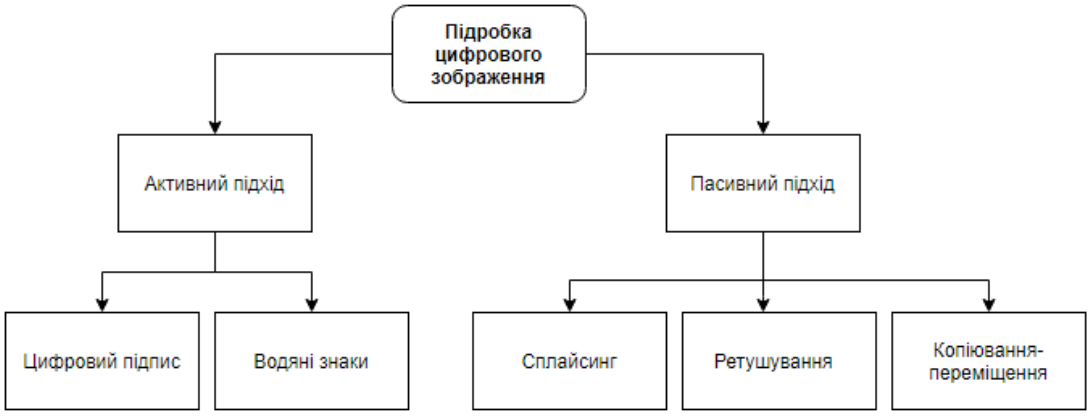

Рисунок 1.1 – Види підробки цифрових зображень

Сплайсинг зображення - це часто застосовувана, досить проста техніка підробки, яка обробляє та компонує зображення з різних джерел. Операція сплайсінгу спричинена витісненням принаймні однієї частини зображення ділянкою іншого зображеня. Існують численні інструменти, доступні для зміни зображень, такі як поліпшення, морфінг тощо. Сплайсинг - це тип фотографічної маніпуляції, який включає комп'ютеризоване сплайсування щонайменше двох знімків в одне складене зображення, яке може не мати подальшої обробки, наприклад, згладжування меж між різними фрагментами. На рисунку 1.1 - приклад сплайсингу зображення. Ця техніка внесення змін може спричинити нерівності в численних ознаках, як, наприклад, надзвичайно різкий перехідний час на краях, і ці нерівності використовуються для виявлення фальшивості. Сплайсування зображень використовується у фотомонтажі з метою об'єднання двох зображень, це одна з найпоширеніших практик підробки цифрових зображень [5].

Рисунок 1.2 – Приклад сплайсування зображення

Інша часто вживання техніка маніпуляції ретушування. Підпадає під цю класифікацію підпадає будь-яке налаштування, або зміна зображення за допомогою програмного забезпечення (графічних редакторів) для досягнення певного результату, наприклад, для висміювання або покращення зображень. Ця процедура кардинально не змінює картину, а скоріше покращує або зменшує конкретний елемент картини [2]. Наприклад, щоб покращити

зображення, деякі обрані його частини можуть зазнати геометричних змін, таких як обертання, масштабування, розширення тощо. Ретуш приносить чіткі переривчасті зв’язки у малюнок. Ці з'єднання можуть бути використані для сприйняття підробки, яка здійснюється ретушшю. Незалежно від того, яка камера використовується для фотографування, можна змінити кожну фотографію, щоб потім усунути будь-які дефекти. Ретуш передбачає безліч процедур, таких як необхідне коригування затінення, зміна шкіри та відновлення фотографій і так далі. Приклад ретушування, задля покращення якості фотографії показано малюнку 5.

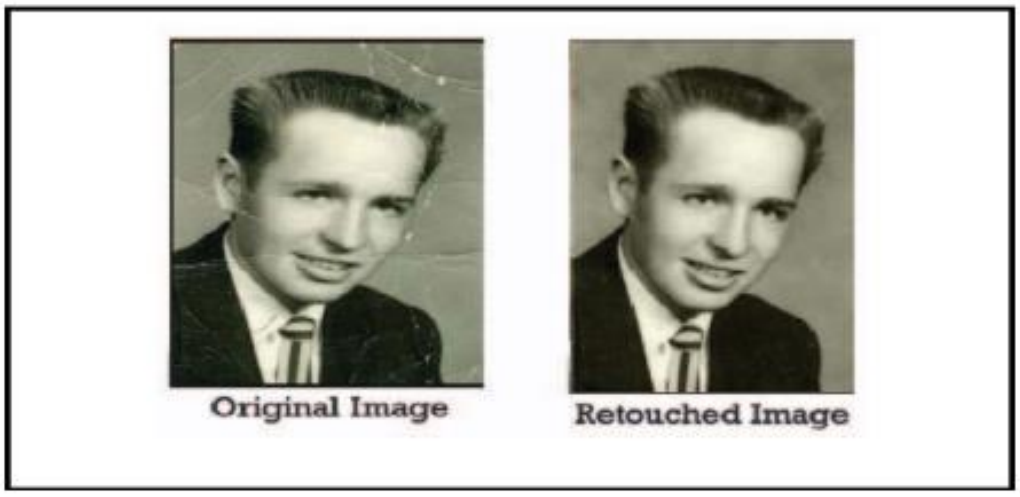

Рисунок 1.3 – Приклад ретушування зображення

Техніка копіювання-переміщення - одна з складних в реалізації типів маніпуляції зображення. При атаці копіювання-переміщення частини оригінального зображення копіюються, переміщуються до потрібного місця та вставляються. Зазвичай це робиться для того, щоб приховати певні деталі або клонувати певні аспекти зображення. Оскільки скопійована частина походить з одного і того ж зображення, його важливі властивості, такі як шум, палітра кольорів та текстура, будуть сумісні з рештою зображення, і тому складніше буде відрізнити та виявити ці частини.

Клонування регіонів - це найпоширеніший метод зміни зображень, що застосовується через простоту та ефективність, коли частину зображення будь-якої форми та розміру в певній області переставляють (копіюють та вставляють) в інший частину у тому ж зображенні, як показано на малюнку 3. [4]. Але також є звичайний випадок, коли патч фонового зображення копіюється та вставляється, щоб приховати об'єкти, присутні на зображенні.

Рисунок 1.4 – Приклад маніпуляції типу копіювання-переміщення

Основне поняття атак клонування може бути узагальнене наступним чином: є два типи об'єктів, що беруть участь, джерело, тобто об'єкт, який було скопійовано, і ціль, тобто клонований об'єкт. У простому випадку цільовий об’єкт просто вставляється у нове положення зображення, не піддавшись обробці (перетворення перекладу). Але також є випадки, які стосуються обробки клонованого об'єкта, наприклад, розмивання, обертання та масштабування. Оскільки репліковано частина починається з тієї ж картини, її основні властивості, наприклад насиченість, колір і зерна не змінюються і ускладнюють процес розпізнавання.

Крім того, існує більш складна трансформація, яка виконується на об'єктах, наприклад оклюзія, де цільовий об'єкт вставляється таким чином,

щоб він мав кілька пікселів, що перекриваються з іншими об'єктами зображення, і в результаті зображення виглядає більше реалістичні.

Логіку, яка стоїть за використанням клонування, можна виявити, знаючи про причину, чому комусь потрібно клонувати ділянки зображення. Наприклад, у 2013 році Північну Корею звинуватили у маніпулюванні їхніми військовими зображеннями. Зокрема, перед тим, як випустити зображення до преси, вони додали кілька клонованих повітряних кораблів, щоб показати світові, що в тренувальних випробуваннях цього дня було більше транспортних засобів [3]. Це може здатися не такою головною великою проблемою в маніпулюваннях цими фотографіями, але це може бути використано як стратегія розповсюдження пропаганди.

Для захисту авторських прав та запобігання підробці необхідна надійна техніка виявлення підробок зображень у багатьох сферах. Застосування, в яких використовується техніка виявлення підробки зображень, - це публіцистика, наукові публікації, криміналістика тощо. Таким чином, техніка виявлення підробок зображень стає важливою і новою сферою дослідження. А підробка зображення все частіше стає інструментом зловмисників з метою введення в оману сприйняття спостерігачів [2].

1.2 Методи виявлення фальсифікації цифрових зображень

Виявлення зображення, які були маніпулюванні, надзвичайно вдосконалились. Для виявлення пасивних цифрових підробок зображень існують декілька найпопулярніших методів, які групуються у п'ять категорій. Ці методи не використовують попередньо вбудованих даних усередині зображення які були добавленні при їх створенні, для запобігання фальсифікації (цифрові підписи, водяні знаки). Ці методи працюють шляхом аналізу двійкових даних зображень [5]. На рисунку 1.5 показано класифікацію існуючих методів виявлення підробки цифрових зображень.

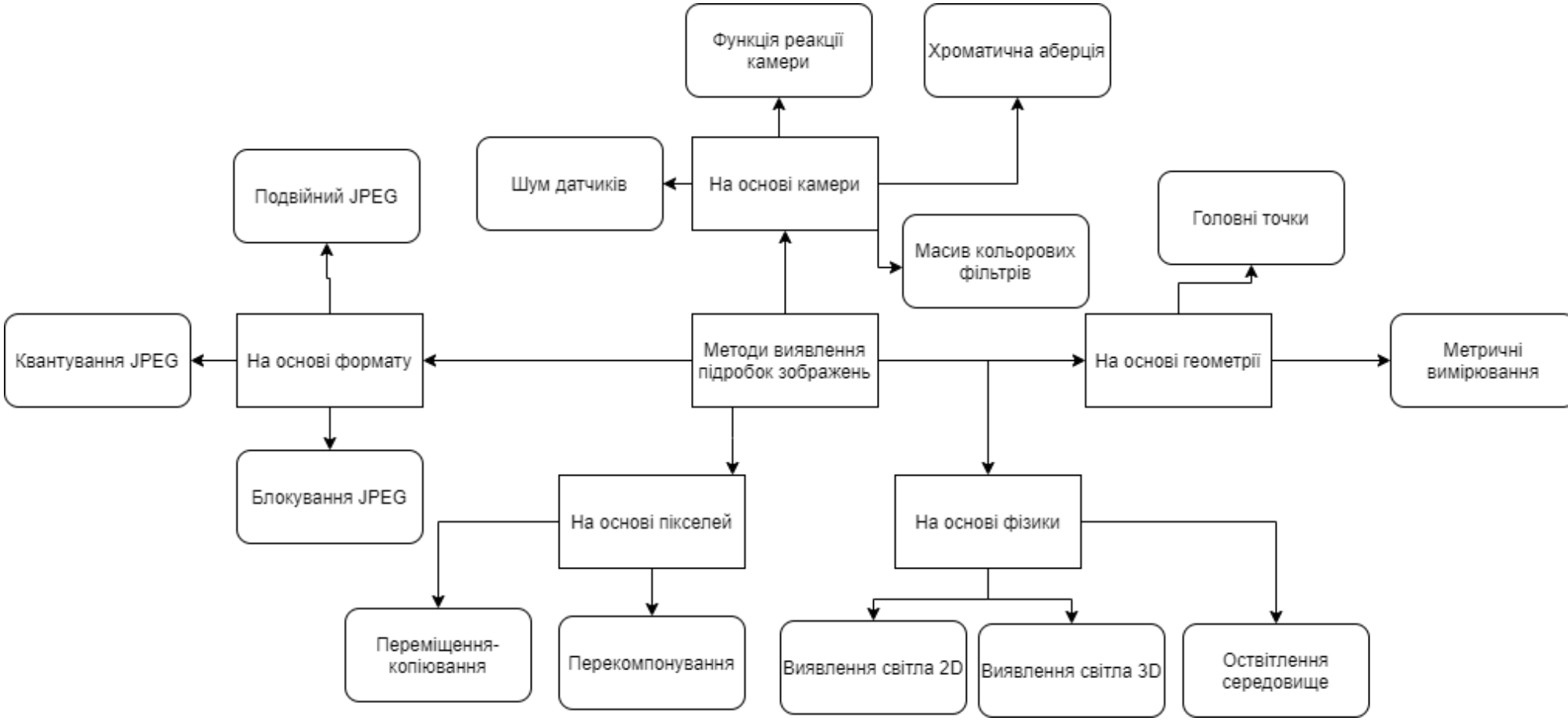

Рисунок 1.5 – Класифікація методів виявлення підроблення зображень

Виявлення підробки цифрового зображення на основі формату. Цей метод використовує особливості формату зображення. Найбільш популярний формат, на якому цей метод використовується це формат JPEG. Реалізація в JPEG, дозволяє використовувати різні методі для виявлення змін у дизайні формата. Маніпуляція із зображеннями спричиняє модифікацію сітки артефактів блоків, особливо у випадку маніпуляцій типу копія-переміщення. Квантування JPEG, блокування JPEG та подвійний JPEG - це три основні критерія, за якими можна розпізнати зображення фальшивий також для стислих картинок[4].

Виявлення підробки цифрових зображень на основі пікселів. Цей метод використовує пікселі цифрового зображення, які є основними структурними блоками. Ці стратегії знімають різні фактичні відхилення, які вводяться на рівні пікселів [5]. Найпоширеніші методи виявлення в цій категорії - це переміщення-копіювання та перекомпонування.

Виявлення підробки цифрових зображень на основі камери. Коли ми знімаємо фотографію з цифрової камери, зображення переміщується з датчика камери в пам'ять, і воно проходить декілька етапів обробки, включаючи квантування, з'єднання тіней, поправку гами, фільтрацію, балансування білого та стиснення JPEG [6]. Ці етапи змінюватися в залежності від моделі камери. Чотири основні методи, що працюють при виявленні підробки цифрових зображень на основі камери, - це шум датчика, масив кольорових фільтрів, хроматична аберація та функція реакція камери [5]

Виявлення підробки цифрових зображень на основі фізичного середовища. Особливості тривимірної асоціації між камерою, світлом та фізичними виробами можуть використовуватися в стратегіях виявлення підробки зображень. Наприклад, використовуючи методи сплайсингу, складно формувати точну відповідності в освітлювальних елементах, як у оригінальної фотографії. Тут відмінності у фоновому освітленні можуть бути використані як доказ зміни. Функціонування алгоритмів відбувається на основі розрізнення впливу

освітлення на різні фізичні об’єкти. 2D-виявлення світла, 3D-виявлення світла та - це три основні категорії цього методу [8].

Виявлення підробки цифрових зображень на основі геометрії. Цей метод вимірює предмети та відносне положення камери. Вимірювання головних точок та різних метрик - два основні методи в техніці на основі геометрії. Головна точка - проекція камери, фокусування на площину зображення. Головна точка зображення розташована близько до її фокусної точки. При зміні об'єкта зображення відбувається відносна зміна головної точки. Ця відмінність бути використаний як доказ зміни. Отримання метричного вимірювання з одиночної картини є надзвичайно цінним у криміналістичних установках, де потрібні безперечна доказова оцінка.

В Інтернеті є багато безкоштовних інструментів, які використовують різні методики аналізу зображень, для виявлення маніпуляцій з ними. Розглянемо декілька з них.

Перший із інструментів, це "Аналіз рівня помилок" (ELA). Цей інструмент порівнює вихідне зображення з його рекомпресованою версією. Це може виявити маніпульовані регіони різними способами. Наприклад, вони можуть бути темнішими або яскравішими порівняно з аналогічними регіонами, якими не маніпулювали Інструмент призначений для виявлення тих областей зображення, які знаходяться на іншому рівні стиснення. Якщо маніпуляції були виконані на зображенні JPEG (наприклад, з елементами, що додаються чи видаляються), інструмент аналізу ELA має виявить та позначить усі маніпульовані області, оскільки відновлення зображення ставить вихідне зображення та доданий елемент на різні рівні стиснення.[5]

Для тестування було використано зображення с доданим до нього парашутиста на літаючу тарілку (Рис 1.6). В результаті аналізу ELA ми бачимо, що виділення скопійованих та вклеєних предметів чітко відрізняється від інших структур на зображенні (Рис. 1.7).

Рисунок 1.6 – Сфальсифікована фотографія

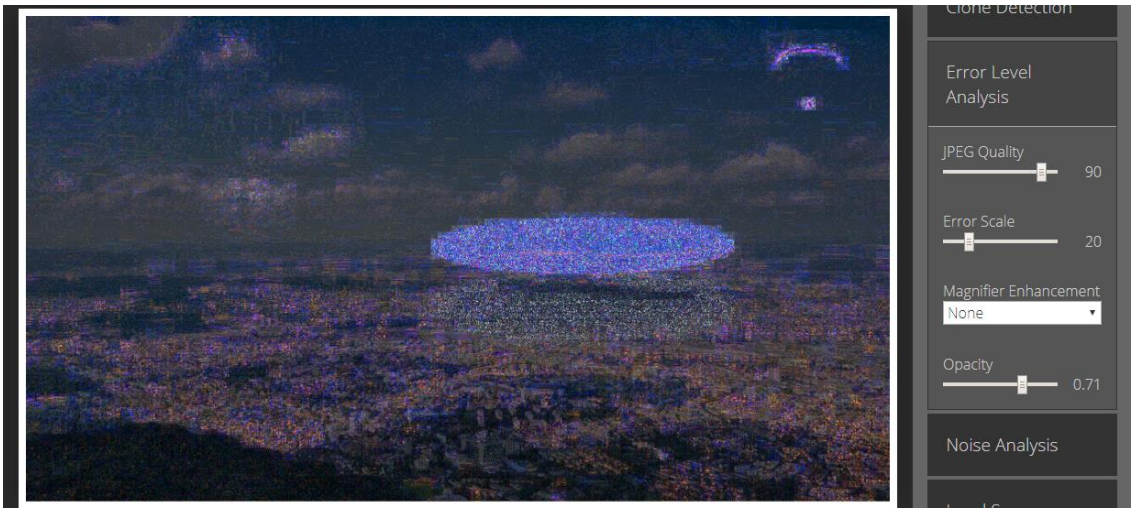

Рисунок 1.7 – Результат аналізу результаті аналізу ELA

Інший інструмент це виявлення клонів. Детектор клонів виділяє подібні ділянки зображення. Це може бути хорошим показником того, що картина була маніпульовано за допомогою інструменту клонування. .

Для перевірки було використано фотографію квітки (рис. 1.8), в якій було клонована пелюстки (рис. 1.9)

Рисунок 1.8 – Оригінал фотографії

Рисунок 1.9 – Сфальсифікована фотографія

Подібні регіони позначені синім кольором і з'єднані червоною лінією. Якщо багато регіонів перекриваються, результат може виглядати білим.

Інструмент виділяє скопійовані/клоновані пелюстки квітки і виявляє деякі інші подібності у фонових структурах (не всі вони являють собою маніпульовані ділянки) (Рис. 1.10)

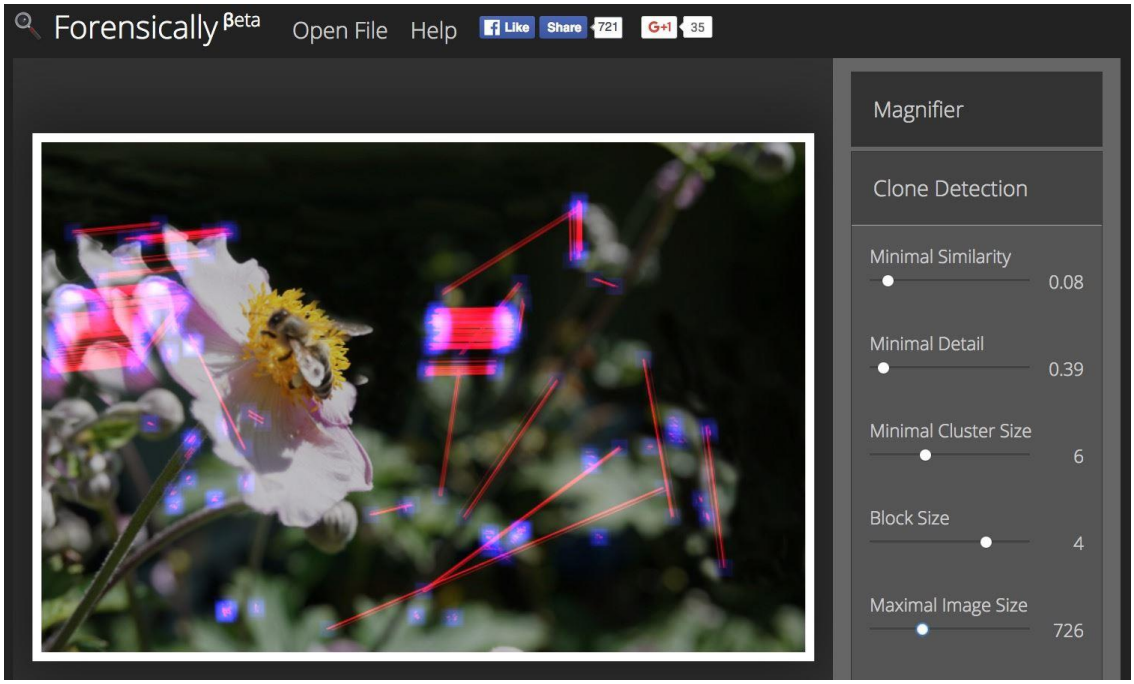

Рисунок 1.10 – Результат Clone Detection на сфальсифікована фотографії

Також було перевірено оригінальне зображення без маніпуляцій рисунок 1.11.

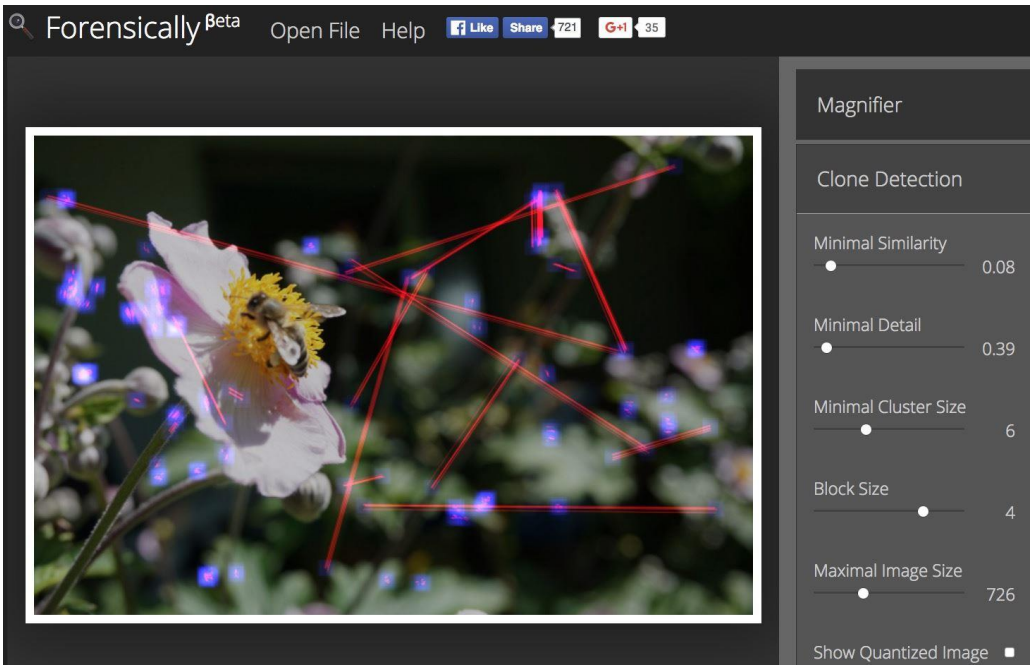

Рисунок 1.11 – Результат Clone Detection на оригіналі

Різниця між оригінальним та маніпульований зображенням залишається чітко видною. Загалом, інструмент допомагає виявити багато клонів - за винятком тих, що мають фонову текстуру.

Також одним із інструментом є ImageJ - програма для обробки зображень у відкритому доступі. Вперше була опублікована у 1997 році та розроблений для підтримки широкого спектру завдань з обробки та аналізу зображень та виявлення маніпуляцій із зображенням у багатьох різних типах зображень. Використовуватимемо операцію віднімання в Калькуляторі зображень - одному з інструментів, доступних у ImageJ. Цей інструмент ми знаходимо у випадаючому меню під заголовком "Обробити".[6]

Для перевірки було взято зображення столу зі інструментами (Рис.1.12), та його модифікацію в якому один із елементів видаляється з фотографії і перезаписується фоновою текстурою. Інший елемент копіюється та використовується для заміни видаленого елемента (Рис. 1.13).

Рисунок 1.12 – Оригінал фотографії

Калькулятор зображень показує як скопійований, так і переміщений елемент зображення, а також клоновану область та інформацію про зображення, яка фактично покрита фоном зображення (Рис. 1.14).

Рисунок 1.13 – Сфальсифікована фотографія

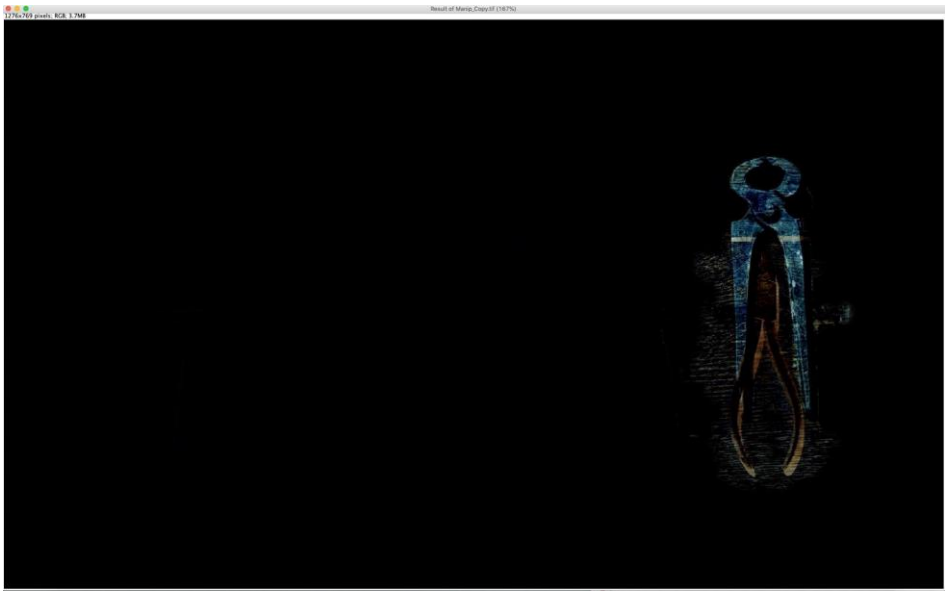

Рисунок 1.14 – Результат ImageJ на сфальсифікованій фотографії

Калькулятор зображень у ImageJ надійно визначає відмінності між не маніпульованими зображеннями та маніпульованими. Особливо для зображень із елементами, які дещо складно чітко визначити та проаналізувати неозброєним оком, цей інструмент добре підходить. Але інструмент дасть такі результати лише в тому випадку, якщо оригінальна версія модифікованого зображення доступна і може бути порівняна з модифікованим. Але це трапляється рідко в реальних сценаріях.

Найсучаснішою розробкою для виявлення клонування на зображеннях, є BusterNet, яка була опублікована для Європейської конференції з комп’ютерного бачення (ECCV). Новинка в BusterNet у галузі виявлення копіювання-переміщення в тому, що в більшості випадків передбачені маски зображення розрізняють джерело та цільові об’єкти, на відміну від інших методів, що змішують джерело-ціль, або мають лише цільові візуалізації. Архітектура BusterNet складається з двох гілок, кожна з яких відповідає за конкретне завдання. Гілка виявлення маніпуляції використовується для локалізації маніпульованих областей зображення і повертає як вихід бінарну маніпуляційну маску підробленого (цільового) об'єкта. Гілка виявлення подібності, з іншого боку, відповідає за аналіз подібності між пікселями зображення за допомогою самокореляції, щоб локалізувати подібні області та повертати як вихід вихідну бінарну маску, показуючи як вихідні, так і цільові об'єкти, але не диференційовані від кожного. Потім модуль синтезу використовується для спільного розгляду результатів двох гілок та прогнозування трикласної маски вхідного зображення, що показує диференційоване джерело (зелене), цільове (червоне) та фонове (синє) пікселів.(Рис 1.15)

BusterNet приймає в якості зображення RGB і повертає передбачувану маску клонування. Прогноз BusterNet - це трикласна маска, де синій колір представляє область, яку модель класифікує як фонові пікселі, зелений - для виявленого вихідного об’єкта, а червоний - для об'єкта, який модель класифікував як цільовий, або клонований об’єкт. Очевидно, що кольори передбачуваних масок не завжди добре виділяються. Бувають випадки, коли одна з двох гілок є більш впевненою, і в результаті маска може не завжди містити всі три кольори. Маски, на яких зображені жирні зелені або червоні зони, слід інтерпретувати як впевнене передбачення моделі. Прогноз із добре розрізненими та жирними пікселями слід додатково проаналізувати, навіть у тому випадку, якщо він не включає зелені чи червоні ділянки, оскільки остаточна маска має

вагомий вплив як з обох гілок, так і з модуля синтезу. Існує випадок, що модель класифікує всі (або більшість) пікселів як фонові пікселі. Це може бути інтерпретовано, як низька можливість того, що вхідне зображення є підробленим. Для кращої ілюстрації щодо значення передбачень BusterNet деякі результати, які були отримані, застосувавши перевірений BusterNet на маніпульованих зображеннях. Немає оригінальних версій зображень, але в більшості випадків людському оку досить легко відрізнити маніпульовані ділянки[3]. (Див. рис 1.15)

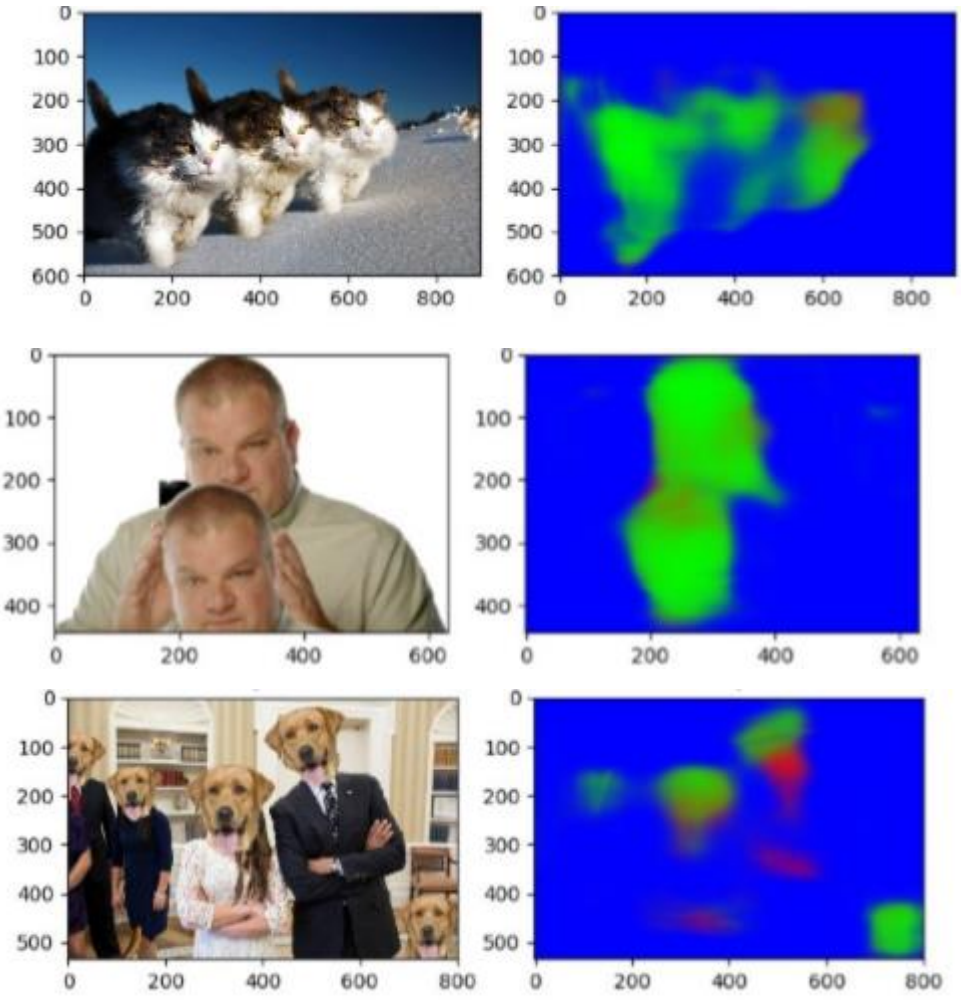

Рисунок 1.15 – Зліва оригінали, справа результати передбачення BusterNet

ManTra-Net - це досить новий продукт, запропонований для виявлення та локалізації різноманітних атак маніпулювання зображеннями. Архітектура ManTra-Net складається з двох різних підмоделей: мережа відстеження

маніпуляцій, яка відповідає за виявлення функції відстеження маніпулювання зображеннями, та локальна мережа виявлення аномалій, яка має досвід для локалізації аномалій зображень. Продукт цього дослідження представляє модель, здатну охопити більш широкий набір відомих атак маніпулювання зображеннями. Дослідники оцінюють запропонований ними метод на 385 різних відомих атаках. Мета ManTra - Net полягає у вивченні відображень для виявлення різниці між етикеткою підробки та її локальною аномалією на рівні функції. Для тренувального процесу мережі відстеження маніпуляцій дослідники використовували оригінальні патчі з пороговою (низькою) однорідністю з бази даних зображень для створення навчальних зразків з випадковими маніпуляціями. Вилучення слідів маніпуляції не є новою методикою, але до цього часу вона використовувалася для виявлення невеликої кількості різних маніпуляційних атак. Новинка ManTra-Net випливає з того, що дослідники розширили цей набір атак, уважно вивчивши вже відомі типи атак та надалі розбивши їх на основі невеликих різновидів, які могли б сформувати новий кластер атаки.

Неважко зрозуміти, що ManTra-Net - це багатофункціональний інструмент, який чудово вирішує проблеми, пов'язані з величезною кількістю сучасних методів маніпуляцій. Прогноз ManTra-Net включає бінарну маску зображення, де білі пікселі представляють виявлені маніпульовані області. Крім того, передбачена вдосконалена маска, яка показує виділені виявлені області маніпульованого зображення[3]. Приклад результатів ManTra-Net можна побачити на рисунку 1.16.

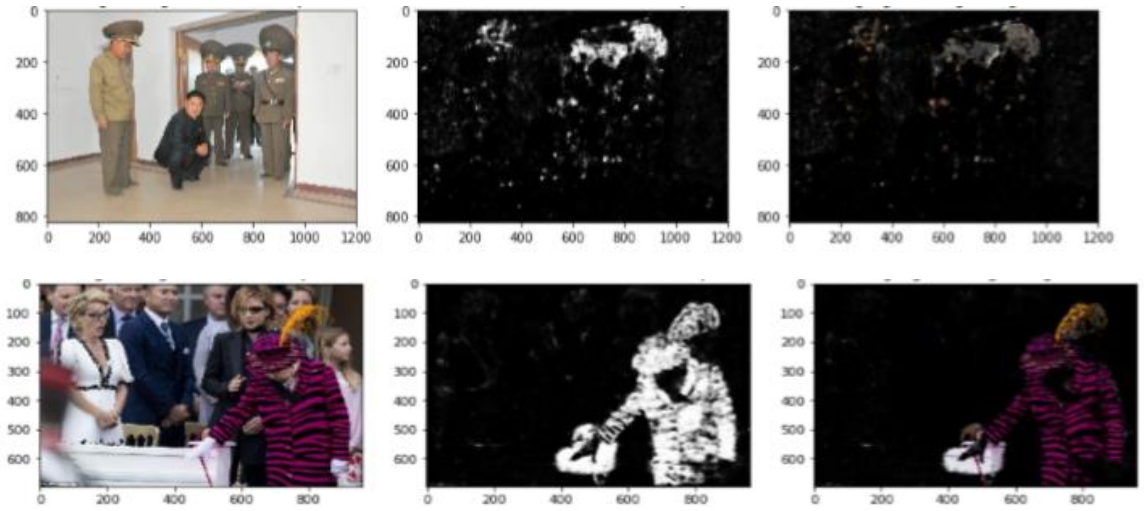

Рисунок 1.16 – Зліва оригінали, справа результат виявлення ManTra-Net

1.3 Постановка задачі

В роботі виконується інформаційний аналіз і синтез системи виявлення елементі зображення, що були клоновані. При цьому застосовується метод блокового співставлення частин зображень. Основними завданнями роботи є:

1) Виконати порівняльний алгоритмів.
2) Сформувати математичний опис системи.
3) Розробити та реалізувати програмно алгоритм блокового співставлення.
4) Розробити та реалізувати програмно алгоритм пошуку за ромбічним шаблоном.
5) Розробити та реалізувати програмно алгоритм адаптивного пошуку за шаблоном.

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 АЛГОРИТМИ БЛОКОВОГО СПІВСТАВЛЕННЯ

Ідея блокового зіставлення полягає в тому, що зображення ділиться на матрицю блоків (ці блоки під назвою «макроблоків»), які можна порівняти з сусідніми блоками, потім створюється вектор руху. Цей вектор передбачає переміщення макроблоку з одного місця в інше. Це рух, розраховане для всіх макроблоків, містять зображення, являє собою рух, що оцінюється в поточному зображенні. Область пошуку для хорошого збігу макроблоків обмежена до s пікселів на всіх чотирьох сторонах відповідного макроблоку в попередньому кадрі. Цей s називається параметром пошуку. Великі рухи вимагають більшого s, і чим більше параметр пошуку, тим вище обчислювальна вартість процесу оцінки руху [9] (Рис. 2.1).

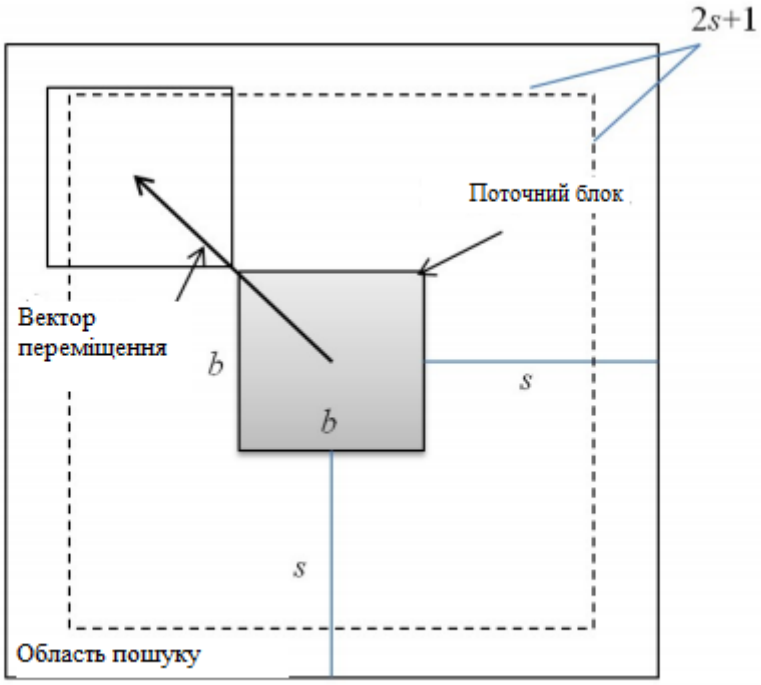

Рисунок 2.1 – Блоковий алгоритм

Розглянемо деякі основні алгоритми блокового зіставлення. Алгоритм блокового співставлення повного пошуку шукає всі можливі піксельні блоки в

діапазоні пошуку. Отже, він може генерувати найкращий вектор руху блоку. Алгоритм може дати найточніший результат, хоча необхідні обчислення непомірно високі через велику кількість точок пошуку для оцінки в області пошуку, яке є переважно високим в порівнянні з будь-яким іншим алгоритмом. Є кілька інших швидких алгоритмів, які зменшують кількість точок пошуку, але намагаються зберегти хорошу точність зіставлення блоків. Оскільки ці алгоритми перевіряють тільки обмежених кандидатів, вони можуть привести до вибору кандидата, відповідного місцевим мінімумам, на відміну від повного пошуку, що завжди призводить до глобального мінімуму.

Алгоритм трьох-етапного пошуку узагальнено полягає в наступному:

- Стадія 1. Починається з пошуку по центру і встановлює параметр пошуку s= 4. Потім виконується пошук в 8 точках з координатами +/-s навколо(0,0), а також в центрі. Вибрати точку з найменшим відхиленням.
- Стадія 2. Стадія починається з обраної точки в стадії 1 і новий параметр пошуку s = s/2=2. Повторюється стадія 1.
- Стадія 3. Для третьої стадії s = s/2 =1. Повторюється стадія 1. Точка з найменшим відхиленням на 3-й стадії знайдена. Побудувати вектор руху (Рис. 2.2).

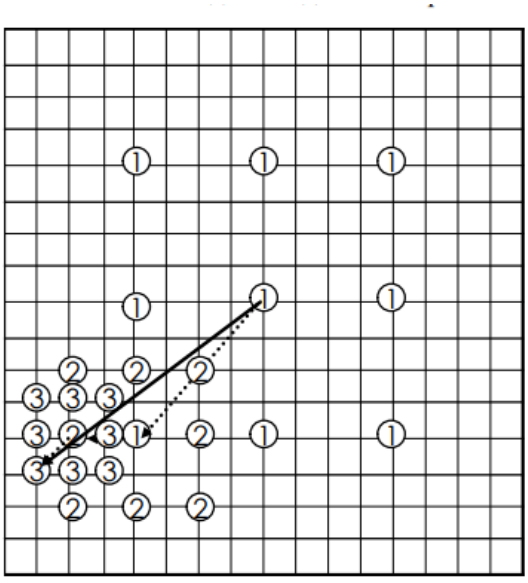

Рисунок 2.2 – Алгоритм трьох-етапного пошуку

Недоліком алгоритму є віддалена відстань між рівномірними точками шаблону, тому він неефективний для областей малого руху.

Алгоритм просто-ефективного пошуку використовує допущення про унімодальної поверхні помилки. Основна ідея алгоритму полягає в тому, що не існує 2 точки з найменшим відхиленням в протилежних напрямках на унімодальної поверхні, звідси, 8-точний пошук фіксованих шаблонів може бути змінений, щоб включити це і зберегти на обчисленнях. Алгоритм і раніше складається з трьох етапів, таких як TSS, але інновація полягає в тому, що на кожному етапі є ще дві фази. Область пошуку розділена на чотири квадранти, і перевіряються три точки A, B і C (Рис. 2.3). Правила визначення квадранта-пошуку для фази в наступному:

Якщо MAD(A) ≥ MAD(B) та MAD(A) ≥ MAD(C) , вибрати (b);

Якщо MAD(A) ≥ MAD(B) та MAD(A) ≤ MAD(C) , вибрати (c);

Якщо MAD(A) < MAD(B) та MAD(A) <MAD(C) , вибрати (d);

Якщо MAD(A) < MAD(B) та MAD(A) ≥ MAD(C) , вибрати (e);

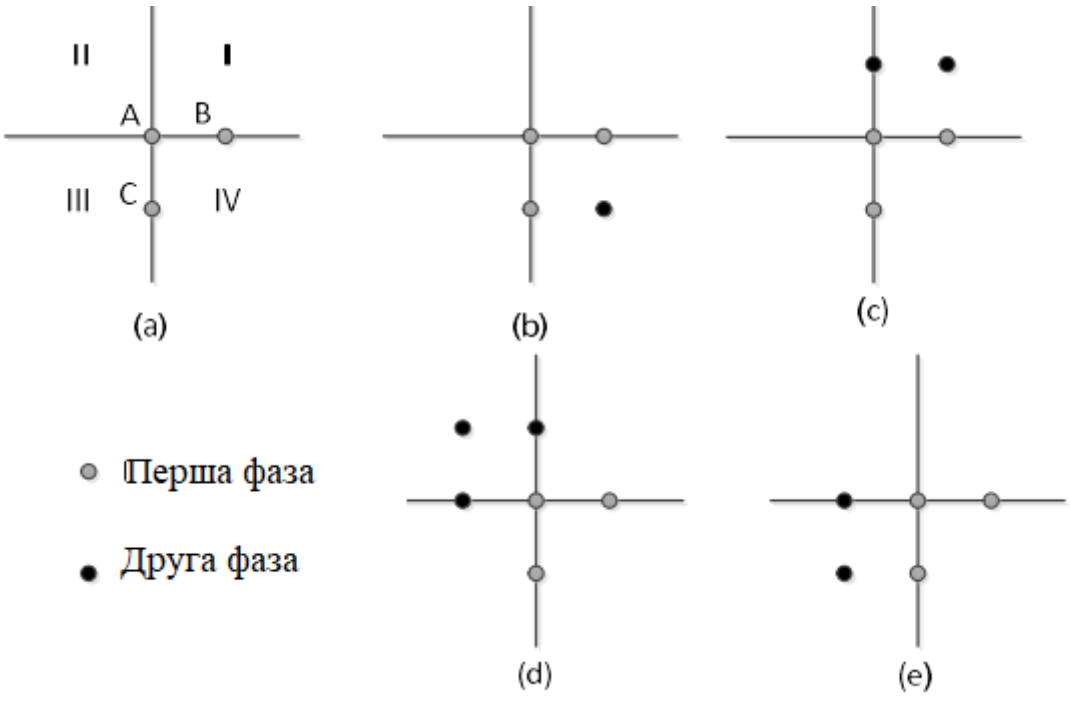

Рисунок 2.3 – Алгоритм просто-ефективного пошуку

Алгоритм чотирьохетапного пошуку аналогічний алгоритму трьохетапного пошуку. Він виконується за наступним чином (Рис 2.4):

- Стадія 1. FSS починається з пошуку по центру і встановлює параметр пошуку s = 2. Потім виконується пошук в 8 точках з координатами */- s навколо(0,0), а також в центрі. Вибрати точку з найменшим відхиленням.
- Стадія 2. Стадія починається з обраної точки в стадії 1 і новий шаблон. Повторюється стадія 1, до тих пір, оптимальна точка в центрі шаблону.
- Стадія 3. Для третьої стадії s=s/2=1Повторюється стадія 1. Точка з найменшим відхиленням на цій стадії стане остаточною. Побудувати вектор руху.

Як правило, алгоритм чотирьохетапний пошуку має велику надійність і ефективність для випадків складних рухів.

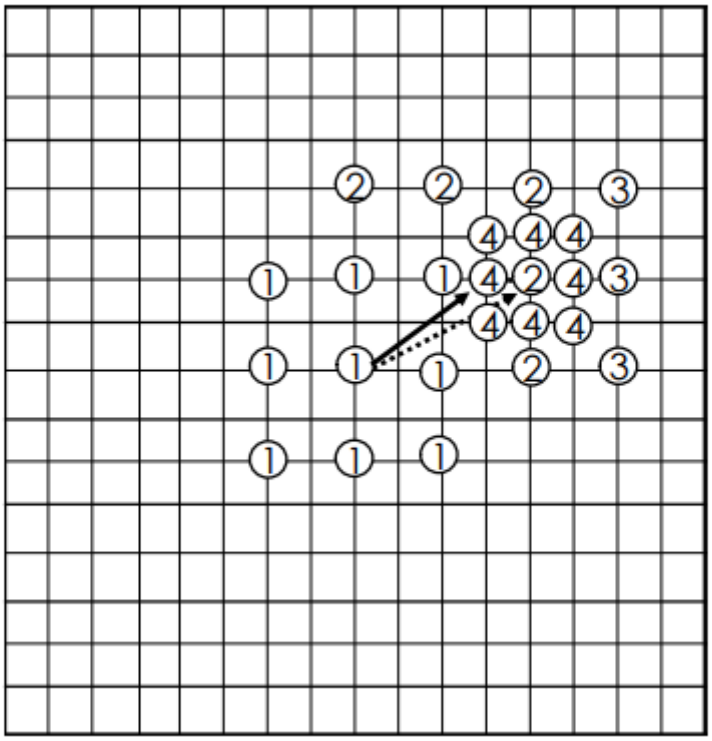

Рисунок 2.4 – Алгоритм чотирьохетапного пошуку

Пасивний Алгоритм пошуку за ромбічним шаблоном виконується виконує пошук по двома фіксованим шаблонами малий ромбічний шаблон пошуку SDSP

(англ. Small Diamond Search Pattern) і великий ромбічний шаблон пошуку LDSP (англ. Large Diamond Search Pattern) (Рис 2.5).

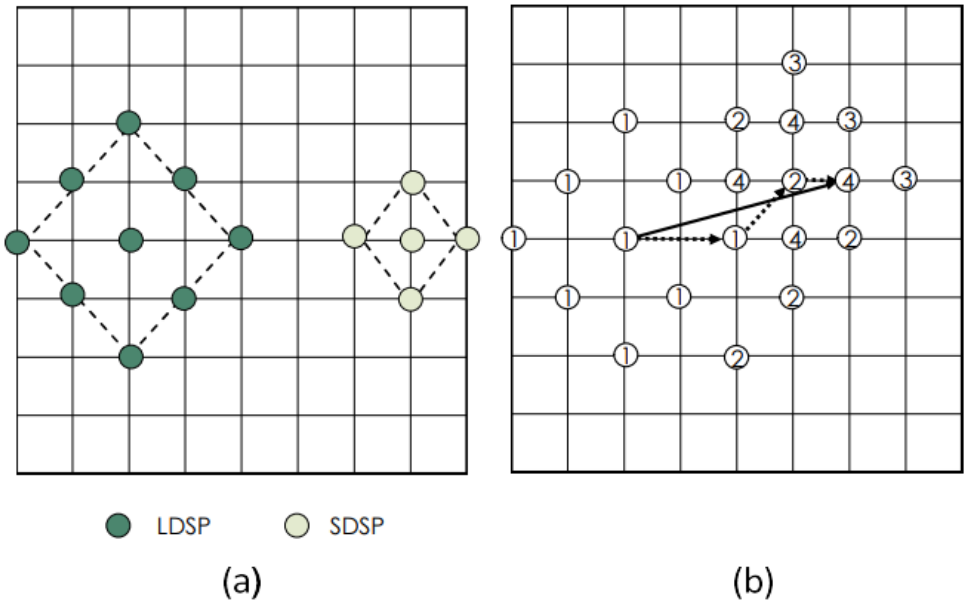

Рисунок 2.5 – Алгоритм пошуку за ромбічним шаблоном

Алгоритм виконується за таким порядком

- Стадія 1. Проведення пошуку по LDSP. Знаходити точку з мінімумом відхилення по LDSP. Якщо оптимальна точка в центрі, то перейти до стадії 2 ; в іншому випадку, побудувати інший шаблон LDSP з центром в знайденій точці і повторити стадію.
- Стадія 2. Проведення пошуку по SDSP. Знаходити точку з мінімумом

Адаптивний пошук за шаблоном описується наступним чином:

- Стадія 1. Пошук вектора-провісник руху блоку. Встановлення довжину кроку на max(|x|,|y|), де (x, y) - координати вектора-провісника руху. Знаходження точки навколо центру, що знаходяться на відстані довжини кроку від центру. Знаходження точки з найменшим відхиленням, в яку потім покласти центр.
- Стадія 2. Пошук по SDSP з центром в знайденій точці в стадії 1. Повторити пошук по SDSP до тих пір, поки точка з найменшим відхиленням не в центрі шаблону. Алгоритм адаптивного пошуку перевершує алгоритм пошуку за

ромбічним шаблоном за критерієм обчислювальної складності, так як алгоритм швидше переходить до пошуку по SDSP.

Алгоритм ієрархічного пошуку (рис. 2.6). Щоб зменшити складність алгоритмів пошуку руху, були запропоновані схеми ієрархічного пошуку від курсу до тонкого. Це скорочення обчислень пов'язано зі зменшенням розміру зображення на більш високому рівні. Прикладом цих схем є ієрархічний пошук. Алгоритм виконується за такою формулою:

$$P_i(x,y) = \left\lfloor \frac{1}{4} \left(\sum_{u=0}^{1} \sum_{v=0}^{1} P_{L-1}\left(2x+u, 2y+v\right)\right)\right\rfloor \quad (2.1)$$

де P(x,y) значення пікселя рівня L в позиції (x,y).

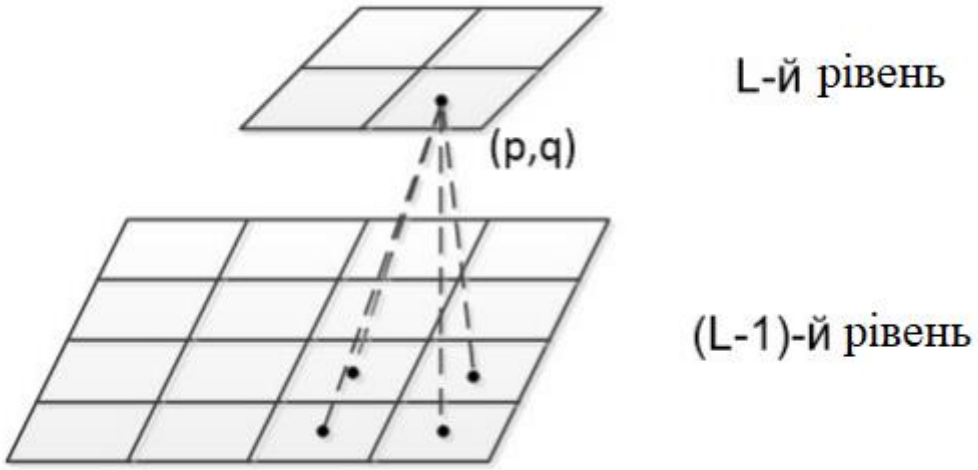

Рисунок 2.6 – Алгоритм ієрархічний пошуку

Наявні алгоритми мають свої переваги, але в загальному містять недоліки, такі як спотворення зображення, великий необхідний час виконання.

2.2 Алгоритм вичерпного пошуку

Вичерпний пошук це найпростіший найбільш очевидний підхід для виявлення клонування. У цьому методі зображення та його кругова зміщена версія перекривається, яка показано на рисунку 2.7, шукаючи зображення, в яких повністю збігається сегменти. Припустимо, що x_{ij} - значення пікселя зображення

в градаціях сірого розміру M × N у точці i, j. При вичерпному пошуку досліджуються відмінності для всі i,j за формулою[10]:

$$|x_{ij} + x_i + k\ mod(M)j + l\ mod(N)|,$$

$$k = 0,1 \dots, M - 1$$

$$l = 0,1, \dots, N - 1$$

Неважко помітити, що порівнювати x_{ij} з його циклічним зсувом [k, l] - те саме, що порівнювати x_{ij} з його циклічним зсувом [k ', l'], де k '= M – k і l'=N–l. Таким чином, достатньо оглянути лише зрушення [k, l] де 1≤ k ≤ M / 2, 1≤l≤N/2, таким чином, скорочуючи складність обчислювальної техніки в 4 рази.

Рисунок 2.7 – Приклад зображення та його зміщеної версії

Для кожного зсуву $[k, l]$ різниця $x_{ij} = |x_{ij} - x_i + k\ mod(M)j\ l\ mod(N)||$, обчислюються з малим порогом. Вибір порогу є проблематичним, оскільки в природних зображеннях велика кількість піксельних пар створюватиме відмінності нижче порогу. Але цікавлять лише пов'язані сегменти певного мінімального розміру. Таким чином, порогова різниця Δx_{ij} додатково обробляється. Зображення спочатку розмивається, а потім розширюється, до мінімального розміру області, переміщеної копією.

Хоча цей простий вичерпний підхід до пошуку ефективний, він також досить обчислювально дорогий. Обчислювальна складність вичерпного пошуку робить його непрактичним практичне використання навіть для зображень середнього розміру. Під час виявлення потрібно перевірити всі можливі

зрушення [k, l] на 1≤ k, l ≤ M / 2. За кожне зсув кожну піксельну пару необхідно порівнювати, порогову, а потім все зображення потрібно розмити та розширити. Порівняння та обробка зображення вимагають порядку операцій MN на одну зміну. Таким чином, загальні обчислювальні вимоги пропорційні $(M*N)^2$. Це робить вичерпний пошук життєздатним варіантом лише для невеликих зображень.

2.3 Автокореляційний алгоритм пошуку

Автокореляція — це деяка, яка має зміщення на певне значення, яке є незалежне, та містить саму себе. Використовують для пошуку певних закономірностей в деякому рядку даних, наприклад де існує періодичність. На практиці користуються в статистичному аналізі, або для обробки сигналів для їх подальшого аналізу[11].

Функція автокореляція зображення x розміром M × N визначається формулою :

$$r_{k,j} = \sum_{i=1}^{M}\sum_{j=1}^{N} x_{i,j}x_{i+k,j+l} \text{ , де } i,k = 0,\dots,M-1, l = 0,\dots,N-1$$

Автокореляція може бути ефективно здійснена за допомогою перетворення Фур'є, використовуючи той факт, що r=x*x^ де

$$\hat{x_{ij}} = x_{M+1-i,N+1-j} \text{ де } i,k = 0,\dots,M-1, j = 0,\dots,N-1$$

Таким чином, маємо r = F – 1 {F (x) F (x^)}, де F позначає перетворення Фур'є.

Логіка, що лежить в основі виявлення клонування, заснованого на автокореляції, полягає в тому, що вихідні та скопійовані сегменти введуть пікові точки функції в автокореляції до зміщення, що відповідають скопійованим переміщеним сегментам. Однак, оскільки природні зображення містять більшу

частину своєї потужності на низьких частотах, якщо автокореляція r обчислюється безпосередньо для самого зображення, r матиме дуже великі пікові точки в кутах зображення та їх околицях. Таким чином, ми обчислюємо автокореляцію не з зображення безпосередньо, а з його високопропускного фільтруваним варіантом. Припускаючи, що мінімальний розмір скопійованого переміщеного сегмента дорівнює B, метод виявлення переміщення копіювання-переміщення автокореляції складається з наступних етапів:

1. Застосування фільтра високих частот.

2. Обчислення автокореляцію r відфільтрованого зображення.

3. Видалення половини автокореляції (Автокореляція симетрична).

4. Встановлення r = 0 по сусідству з двома кутами всієї автокореляції.

5. Знайти максимум r, визначити вектор зсуву, використовуючи вичерпний метод (це зараз ефективно, тому що нам не доведеться виконувати вичерпний пошук багатьох різних векторів змін).

6. Якщо виявлена площа більша за B, закінчіть, інакше повторіть крок 5 із наступним максимумом r.

Хоча цей метод простий і не має великої обчислювальної складності, він часто не вдається виявити підробку, якщо розмір кованої площі не менше ¼ лінійних розмірів зображення згідно експериментів [10].

І вичерпний пошук, і метод автокореляції були відмовлені на користь блокового підходу, якій працює начно краще і швидше, ніж попередні підходи.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Формування вхідних даних

Будь-яка підробка клонування вводить залежність між початковим сегментом зображення та новим. Цей взаємозв'язок може бути використаний як основа для успішного виявлення клонування. Оскільки підробка, ймовірно, буде збережена у форматі JPEG, та оскільки можливе використання інструмента ретуші чи інших локалізованих інструментів обробки зображень, сегменти можуть не збігатися точно, а лише приблизно. Таким чином, ми можемо сформулювати наступні вимоги до алгоритму виявлення клонування:

1. Алгоритм виявлення повинен передбачати приблизну відповідність невеликих сегментів зображення.

2. Він повинен працювати в розумний час, та можливе виявлення кілька помилкових позитивних результатів (тобто виявлення неправильне узгодження областей).

3. Знайдений сегмент, швидше за все, буде цілісним компонентом, а не набором дуже маленьких патчів або окремих пікселів.

Ідея виявлення відповідності полягає в тому, що ми не впорядковуємо і не узгоджуємо піксельне представлення блоків, а використовуємо надійне подання блоків у вигляді їх квантових коефіцієнтів DCT, які використовуються в процесі стиснення JPEG.

Дискретне косинусне перетворення – це вид математичної ортогональної трансформації. Основна мета цих операцій - прийняти сигнал і перетворити його з одного типу подання в інший. Наприклад, зображення - це двовимірний сигнал, який сприймається зоровою системою людини. DCT може використовуватися для перетворення сигналу (просторової інформації) в числові дані ("частотна" або "спектральна" інформація), щоб інформація зображення була в кількісній формі, яка може бути маніпульована для стиснення. Сигнал для графічного

зображення можна розглядати як тривимірний сигнал. Осі X і Y сигналу зображення є двома розмірами екрана, тоді як амплітуда сигналу - вісь Z - це значення пікселя при (X, Y). Це може бути зображено візуально двовимірним масивом, де кожна комірка містить числове значення пікселя в цьому місці.

Етапи квантування обчислюються із визначеним параметром коефіцієнта якості Q . Цей параметр еквівалентний коефіцієнту якості при стисненні JPEG, тобто Q визначає етапи квантування для коефіцієнтів перетворення DCT. Оскільки більш високі значення коефіцієнта Q призводять до більш точного квантування, значення блоків повинні бути більше цього значення, щоб бути ідентифікованими як подібні. Більш низькі значення коефіцієнта Q збільшує кількість узгоджених блоків, можливо, деякі помилкові збіги.

Зображення сканується з верхнього лівого кута в нижній правий кут блоками B × B. Для кожного блоку обчислюється перетворення DCT, коефіцієнти DCT квантуються і зберігаються як один рядок у матриці A. Матриця матиме (M – B + 1) (N – B + 1) рядки та B × B стовпці як для конкретного випадку відповідності.

Рядки A лексикографічно сортуються. Оскільки, замість пікселів порівнюються квантовані значення коефіцієнтів DCT для кожного блоку, алгоритм може виявити занадто багато блоків відповідності (помилкових збігів). Таким чином, алгоритм також розглядає взаємні позиції кожної пари відповідних блоків і виводить певну пару блоків, тільки якщо в одній взаємній кількості є багато інших пар, що відповідають їх положенню (вони мають однаковий вектор зсуву). Для цього, якщо знайдено два послідовних рядки відсортованої матриці A, алгоритм зберігає позиції відповідних блоків в окремому списку (наприклад, координати верхнього лівого пікселя блоку можуть бути прийняті як його положення) і збільшується лічильник зсуву вектора зсуву C. Формально нехай (i_1, i_2) та (j_1, j_2) - позиції двох блоків, що співпадають. Вектор зсуву s між двома відповідними блоками обчислюється як

$$s = (s_1, s_2) = (i_1 - j_1, i_2 - j_2) \quad (3.1)$$

Оскільки вектори зсуву –s і s відповідають однаковому зсуву, вектори зсуву s нормалізуються, при необхідності, множенням на –1, так що s1 ≥ 0. Для кожної пари пар, що співпадає, збільшуємо нормований лічильник вектора зсуву C одним:

$$C(s_1, s_2) = C(s_1 , s_2) + 1 \quad (3.2)$$

Вектори зсуву обчислюються, а лічильник C збільшується для кожної пари послідовних співпадаючих рядків у відсортованій матриці A. Вектор зсуву C ініціалізується нулем до запуску алгоритму. В кінці процесу узгодження лічильник C вказує частоти, з якими трапляються різні нормалізовані вектори зсуву. Тоді алгоритм знаходить усі нормовані вектори зсуву s (1), s (2),…, s (K), поява яких перевищує заданий користувачем поріг T

$$C (s (r))> T \text{ для всіх } r = 1, \ldots, K. \quad (3.3)$$

Для всіх нормалізованих векторів зсуву блоки, які сприяли цьому конкретному вектору зсуву, пофарбовані одним кольором і таким чином ідентифікуються як сегменти, які могли бути скопійовані та переміщені.

Значення порогу T пов'язане з розміром найменшого сегмента, який можна ідентифікувати за алгоритмом. Більші значення можуть призвести до того, що алгоритм пропустить деякі не дуже близько узгоджувані блоки, тоді як занадто мале значення T може ввести занадто багато помилкових збігів. Qfactor контролює чутливість алгоритму до ступеня відповідності між блоками, тоді як розмір блоку B і поріг T контролюють мінімальний розмір сегмента, який можна виявити.

Для надійної відповідності обрано розмір блоку, B = 16, щоб запобігти занадто багато помилкових збігів (більші блоки мають більшу мінливість коефіцієнтів DCT). Однак більший блок розмір означає, що необхідно описату матрицю квантування 16 × 16, а не використовувати стандартну матрицю квантування JPEG, яка є для блоку 8 x 8 .

Матриця квантування коефіцієнтів DCT у кожному блоці 16 × 16, має такий вигляд:

$$Q_{16} = \begin{pmatrix} Q_8 & 2.5q_{18}I \\ 2.5q_{81}I & 2.5q_{88}I \end{pmatrix}, \text{де } Q_8 = \left(\begin{pmatrix} 2.5q_{00} & \cdots & 2.5q_{18} \\ \vdots & \ddots & \vdots \\ 2.5q_{81} & \cdots & 2.5q_{88} \end{pmatrix} \right)$$

q_{iy} є стандартною матрицею квантування JPEG з коефіцієнтом якості Q, а I є одиничною матрицею 8 × 8. Ця форма є досить специфічною, але матриця дала дуже хороші показники на практичних тестах. Невеликі зміни в матриці дуже мало впливають на результати.

Якщо на вхід подається кольорове зображень: і в точній, і в надійній відповідності, якщо аналізоване зображення є кольоровим зображенням, воно спочатку перетворюється на зображення в градаціях сірого перш ніж продовжувати подальший аналіз за формулою:

$$I = 0{,}299\ R + 0{,}587\ G + 0{,}114\ B \quad (3.5)$$

Реалізований алгоритм, є варіацією блочного алгоритму.

3.2 Програмні продукти для проектування системи

MATLAB - це обчислювальна середовище та мову програмування. Підтримуваний MathWorks, MATLAB дозволяє легко маніпулювати матрицями, будувати функції і дані, реалізовувати алгоритми, створювати призначені для користувача інтерфейси і взаємодіяти з програмами на інших мовах. Область його застосування безмежно широка: IoT, фінанси, медицина, космос, автоматика, робототехніка, бездротові системи та багато-багато іншого. Загалом

майже необмежені можливості по збору і візуалізації даних, а також прогнозування.

Мова MATLAB - інструмент, що забезпечує взаємодію оператора, або програміста з усіма доступними можливостями аналізу, збору і представлення даних. У нього є очевидні плюси і мінуси, властиві мові живе в замкнутій екосистемі. [6]

Недоліки:

- Повільний і перевантажений операторами, командами, функціями мову, основною метою якого є поліпшення візуального сприйняття.
- Вузьконаправлений. Немає ніякої більше програмної платформи, де б MATLAB був корисний;
- дорожнеча ПО (є безплатна версія для студентів);
- невисокий попит. Незважаючи на великий інтерес до MATLAB практично у всіх сферах, фактично і легально його використовують лише деякі.

Переваги:

- мова легка для вивчення, має простий і зрозумілий синтаксис;
- величезні можливості. Але це перевага всього продукту в цілому;
- часті оновлення, як правило помітні позитивні перетворення відбуваються не рідше пари раз на рік;
- Програмне середовище дозволяє перетворювати його в "швидкий" код на C, C++.

Проаналізувавши всі недоліки, та переваги Mathlab була обрана як інструментальний засіб для розробки.

3.3 Короткий опис програмної реалізації алгоритму

Алгоритм було реалізовано за допомогою мови програмування Mathlab у вигляді проекту cpmvded.mtlab. В файлі compdctmatrix.mtlab були описані матриця квантування, та функція обчислення DCT.

Таблиця 3.1 – Основні функції які були використані в satpr.cpp.

№	Назва процедури	Короткий опис
1	copy_move_2_adj(color_image, quality_factor, threshold, not_color)	Основна функція запуска програми. В функцию передається картинка, фактор якості,, та чи кольорова картинка
2	compute_dct_matrix()	Обчислення DCT блока
3	im2col()	Стандартна функція mathlab. розбиття на блоки зображення
4	sortrows()	Стандартна функція mathlab. Використанна для сортування
5	shift_vector()	Стандратна фунційя mathlab. Використана для обчислення вектора ссува

Програма працює за наступним алгоритмом:

1. Розбиваємо зображення на блоки (16x16)

2. Для кожного блока 16x16 зображення, обчислюємо дискретне косинусне перетворення (DCT) та зберігаємо його в матрицю. Спочатку знаходимо верхній лівий кут цього блоку на зображенні потім перетворюємо кожен блок у векторний рядок, та обчислюємо його DCT

3. Сортування матриці DCT. Зберігаймо x, у верхнього лівого кута кожного блоку в останніх двох стовпцях цієї матриці, а потім сортуємо матрицю на основі перших 1 - n-2 стовпців (тим самим зберігаючи інформацію даних про блоки, щоб їх можна було, коли ми хочемо в зображенні).Виходить дві матриці, одна з локаціями та інша зі значеннями DCT.

4. Знаходимо відповідні блоки та будуємо вектори руху. Тут нам потрібно переконатися, що ми порівнюємо всі стовпці, окрім останніх двох, щоб ми не випадково порівнювали місця положення одного блоку на зображенні.

5. Порівнюємо кожен рядок із суміжним його рядком, якщо вони рівні захоплюємо координати цих блоків у зображення, та обчислюємо вектор зсуву між двома відповідними блоками та зберігаємо.

6. Тепер ми маємо підрахунок усіх векторів руху та кількість їх появи. Якщо є вектор руху, який відбувся більше порогового значення, то знаходимо усі входження цього вектора руху та фарбуємо ці пари.

Текст програми наведено у додатку.

3.3 Тестування розробленої інформаційної системи

Реалізований алгоритм був протестований декількома різними фотографіями. Показано, що алгоритм добре працює на зображеннях з областями клонування, які не включають масштаб або обертання. На рис. 3.1 алгоритм запускається на фотографії з п'ятьма предметами, на тротуарі. Незважаючи на схожість бруківки, при використанні порогу 10 та коефіцієнта якості 0,5 алгоритм міг правильно визначити, де було клонування. У вихідному маніпульова область виділяється і відображається зеленим кольором. Інший приклад наведено на рис. 3.2. На цій фотографії було повністю скритий прапор, за допомогою клонування певних частин неба, а також використано інструмент ретушування (замилення), для ускладнення розпізнавання клонованих частин. За допомогою коефіцієнта якості 0.5 та порогу 50 алгоритм зміг виявити аномалії, при цьому є певні неточності. При цьому при тестуванні оригіналу з цими параметрами, аномалій не було виявлено.

В третьому тесті було використано велике за розміром картинку, з багатьма елементами. В якості об’єкта клонування була обрана людина. Тест зміг розпізнати клоновані регіони, але також неправильно визначив регіони, такі як стіни, текстура одягу. Ці регіони будуть висвітлюватися алгоритмом випадковим

чином і виглядатимуть як шум на виході. Покращити результат можливо збільшенням розміру блока для тестування, та зміни параметрів якості та порогу.

Рисунок 3.1 – Перше вхідне зображення з клонуванням

Рисунок 3.2 – Результат виявлення клонування на першому зображенні

Рисунок 3.3 – Друге вхідне зображення (справа орігінал, зліва підробка)

Рисунок 3.4 – Результат виявлення клонування на другому зображенні

Рисунок 3.5 – Третє вхідне зображення (клонована людина)

Рисунок 3.6 – Результат виявлення клонування на третьому зображенні

Цей алгоритм працює для підроблених підробок, які не мають обертання або масштабування. Однак пошук зображень, які були піддані маніпулюванню, щоб включати обертові та масштабовані регіони, є загальним явищем, і ефективний алгоритм для виявлення клонування також повинен був би визначити ці типи підробок. Надалі варто було б дослідити інші дескриптори, інваріантні обертанню та масштабуванню.

ВИСНОВКИ

В роботі було виконано інформаційний аналіз і синтез системи виявлення елементів зображення, що були клоновані. При цьому застосовується метод блокового співставлення частин зображень. В ході роботи було розв’язано такі задачі:

1) Виконано порівняльний аналіз методів, алгоритмів і програмних засобів, що дозволяють розв’язати задачу виявлення клонованих частин на зображенні.
2) Сформовано математичний опис системи виявлення елементів зображення, що були клоновані.
3) Розроблено та програмно реалізовано алгоритм блокового співставлення елементів зображення, що аналізується.
4) Розроблено та програмно реалізовано алгоритм пошуку за ромбічним шаблоном.
5) Розроблено та програмно реалізовано алгоритм адаптивного пошуку за шаблоном.
6) Перевірено працездатність системи виявлення елементів зображення, що були клоновані.

СПИСОК ЛІТЕРАТУРИ

1. Нгуен Ван Чыонг. Разработка и исследование методов повышения эффективности сжатия в современных видеокодеках/ Нгуен Ван Чыонг//. - М.: Санкт-Петербург – 2018
2. Seadle M. Quantifying Research Integrity. Humboldt-Universitat Berlin. 2017
3. Y. Wu, W. Abd-Almageed, and P. Natarajan, “BusterNet: Detecting Copy-Move Image Forgery with Source/Target Localization,” in Computer Vision – ECCV 2018, vol. 11210, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 170–186.
4. Thorsten S. Beck. How to Detect Image Manipulations - Clone Detection in Practice. 2017: https://headt.eu/How-to-Detect-Image-Manipulations-Part-2/
5. Thorsten S. How to Detect Image Manipulations Part IV: InspectJ and ImageJ. 2017: https://headt.eu/How-to-Detect-Image-Manipulations-Part-4/
6. Choraś M., Choraś R.S. (Eds.) Image Processing and Communications: Techniques, Algorithms and Applications. – Springer, 2020. — 337 p
7. Oliva D., Hinojosa S. (Eds.) Applications of Hybrid Metaheuristic Algorithms for Image Processing. – Springer, 2020. — 486 p.
8. Laplante P.A. (ed.) Encyclopedia of Image Processing. – Boca Raton: CRC Press, 2019. — 875 p.
9. Vyas A., Yu S., Paik J. Multiscale Transforms with Application to Image Processing. – Springer, 2018. — 258 p.
10. Дьяконов В. П. MATLAB 6.5/7.0 + Simulink 5/6 в математике и моделировании. Библиотека профессионала. — М.: «СОЛОН-Пресс», 2005. — 576 с. — ISBN 5-98003-209-6.
11. Şen Z. Earth Systems Data Processing and Visualization Using MATLAB. – Springer, 2020. — 284 p.
12. Yaroslavsky L.P. Advanced Digital Imaging Laboratory Using MATLAB .– IOP Publishing. 2014. — 124

ДОДАТОК

```
function Q_16x16 = compute_dct_matrix()
 Q_8x8 =[ 4, 4, 6, 11, 24, 24, 24, 24 ;
          4, 5, 6, 16, 24, 24, 24, 24 ;
          6, 6, 14, 24, 24, 24, 24, 24;
          11, 16, 24, 24, 24, 24, 24, 24;
          24, 24, 24, 24, 24, 24, 24, 24;
          24, 24, 24, 24, 24, 24, 24, 24;
          24, 24, 24, 24, 24, 24, 24, 24;
          24, 24, 24, 24, 24, 24, 24, 24; ];
Q_8x8_prime = Q_8x8;
Q_8x8_prime(1,1) = 2*Q_8x8_prime(1,1);
Q_8x8_prime(2:8,2:8) = 2.5*Q_8x8_prime(2:8,2:8);%Top left
corner block
Q_18 = zeros(8,8) + 2.5*Q_8x8(1,8); %top right corner block
Q_81 = zeros(8,8) + 2.5*Q_8x8(8,1); %bottom left corner
block
Q_88 = zeros(8,8) + 2.5*Q_8x8(8,8); %bottom right corner
block
%Q_16x16 is the concactenation of all of the blocks
described above
Q_16x16 = vertcat(horzcat(Q_8x8_prime, Q_18),horzcat(Q_81,
Q_88));
end

disp('sort rows of dct matrix');
dct_matrix = sortrows(dct_matrix, 1:size(dct_matrix,2) - 2
);
dct_locs = dct_matrix(:,size(dct_matrix,2) - 1: size(
dct_matrix,2));
dct_matrix = dct_matrix(:, 1:size(dct_matrix,2)-2);
if nargin < 4 || isequal(not_color, false)
    image_copy = color_image;
    input_image = 255*im2double(rgb2gray(color_image));
    %verified that the conversion to grayscale is same
    used in the paper
elseif isequal(not_color, true)
    image_copy = color_image;
    input_image = color_image;
end
```

```
block_size = 16; %want 16x16 blocks
%quality_factor = 0.5;
%threshold = 10;
Q_16x16 = compute_dct_matrix();
[height, width] = size(input_image);
patches = im2col(input_image(:,:,1),[block_size,block_size
],'sliding');
[m1, n1] = size(patches);
num_blocks = (1:n1);
disp('Computing the DCT Transform of each 16x16 window');
num_rows = (size(input_image, 1) - block_size + 1) * (size
input_image, 2) - block_size + 1);
dct_matrix = zeros(num_rows, block_size*block_size + 2);
%the plus two is for xy locationof block in image
[rows,cols] = ind2sub(size(input_image(:,:,1))- block_size
+ 1, num_blocks);
for ind = 1:num_rows %for every 16x16 window in the image,
compute
    x1 = rows(ind);
    y1 = cols(ind);
    temp_mat = reshape(patches(:,ind), [block_size
    block_size]);
    temp_mat = round((dct2(temp_mat)./quality_factor)./
    Q_16x16);
    temp_mat = temp_mat(:)';
    dct_matrix(ind,:) = [temp_mat, x1, y1];
end
disp('Constructing shift vectors for all the matching
block pairs');
num_match_index = 1;
shift_vector = zeros(num_rows, 2);
match1 = zeros(num_rows, 2);
match2 = zeros(num_rows, 2);
shift_vector_count = zeros(max(height+1, width+1),max(
height+1, width+1));
for ind = 1:num_rows - 1
    if isequal(dct_matrix(ind,:), dct_matrix(ind + 1,:))
    %compare every row to its adjacent row
        x1y1 = dct_locs(ind, :);
```

```
        x2y2 = dct_locs(ind + 1, :);
        shift_vector(num_match_index,:) = [abs(x1y1(1) -
        x2y2(1)), abs(x1y1(2) - x2y2(2))];
        if ~isequal(shift_vector(num_match_index,:), [0,1])
         && ...
            ~isequal(shift_vector(num_match_index,:), [
            1,0]) && ...
            ~isequal(shift_vector(num_match_index,:), [
            1,1])

            match1(num_match_index,:) = x1y1;
            match2(num_match_index,:) = x2y2;
            shift_vector_count(abs(x1y1(1) - x2y2(1)) + 1,
            abs(x1y1(2) - x2y2(2)) + 1) ...
                = shift_vector_count(abs(x1y1(1) - x2y2(1))
                 + 1, abs(x1y1(2) - x2y2(2)) + 1) + 1;
            num_match_index = num_match_index + 1;
            %incrememnt index in structure
        end
    end
end
match1 = match1(1:num_match_index, :);
match2 = match2(1:num_match_index, :);
shift_vector = shift_vector(1:num_match_index, :);
disp('Looking for common shift vectors among matching
blocks');
 [shiftx,shifty] = ind2sub(size(shift_vector_count), find(
 shift_vector_count > threshold));
disp(strcat('there are:',{' '}, num2str(size(min(shiftx,
shifty),1)), ' shift vectors'));
for ind = 1:size(min(shiftx,shifty), 1)
    locs = ind2sub(size(shift_vector,1), ...
        find(shift_vector(:,1) == (shiftx(ind) - 1) &
        shift_vector(:,2) == (shifty(ind) - 1)));
    for ind2 = 1:size(locs, 1)        x1y1 = match1(locs(
    ind2),:);
        x2y2 = match2(locs(ind2),:);
         %color them the same color
         image_copy(x1y1(1):x1y1(1)+block_size - 1, x1y1(2
```

```
        ):x1y1(2)+block_size - 1, 1 + mod(ind,3)) = 250;
        image_copy(x2y2(1):x2y2(1)+block_size - 1, x2y2(2
        ):x2y2(2)+block_size - 1, 1 + mod(ind,3)) = 250;

        test_image(x1y1(1):x1y1(1)+block_size - 1, x1y1(2
        ):x1y1(2)+block_size - 1, 1 + mod(ind,3)) = 1;
        test_image(x2y2(1):x2y2(1)+block_size - 1, x2y2(2
        ):x2y2(2)+block_size - 1, 1 + mod(ind,3)) = 1;
end end
```