Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Васалатій Роман Ігорович,
студент групи РЗ-181

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Модифікація алгоритму виявлення та локалізації областей клонування в цифрових зображеннях

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Лебедєва Олена Юріївна
к.т.н., доцент

Одеса – 2022

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ

_____
д.т.н.,проф. А.А.Кобозєва
____ _____202_р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Васалатію Роману Ігоровичу

1.Тема роботи: *Модифікація алгоритму виявлення та локалізації областей клонування в цифрових зображеннях*,
керівник роботи *Лебедєва Олена Юріївна*, *к.т.н., доцент,*
затверджені наказом ректора від „17”05. 2022 р. №168-в.
2. Зміст роботи: *види фальсифікацій зображень, огляд проблеми виявлення порушення цілісності цифрового зображення, цифрові зображення та їх представлення, методи клонування областей, алгоритм виявлення та локалізації клонованих областей, інтерфейс програмного продукту та результати експериментів*.
3. Перелік ілюстративного матеріалу: *приклади растрового та векторного зображень*, *приклад представлення зображення, блоки квадратної форми, маркери блоку квадратної форми, інтерфейс головного вікна розробленого програмного додатку*.

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
Охорона праці	Доц. Ярова І.А.	Завдання видав	Завдання рийняв

5. Дата видачі завдання "_____"_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	*15.11.2021*	*виконано*
2	*Обґрунтування вибору рішення. Збір даних*	*15-12-2021*	*виконано*
3	*Аналіз методів та засобів спілкування людей та захисту інформації в месенджерах*	*11-01-2022*	*виконано*
4	*Аналіз методів шифрування даних*	*20-02-2022*	*виконано*
5	*Розробка та програмна реалізація алгоритму шифрування*	*30-03-2022*	*виконано*
6	*Підготовка тексту роботи*	*05-05-2022*	*виконано*
7	*Підготовка презентації та доповіді*	*31-05-2022*	*виконано*
8	*Попередній захист*	*02-06-2022*	*виконано*
9	*Нормоконтроль, рецензування*	*20-06-2022*	*виконано*

Здобувач вищої освіти _____ *Васалатій Р.І.*

Керівник роботи _____ *Лебедєва О.Ю.*

ЗАВДАННЯ

на розробку розділу «Охорона праці» у кваліфікаційній роботі бакалавра

студенту *Васалатію Роману Ігоровичу*

Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій

Кафедра кібербезпеки та програмного забезпечення

Дата отримання завдання 09.06.2022

Консультації 09.06.2022, 14.06.2022

Дата закінчення розділу 14.06.2022

Тема роботи «Модифікація алгоритму виявлення та локалізації областей клонування в цифрових зображеннях»

Зміст розділу

1. Аналіз умов праці і вибір основних заходів виробничої безпеки
2. Аналіз пожежної безпеки і вибір заходів і засобів пожежної безпеки

Керівник дипломної роботи

_____ Лебедєва О.Ю.
(підпис)

« __ » _____2022 р.

Консультант з охорони праці

_____ Ярова І. А.
(підпис)

« __ » _____ 2022 р.

АНОТАЦІЯ

Кваліфікаційна робота на тему “Модифікація алгоритму виявлення та локалізації областей клонування в цифрових зображеннях” на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 17 рисунків, 3 таблиці, 1 додаток, 24 літературних джерела за переліком посилань. Робота виконана на 53 сторінках загального тексту і 42 сторінках основного тексту.

Метою роботи є підвищення ефективності алгоритму виявлення та локалізації областей клонування в цифрових зображеннях шляхом його модифікації, заснованої на використанні маркерів.

У роботі проведено огляд проблем виявлення порушень цілісності цифрових зображень. Проаналізовано алгоритм виявлення та локалізації областей клонування та запропоновано використання маркерів блоків для підвищення ефективності роботи алгоритму виявлення та локалізації областей клонування в цифрових зображеннях.

У результаті виконання кваліфікаційної роботи розроблено модифікацію алгоритму виявлення та локалізації областей клонування в цифрових зображеннях та програмно реалізовано у вигляді додатку, за допомогою якого можна виявити та локалізувати клонування в цифровому зображенні. У майбутньому цей додаток може бути використаний в виявленні фальсифікації зображення, або може бути впроваджений в комплексну програму виявлення фальсифікацій зображення, яка може використовуватися в будь-якій сфері діяльності.

ЦИФРОВЕ ЗОБРАЖЕННЯ, ФАЛЬСИФІКАЦІЯ ЗОБРАЖЕННЯ, КЛОНУВАННЯ, МАРКЕРИ БЛОКУ ЗОБРАЖЕННЯ.

ANNOTATION

Qualification work on "Modification of the algorithm for detection and localization of cloning areas in digital images" for the first (bachelor's) level of higher education in 125 Cybersecurity, specialization, educational program: Cybersecurity, contains 17 figures, 3 tables, 1 appendix, 24 references sources on the list of links. The work is performed on 52 pages of general text and 42 pages of main text.

The aim of the work is to increase the efficiency of the algorithm for detecting and localizing cloning areas in digital images by modifying it based on the use of markers.

The paper reviews the problems of detecting violations of the integrity of digital images. The algorithm for detecting and localizing cloning areas is analyzed and the use of block markers is proposed to increase the efficiency of the algorithm for detecting and localizing cloning areas in digital images.

As a result of qualification work, a modification of the algorithm for detecting and localizing cloning areas in digital images was developed and implemented in software as an application that can detect and localize cloning in a digital image. In the future, this application can be used to detect image falsification, or can be implemented in a comprehensive program to detect image falsification, which can be used in any field of activity.

DIGITAL IMAGE, IMAGE FALSIFICATION, CLONING, IMAGE BLOCK MARKERS.

ЗМІСТ

ВСТУП

Значну роль у сучасному світі відіграють цифрові сигнали, зокрема, цифрові зображення (ЦЗ), що використовуються у науці, медицині, судових розглядах, пресі тощо. Сьогодні камера є у кожного, якщо подія значуща, то у вашому розпорядженні опиняться тисячі фотографій. Проте змінити фотографію дуже просто. Це можуть зробити навіть любителі. Часто такі фальсифікації практично неможливо виявити неозброєним оком.

Багато графічних редакторів знаходяться у відкритому доступі в мережі інтернет, наприклад Adobe Photoshop, GIMP, Paint, Paint3D та інші, не враховуючи онлайн сервіси. І виникає питання, як захистити себе та інших людей від фальсифікації, а саме фальсифікації методом клонування елементів зображення, в мережі інтернет.

Через загальнодоступність програмного забезпечення, що дозволяє обробляти і редагувати ЦЗ, а також у зв'язку з різноманітністю способів їх фальсифікації, зростає необхідність у вдосконаленні існуючих та розробці нових методів виявлення порушення цілісності ЦЗ, що є обов'язковою складовою частиною будь-якої сучасної комплексної системи захисту інформації. Тому зростає актуальність методів виявлення порушення цілісності цифрових зображень.

Метою роботи є підвищення ефективності алгоритму виявлення та локалізації областей клонування в цифрових зображеннях шляхом його модифікації, заснованої на використанні маркерів.

Для вирішення поставленої мети необхідно вирішити такі завдання:

- огляд методів та засобів виявлення областей фальсифікації в цифрових зображеннях;
- проаналізувати алгоритм виявлення та локалізації областей клонування в цифрових зображеннях та провести вибір маркерів для блоків, які використовуються для виявлення областей фальсифікації;
- розробити модифікацію алгоритму виявлення та локалізації областей

клонування в цифрових зображеннях.

– програмно реалізувати розроблену модифікацію алгоритму виявлення та локалізації областей клонування в цифрових зображеннях та оцінити її ефективність.

Під ефективністю в цій роботі будемо розуміти зменшення часу роботи алгоритму виявлення та локалізації областей клонування в цифрових зображеннях.

Об’єктом дослідження є процеси несанкціонованої зміни цифрових зображень.

Методи дослідження базуються на теорії алгоритмів та методах обробки зображень, які використовувались при розробці модифікації методу виявлення та локалізації областей клонування в цифрових зображеннях.

Практична цінність роботи полягає у тому, що удосконалено алгоритм виявлення клонування, як фальсифікації цифрового зображення, шляхом модифікації із використанням маркерів, що призвело до зменшення часу, необхідного для виявлення результатів клонування цифрових зображень засобами графічних редакторів, та програмна реалізація розробленої модифікації, яка може бути використана будь-якою людиною з метою перевірки цифрового зображення на предмет наявності фальсифікації.

1 ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ ВИЯВЛЕННЯ ОБЛАСТЕЙ ФАЛЬСИФІКАЦІЇ В ЦИФРОВИХ ЗОБРАЖЕННЯХ

1.1 Види фальсифікації зображень

Завдяки наявності потужних комп’ютерів, розвитку інформаційних технологій та програмного забезпечення – цифровими зображеннями легко маніпулювати та редагувати. На сьогоднішній день можна легко та без аби-яких навичок додавати чи видаляти важливі об’єкти із зображення не залишаючи очевидних слідів підробки. Тому виникає питання захисту та виявлення фальсифікації.

Під фальсифікацією цифрового зображення будемо розуміти порушення цілісності цифрового зображення.

Фальсифікація цифрового зображення – пошкодження або зміна цифрового зображення, при виконанні будь-якої операції над ним, в наслідок передачі, зберігання або відтворення.

Способами порушення цілісності цифрового зображення є наступне.

– клонування;
– фотомонтаж;
– обробка цифрового зображення.

Клонування – частина самого зображення копіюється та вставляється в іншу частину того ж самого цифрового зображення (рисунок 1.1).

Зазвичай клонування робиться з наміром змусити об'єкт «зникнути» з зображення, покривши його сегментом, скопійованим з іншої частини зображення, або навпаки продублювати потрібний об’єкт стільки разів скільки нам потрібно. Оскільки скопійовані частини походять із одного зображення, його шумовий компонент, кольорова палітра, динамічний діапазон та більшість інших важливих властивостей будуть сумісні з рештою зображення і, таким чином, не буде виявлено за допомогою методів, які шукають несумісність у статистичних заходах в різних частинах зображення.

Рисунок 1.1 – Приклад фальсифікації цифрового зображення шляхом клонування

Фотомонтаж – процес і результат створення зображень, складених з частин різних фотографій (рисунок 1.2). Фотомонтаж широко застосовується при виготовленні плакатів, реклам, політичних карикатур тощо. По суті фотомонтаж – це вирізання будь-яких об'єктів на фотографії і з'єднання (поєднання) їх з іншою фотографією.

Рисунок 1.2 – Приклад фальсифікації цифрового зображення шляхом фотомонтажу

До фотомонтажу можна віднести такі зміни, внесені у фотографії [1]:

– заміна заднього фону;
– об'єднання кількох різних зображень в одне;

– заміна персонажа іншим або видалення його з фотографій;
– проведення різних маніпуляцій, які сильно змінюють те, що було зображено спочатку на фотографії.

Механічний фотомонтаж – з фотографій вирізують потрібні зображення, підганяють їх шляхом збільшення під необхідний масштаб, склеюють на аркуші паперу, ретушують, потім перезнімають.

Проекційний фотомонтаж – на фотопапері послідовно друкують зображення з декількох негативів. При цьому нерідко використовують маски, послідовно перекриваючи ними ті чи інші частини негативу.

Комп'ютерний (цифровий) фотомонтаж – редагування зображень здійснюється переважно на комп'ютері растровими графічними редакторами в цифровому вигляді. Для цього зображення, навіть отримане з традиційного носія (фотоплівки), переводиться в цифровий вигляд – наприклад, за допомогою сканера.

З розвитком комп'ютерної техніки з'явився цілий арсенал програмного забезпечення для фотомонтажу. Найбільш поширені програми для фотомонтажу – графічні редактори Adobe Photoshop, PaintShop Pro, Corel Photo-Paint, GIMP, Ulead PhotoImpact.

Обробка зображень – будь-яка форма обробки інформації, для якої вхідні дані представлені зображенням, наприклад, фотографіями або відеокадрами (рисунок 1.3). Обробляння зображень може здійснюватися як для отримання зображення на виході (наприклад, підготовка до поліграфічного тиражування, до телетрансляції тощо), так і для отримання іншої інформації (наприклад, розпізнання тексту, підрахунок числа і типу клітин в полі мікроскопа і т. д.). Крім статичних двомірних зображень, обробляти потрібно також зображення, що змінюються з часом, наприклад відео.

У цифровій обробці зображень широко застосовується спеціалізоване обладнання, таке як процесори з конвеєрною обробкою інструкцій та багатопроцесорні системи. Особливою мірою це стосується систем обробки відео. Обробка зображень виконується також за допомогою програмних засобів

комп'ютерної математики, наприклад, MATLAB, Mathcad, Maple, Mathematica і інші. Для цього в них використовуються як базові засоби, так і пакети розширення Image Processing.

Рисунок 1.3 – Приклад фальсифікації цифрового зображення шляхом обробки цифрового зображення

Типові завдання обробки зображення є нижче наведені варіанти.

– геометричні перетворення, такі як обертання і масштабування;
– колірна корекція: зміна яскравості і контрасту, квантування кольору, перетворення в інший колірний простір;
– порівняння двох і більше зображень. Як окремий випадок – знаходження кореляції між зображенням і зразком, наприклад, в детекторі банкнот;
– комбінування зображень різними способами;
– інтерполяція і згладжування;
– поділ зображення на області (сегментація зображень), наприклад, для

спрощення передачі каналами зв'язку;

– редагування та ретушування;
– розширення динамічного діапазону шляхом комбінування зображень з різною експозицією;
– компенсація втрати різкості, наприклад, шляхом нерізкого маскування.

У обробці сигналів широко використовуються перетворення Фур'є, а також вейвлет-перетворення і фільтр Габора. Обробку зображень поділяють на обробку в просторовій області (перетворення яскравості, контрасту, гама-корекція тощо) і частотній (перетворення Фур'є, фільтрація тощо).

1.2 Огляд проблеми виявлення порушень цілісності цифрового зображення

Для всебічного вирішення питань інформаційної безпеки ефективною є комплексна система захисту інформації (КСЗІ), що поєднує в собі наступні заходи [2]: законодавчі, морально-етичні, фізичні, адміністративні, технічні, криптографічні та програмні.

Усі методи захисту інформації можна розділити на методи активного захисту (МАЗІ), спрямовані на запобігання несанкціонованого доступу, витоку, зміни інформації, і методи пасивного захисту інформації (МПЗІ), призначені для того, щоб визначити, чи було зроблено навмисне порушення цілісності інформації [3].

МАЗІ за способом їх реалізації поділяють на програмні, криптографічні, технічні та організаційні. МПЗІ в свою чергу поділяють за способом їх реалізації на методи експертної оцінки, програмно-технічні та програмні.

Один з найбільш ефективних програмно-технічних методів захисту ЦЗ від несанкціонованих змін заснований на аналізі вбудованого в об'єкт, що захищається, цифрового водяного знаку (ЦВЗ) [4 – 9]. Розробки в цій галузі ведуть найбільші фірми в усьому світі. На відміну від звичайних водяних знаків ЦВЗ можуть бути не тільки видимими, але і (як правило) невидимими. ЦВЗ можуть містити деякий автентичний код, інформацію про власника, або керуючу інформацію [4].

Широке поширення сьогодні набули методи, засновані на аналізі EXIF-

даних – додаткової інформації, що додається в медіафайли цифровою технікою безпосередньо при їх створенні [4]. За допомогою EXIF-даних можна встановити умови і способи отримання медіафайлу, авторство, координати місця зйомки (за наявності вбудованого приймача GPS) тощо.

Основним недоліком програмно-технічних методів захисту інформації є жорстка прив'язаність до технічного пристрою, його можливостей і властивостей або впливу оточуючих факторів на запис сигналу. Крім того в більшості випадків при виявленні фальсифікації дані методи не здатні локалізувати її область.

На відміну від програмно-технічних і експертних методів програмні методи не мають прив'язки до технічних пристроїв, за допомогою яких було отримано інформацію, а також не вимагають участі експерта в ідентифікації порушень її цілісності.

На даний час активно розвивається галузь експертизи цифрових контентів, створюються нові та вдосконалюються існуючі програмні методи виявлення порушень цілісності ЦЗ, таких як клонування (заміна частини основного зображення замінюється частиною цього ж зображення) [10-14, 19, 20], колаж (комбінація частин різних зображень) [15], масштабування (зміна розмірів та (або) поворот частин ЦЗ) [16-17], корекція яскравості [18, 21], постобробка ЦЗ після його фальсифікації (ретуш, зміна різкості, регулювання контрасту, розмиття).

Під порушенням цілісності у даній роботі будемо розуміти несанкціоновану зміну цифрового зображення.

2 РОЗРОБКА МОДИФІКАЦІЇ АЛГОРИТМУ ВИЯВЛЕННЯ ТА ЛОКАЛІЗАЦІЇ КЛОНУВАННЯ В ЦИФРОВИХ ЗОБРАЖЕННЯХ

2.1 Цифрові зображення та їх представлення

Зображення – об'єкт, образ, явище, в тій або іншій мірі подібне, але не ідентичне змальовуваному або сам процес їх створення.

Під зображенням розуміється інформація, організована у вигляді деякої квадратної числової матриці, відтворююча властивості змальованого об'єкту (сцени) і деформації, які пов'язані із способом і процесом здобуття зображення.

Виходячи з особливостей інформаційної надмірності зображень, можна виділити наступні основні класи зображень [22].

Клас 1. Монохромні зображення (відскановані креслення і текстові документи). Містять великі області одного кольору, контекстні залежності між сусідніми пікселями, часто кількість пікселів одного кольору сильно перевищує кількість пікселів іншого.

Клас 2. Кольорові зображення з невеликим числом кольорів з певної палітри (ділова графіка, деякі види мультиплікації). Містять великі області одного кольору, контекстні залежності між сусідніми пікселями, часто кількість пікселів одного кольору сильно перевищує кількість пікселів іншого.

Клас 3. Зображення в градаціях сірого (рентгенівські знімки, чорно-білі фотографії). Значення яскравості сусідніх пікселів зазвичай мало відрізняються. Характерна плавна зміна яскравості між різними ділянками зображення.

Клас 4. Повнокольорові зображення (кольорові фотографії, фотореалістичні тривимірні сцени). Значення колірних компонент сусідніх пікселів зазвичай мало відрізняються. Характерна плавна зміна значень колірних компонент між різними ділянками зображення. Крім того, є області, неоднорідні по яскравості, але що мають однаковий відтінок і насиченість кольорів.

Надалі у нашій роботі ми будемо розглядати зображення четвертого класу, тобто повнокольорові зображення.

Існують три основні способи цифрового представлення зображень:

– векторна графіка;
– растрова графіка;
– фрактальна графіка.

У векторній графіці зображення будуються з простих об'єктів - прямих ліній, дуг, кіл, еліпсів, прямокутників, областей однотонного або змінного кольору (наповнювачів) і т. п., які називаються примітивами (рисунок 2.1).

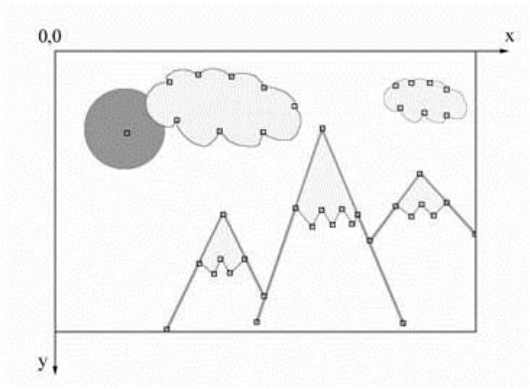

Рисунок 2.1 – Векторне зображення та вузли його примітивів

Векторна графіка не залежить від роздільної здатності, тобто може бути показана в різноманітних вихідних пристроях з різним роздільними здатностями без втрати якості. Збільшення або зменшення об'єктів приводе к збільшенням або зменшенням відповідних коефіцієнтів в математичних формулах.

Фрактальна графіка, як і векторна, заснована на математичних обчисленнях. Базовим елементом фрактальної графіки є сама математична формула, тобто ніяких об'єктів в пам'яті комп'ютера, не зберігається і зображення будується виключно по рівняннях (рисунок 2.2).

Растрова графіка описує зображення з використанням кольорових крапок, названих пікселями, розташованих на сітці (рисунок 2.3).

Піксель – основний елемент растрових зображень, це одна клітинка. Саме із сукупності пікселів і складається растрове зображення.

Розміри зображення і розташування пікселів в ньому – це дві основні характеристики, які файл растрових зображень повинен зберегти, щоб створити картинку. Ще одна – колір. Наприклад, зображення описується конкретним

розташуванням кожної точки сітки, що створює зображення приблизно так як в мозаїці.

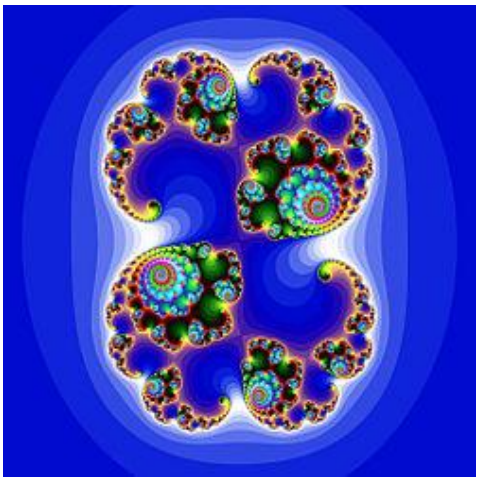

Рисунок 2.2 – Приклад фрактального зображення

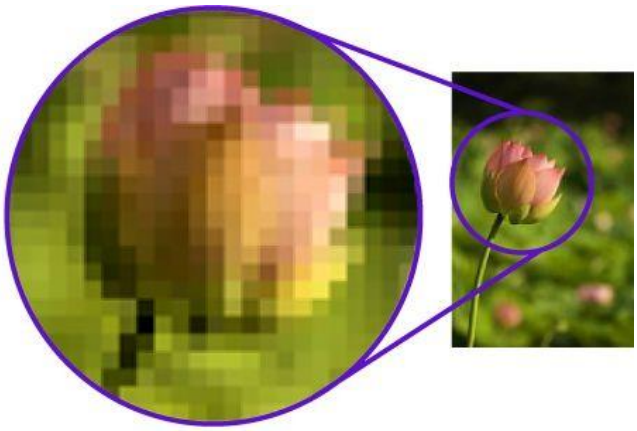

Рисунок 2.3 – Приклад растрового зображення

При редагуванні растрової графіки редагуються пікселі, а не лінії, як у векторній графіці. Растрова графіка залежить від роздільної здатності, оскільки інформація, що описує зображення, прикріплена до сітці певного розміру. При редагуванні растрової графіки, якість її подання може змінитися.

Роздільна здатність – це кількість пікселів на одиницю довжини, найчастіше на дюйм - dpi, причому, чим вище дозвіл, тим більше пікселів поміщається в дюймі і тим якісніше зображення. Глибина кольору визначає ту кількість відтінків, в діапазоні яких точка може змінювати свій колір.

Растрова графіка ефективно представляє зображення фотографічної якості. На растровому зображенні може бути зображено все, що завгодно: як знімок з фотокамери, так і намальоване на комп'ютері зображення. До растрових зображень можна застосовувати найрізноманітніші ефекти. Растрові формати зображень використовуються при створенні веб-сторінок в Інтернеті.

Растрова графіка використовується при обробці зображень відомими графічними редакторами, такими як Adobe Photoshop.

Цифрове зображення представлене в вигляді числової матриці, елементами котрої є значення інтенсивності кольору, а розмір матриці це розмір зображення. Наприклад, для чорно-білих зображень така матриця має нулі для білих пікселів та одиниці для чорних пікселів (рисунок 2.4).

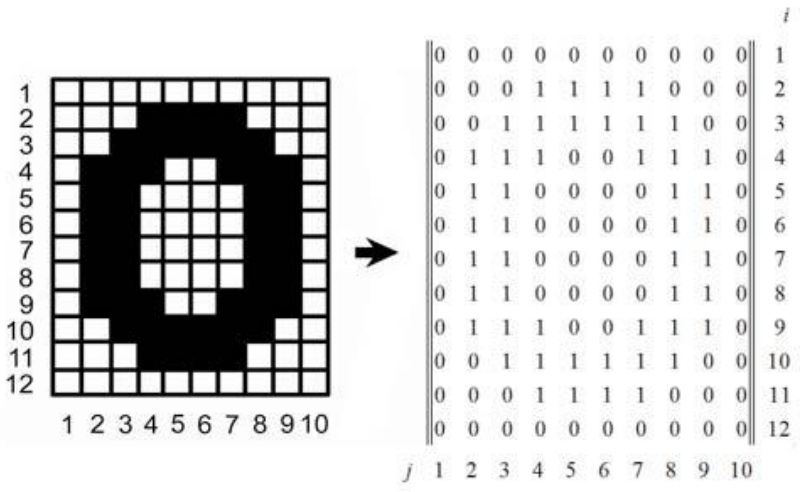

Рисунок 2.4 – Приклад представлення зображення

2.2 Методи клонування областей в цифрових зображеннях

На сьогоднішній день, будь-яка людина без аби-яких професійних навичок може зредагувати зображення і сфальсифікувати зображення способом «клонування». Дана можливість з’явилася завдяки різкому розвитку технічних і програмних технологій.

Існує декілька способів для клонування частини самого зображення та вставляння в іншу частину того ж самого цифрового зображення. Так в

графічному редакторі Adobe Photoshop для цих цілей можна використовувати такі інструменти як Штамп (Clone Stamp), Пензель відновлення (Healing Brush) і Латка (Patch).

Для більш вдалого клонування зловмисники також можуть використовувати Гумка (Erase) и Розмиття (Blur) для «замітання слідів» своєї фальсифікації.

Інструмент «Штамп» дозволяє скопіювати одну область зображення на іншу область зображення. Це надзвичайно простий у використанні інструментів програми. Фотографії складаються з крихітних пікселів, а «Штамп» дублює їх. По суті, інструмент «Штамп» замінює старі пікселі новими і робить ретушування невидимим.

Інструмент «Пензель відновлення» працює за наступним принципом – область покриття пензля змішується з периферійної областю об'єкта. Тобто, інструмент обчисливши колір і яскравість об'єкту, що видаляється, заливає його мікшованим кольором периферійної області.

Інструмент «Латка» вміє зміксовувати схожі за кольором пікселі об'єкта і периферії, що сприяє суттєвому полегшенню ретуші складних ділянок. Латка корисна в застосуванні, якщо ви хочете замінити будь-який фрагмент зображення цілком.

Інструмент «Гумка» змінює колір пікселів на фоновий або робить їх прозорими. Інструмент «Гумка» стирає пікселі шару при перетягуванні курсора, надаючи їм прозорість. Є можливість стерти лише фон зображення, зберігши края об'єкта переднього плану. Ставлячи різні параметри відбору зразків і допуску, можна управляти діапазоном значень прозорості і різкістю кордонів.

Інструмент «Розмиття» дозволяє розмити чіткі контури або області зображення, знизивши його чіткість. Розмиття фону з багатьма деталями допоможе поставити акцент на центральному фрагменті. Для цієї мети також можна скористатися фільтрами групи «Розмиття».

2.3 Алгоритм виявлення та локалізації клонування в цифрових зображеннях

Для представлення цифрового зображення розміром $n \times m$ пікселів в роботі

використовується: $n \times m$ – матриці R, G, B (колірна схема RGB); $n \times m$ – матриця яскравості Y (колірна схема YUV).

В процесі роботи алгоритму зображення розбивається на пересічні блоки. В алгоритмі, що модифікується розглядаються квадратні блоки, різного розміру, а саме 4x4, 8x8, 16x16 (рисунок 2.5).

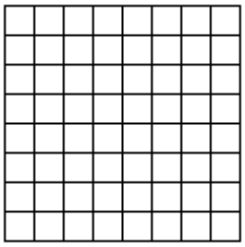

Рисунок 2.5 – Блоки квадратної форми розміру 8x8

В алгоритмі виявлення та локалізації областей клонування в цифрових зображеннях здійснюється пошук подібних блоків. В якості метрики подібності використовується коефіцієнт кореляції Пірсона. Він вказує силу зв'язку між досліджуваними об'єктами, наприклад, блоками D, C та вираховується за формулою:

$$correlation(D, C) = \frac{\sum_{rt}(d_{rt} - \overline{d})(c_{rt} - \overline{c})}{\sqrt{\sum_{rt}(d_{rt} - \overline{d})^2 \sum_{rt}(c_{rt} - \overline{c})^2}}, \quad (2.1)$$

де d_{rt}, c_{rt} – значення яскравостей пікселей в блоках D и C відповідно; $\overline{d}$, $\overline{c}$ – середнє значення яскравості в блоках D та C відповідно.

Для зменшення часу роботи алгоритму виявлення та локалізації областей клонування з використанням блоків квадратної форми пропонується використання маркерів для блоків розміру 4x4, 8x8, 16x16 (рисунок 2.6).

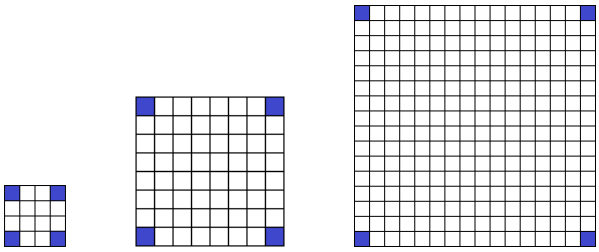

Рисунок 2.6 – Маркери блоку квадратної форми розміру 4x4, 8x8, 16x16

Під маркерами будемо розуміти декілька точок всередині блоку, які визначають вид блоку при розбитті.

Для блоків, розміром 4x4 в якості маркеру для цілого блока беруться лише 4 значення яскравості, що відповідають пікселям, які знаходяться в кутах блоку. Це значно підвищить ефективність роботи алгоритму виявлення областей клонування, але якщо брати блоки розміром 16x16, виявлена область клонування буде меншою, ніж є насправді. Оптимальним є використання маркерів для квадратних блоків розміром 8x8.

Перед тим як вирахувати коефіцієнт кореляції при порівнянні двох блоків, будемо порівнювати між собою маркери блоків. Якщо маркери у двох блоків співпадають, то припускаємо, що маємо справу зі схожими (клонованими та оригінальними) блоками та тільки тоді будемо вираховувати коефіцієнт кореляції для остаточного підтвердження цього припущення.

Тоді модифікований алгоритм для виявлення та локалізації областей клонування буде мати наступні шаги:

1. Ініціалізувати область, що треба виявити $res = \emptyset$ та розбити матрицю яскравості Y цифрового зображення на множину блоків, що перетинаються, розміром $p \times p$ пікселей $C = \{c_1, c_2, \dots, c_s\}$ таких, що:

$$\bigcup_{i=1}^{s} c_i = Y,$$

(тут кожний наступний блок c_i відрізняється від попереднього c_{i-1} зсувом на 1 піксель вправо, вліво, вниз та угору).

2. Кожний блок $c_i,\ i = 1, \dots, s$, розглянути в парі з усіма $c_j,\ j = i + 1, \dots, s$, відповідно. Для кожної пари:

2.1 Для кожної пари блоків c_i та c_j:

2.2 Отримати маркери mc_i та mc_j.

Якщо маркери mc_i не дорівнюють mc_j, то розглядати наступну пару блоків.

Інакше виконати наступні шаги:

2.2.1 Розрахувати коефіцієнт кореляції:

$$cor = correlation(c_i, c_j).$$

2.2.2 Якщо $cor = 1$, то блоки c_i и c_j являються оригінальним та клонованим,

$$res = res \cup c_i \cup c_j.$$

3. Вивести знайдену область res.

Отже, в даному розділі розглянуто алгоритм виявлення та локалізації областей клонування в цифрових зображеннях. Алгоритм має суттєвий недолік – обчислювальну складність. Для усунення цього недоліку були запропоновані маркери для блоків, які можна використовувати для виявлення областей фальсифікації та розроблено модифікацію алгоритму виявлення та локалізації областей клонування з використанням цих маркерів.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Середовище програмування

Visual Studio Community – безкоштовне повнофункціональне розширюване середовище IDE для створення сучасних програм Android, iOS та Windows, а також веб-додатків та хмарних служб.

Дане середовище має всі необхідні інструменти для комфортної та швидкої розробки додатків. Орієнтована на індивідуальних розробників.

Можливості та переваги Visual Studio Community [23]:

- створення та розробка додатків для всіх популярних платформ;
- наявність великого набору інструментів розробки (емулятори, налагоджувачі, симулятори);
- розширений редактор коду (з IntelliSense, автозавершення коду, підсвічуванням синтаксису, перевіркою коду);
- підтримує всі популярні мови програмування (C#, Visual Basic, F#, C++, HTML, JavaScript, TypeScript, Python);
- інтеграція з пакетом SDK для Azure (дозволяє завантажувати проекти в Azure та аналізувати їх продуктивність/ефективність за допомогою служби Application Insights);
- підтримка модулів, що забезпечує доступ до тисяч розширень у галереї Visual Studio;
- інтеграція з Git;
- розповсюджується на безоплатній основі;
- спрощена модульна установка.

Останнім часом C і C++ є найбільш використовуваними мовами для розробки комерційних і бізнес додатків. Ці мови влаштовують багатьох розробників, але насправді не забезпечують належної продуктивності розробки. Ці мови не орієнтовані на взаємодію з системами, що з'являються сьогодні, і дуже часто вони не відповідають існуючій практиці програмування для Web.

Враховуючи всі побажання розробників, Microsoft розробила нову мову –

C#. До нього входить багато корисних особливостей - простота, об'єктна орієнтованість, типова захищеність, "складання сміття", підтримка сумісності версій та багато іншого [24]. Дані можливості дозволяють швидко та легко розробляти програми, особливо COM+ програми та Web сервіси. При створенні C# його автори враховували досягнення багатьох інших мов програмування: C++, C, Java, SmallTalk, Delphi, Visual Basic тощо.

У C# було уніфіковано систему типів, тепер можна розглядати кожен тип як об'єкт. Незважаючи на те, чи використовується клас, структура, масив або вбудований тип, можна звертатися до нього як до об'єкта. Об'єкти зібрані в просторі імен (namespaces), які дозволяють програмно звертатися до будь-чого. Це означає, що замість списку файлів заголовків у своїй програмі необхідно написати простори імен, для доступу до об'єктів і класів усередині них.

Windows Forms – це технологія інтелектуальних клієнтів для NET Framework. Вона являє собою набір керованих бібліотек, які спрощують виконання стандартних завдань, таких як читання з файлової системи та запис до неї.

У Windows Forms форма – це візуальна поверхня, на якій відображається інформація для користувача. Зазвичай програма Windows Forms будується шляхом розміщення елементів керування на форму та написання коду для реагування на дії користувача, такі як клацання миші або натискання клавіш.

Елемент управління – це окремий елемент інтерфейсу користувача, призначений для відображення або введення даних.

При виконанні користувачем будь-якої дії з формою або одним з елементів управління створюється подія. Програма реагує на ці події за допомогою коду та обробляє події при їх виникненні.

Windows Forms включає широкий набір елементів керування, які можна додавати на форми: текстові поля, кнопки, списки, перемикачі і веб-сторінки.

До складу Windows Forms входять багатофункціональні елементи інтерфейсу користувача, що дозволяють відтворювати можливості таких складних програм, як Microsoft Office. Використовуючи елементи керування ToolStrip та

MenuStrip, можна створювати панелі інструментів та меню, що містять текст та малюнки, підменю та інші елементи керування, такі як текстові поля та поля зі списками.

Можна створити елементи керування FlowLayoutPanel, TableLayoutPanel і SplitContainer для створення складних макетів форм. Якщо потрібно створити свої власні елементи інтерфейсу користувача, простір імен System.Drawing містить широкий набір класів, необхідних для відтворення ліній, кіл та інших фігур безпосередньо на формі.

3.2 Інтерфейс програмного продукту

Для зручного користування модифікованим алгоритмом виявлення та локалізації областей клонування в цифрових зображеннях було розроблено простий та інтуїтивно зрозумілий інтерфейс для будь-якого користувача з мінімальним набором кнопок та з зрозумілим текстовим описом та виводом результату.

При запуску програмного продукту можна бачити головне вікно додатку, яке представлене на рисунку 3.1.

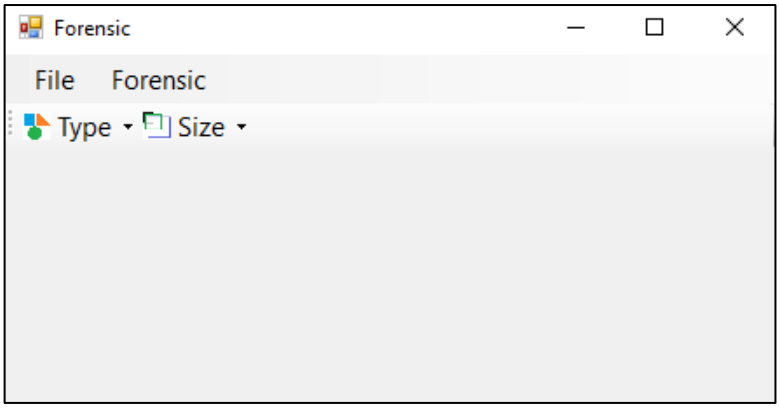

Рисунок 3.1 – Інтерфейс головного вікна розробленого програмного додатку

В меню головного вікна програми розташований пункт File, в якому, натиснувши на Open image, можна обрати зображення для обробки та вийти з програми, натиснувши Exit. Для завантаження фотографії відкриється вікно, в

якому можна обрати фотографію, яку необхідно обробити (рисунок 3.2).

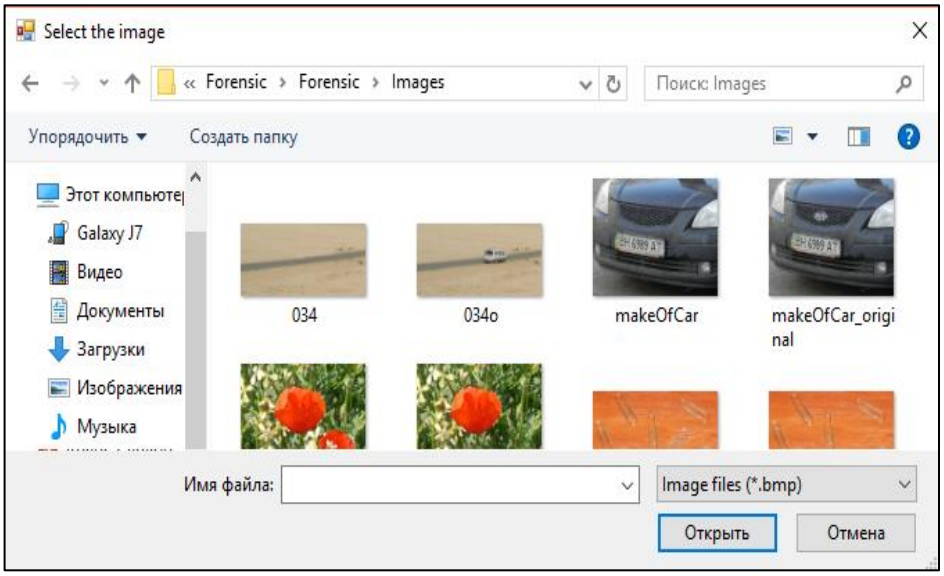

Рисунок 3.2 – Вікно вибору зображення для завантаження

Обравши фотографію переходимо до пункту Type (рисунок 3.3), в якому обирається тип блоків для роботи алгоритму виявлення області клонування. В роботі була реалізована робота з квадратними блоками (Square), інші типи блоків (Triangle та Circle) планується реалізувати в майбутньому.

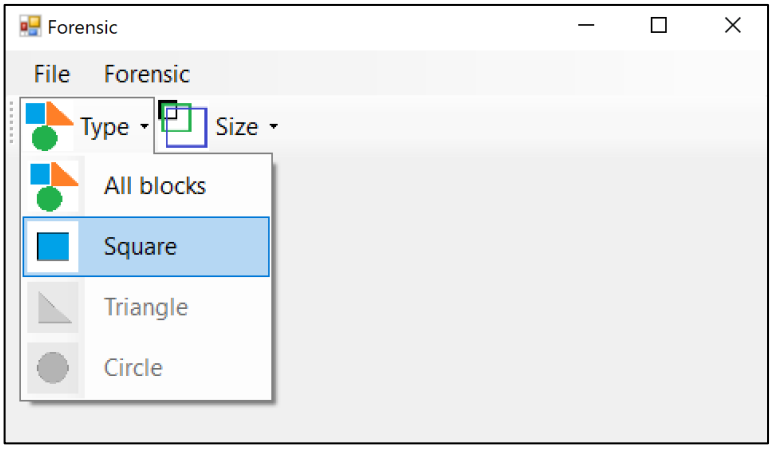

Рисунок 3.3 – Вибір типу блоків

В додатковому меню також є пункт Size, в якому можна обрати розмір блоків (рисунок 3.4).

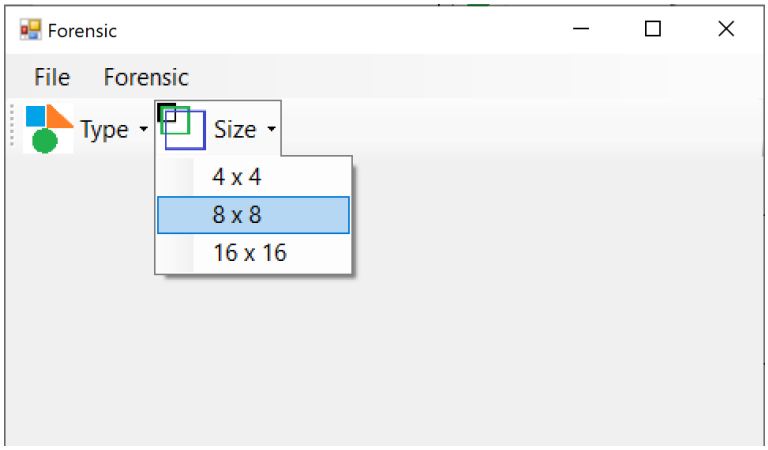

Рисунок 3.4 – Вибір розміру блоків

Обравши фотографію, тип блоків та розмір блоків в пункті Forensic меню можна обрати метод пошуку об’єкта клонування на зображенні: клонування Cloning чи клонування з маркерами Cloning with markers (рисунок 3.5).

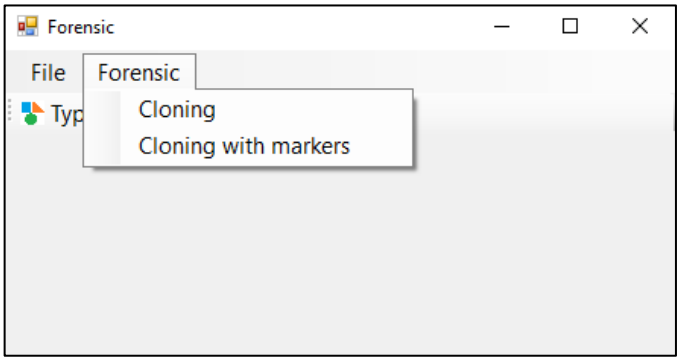

Рисунок 3.5 – Вибір методу пошуку області клонування

Щоб побачити за який час працює алгоритм пошуку області клонування Cloning спочатку оберемо, наприклад, тип Square та розмір блоків 8x8. З’явиться нове вікно (рисунок 3.6).

В цьому вікні бачимо інформаційне поле з параметрами, які ми обрали в попередньому головному вікні для сканування зображення. Натиснувши кнопку Start, починається процес обробки зображення та пошук в ньому області клонування, якщо така є. Процес загрузки можна побачити в progress bar, що показує скільки блоків оброблено, додаючи одне ділення з кожним обробленим

блоком.

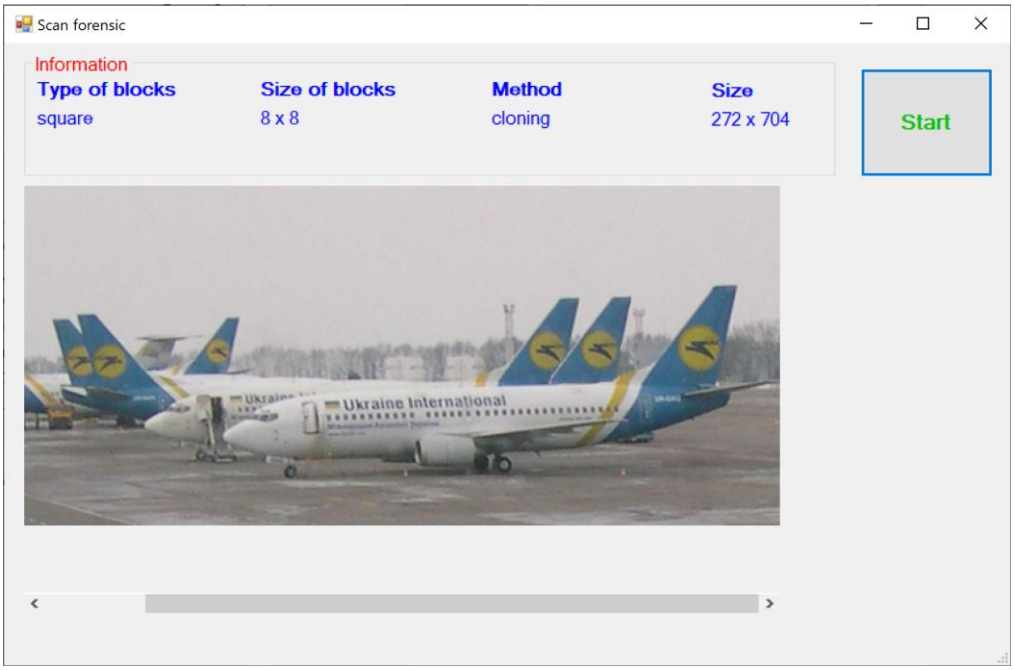

Рисунок 3.6 – Вікно роботи обраного методу

По завершенню обробки зображення в нашому вікні з’являється нове зображення (рисунок 3.7). Якби ми обрали оригінальну фотографію, то на зображенні з результатом (праворуч) побачимо чорний фон, який свідчить, що алгоритм не знайшов фальсифікації, а саме клонування. В нашому прикладу алгоритм знайшов області клонування (оригінальну та клоновану) та відобразив їх кольоровими областями.

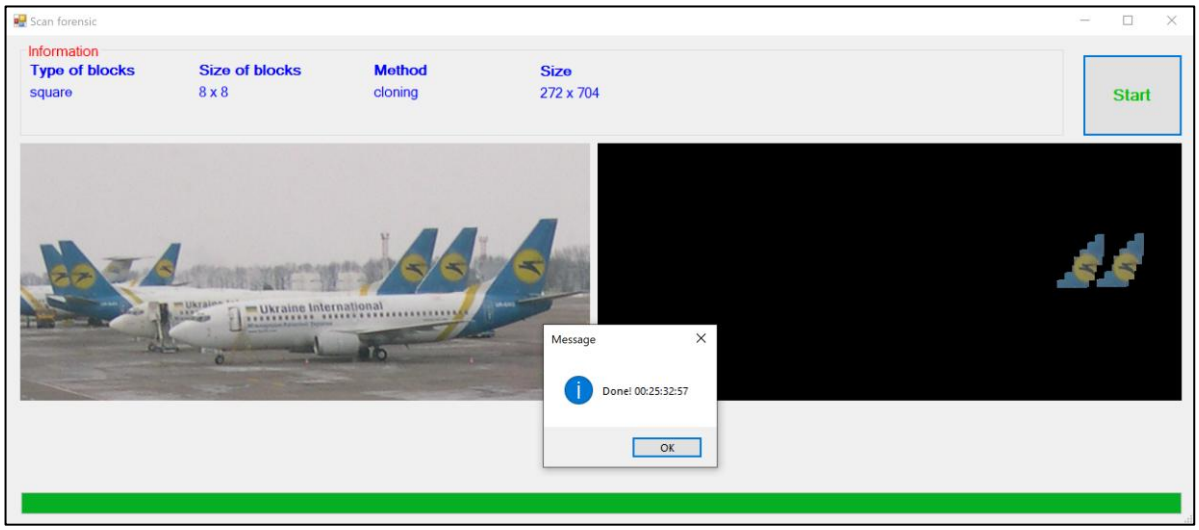

Рисунок 3.7 – Вікно з результатом роботи

Після роботи алгоритму виводиться вікно повідомлень, в якому зазначено скільки часу витрачено на пошук клонованої області. В нашому прикладі, коли пошук ведеться по блокам без використання маркерів, витрачено 25 хвилин 32 секунди. Якщо обрати теж саме зображення та запустити пошук по блокам з використання маркерів, то буде витрачено 2 хвилин 10 секунд. Знайдені клоновані області при запусках без використання маркерів та з використанням маркерів будуть однакові (рисунок 3.8).

а

б

в

Рисунок 3.8 – Результат роботи алгоритму: а – зображення, що обробляється; б – пошук по блокам без використання маркерів; в – пошук по блокам з використання маркерів

3.3 Результати експериментів

При дослідженні ефективності модифікованого алгоритму виявлення та локалізації областей клонування в цифрових зображеннях проведено обчислювальний експеримент з використанням ста зображень.

Оцінку ефективності розробленої модифікації виконано у вимірюванні часу виявлення та локалізації областей клонування до модифікації та після.

Нижче в таблиці 3.1 наведені дані експерименту деяких з цих зображень при використанні блоків 8x8.

Таблиця 3.1 – Дані експериментів при використанні блоків 8x8

№ зображення	Розмір зображення	Час роботи оригінального алгоритму (годин:хвилин:секунд)	Час роботи модифікованого алгоритму (годин:хвилин:секунд)
1	240 x 480	00:09:44	00:00:54
2	272 x 400	00:21:59	00:00:49
3	320 x 880	00:47:52	00:03:51
4	320 x 832	00:39:43	00:03:27
5	272 x 704	00:21:00	00:01:49
6	320 x 480	00:14:30	00:01:11
7	880 x 512	02:03:02	00:10:07
8	240 x 512	00:08:28	00:00:44
9	240 x 352	00:05:41	00:00:23
10	176 x 352	00:02:17	00:00:11

4 ОХОРОНА ПРАЦІ

Основною задачею кваліфікаційної роботи являється модифікація та програмна реалізація алгоритму виявлення та локалізації областей клонування в цифрових зображеннях. Подібні завдання виконуються за персональним комп'ютером (ПК). Отже в цьому розділі необхідно розглянути всі питання стосовно трудової діяльності користувача ПК, визначити заходи щодо запобігання виникненню ситуації, котра несе в собі шкоду для здоров'я, розробити рекомендацій щодо комфортних умов праці.

Приміщення, в якому працює оператор, має загальну площу 40 м^2, а висоту стелі 2,8 м. У приміщенні присутні 5 робочих місць, які включають в себе робочий стіл площею 1,5 м^2, стілець та персональний комп'ютер (ПК), який в свою чергу складається з монітора, системного блоку, клавіатури та миші. Одне з робочих місць обладнане спеціальним устаткуванням для проведення робот з паяльником. Варто відзначити, що площа одного робочого місця оператора ПК не повинна бути меншою за 6 м^2, а об'єм не менший за 20 м^3. Можна зробити висновок, що дане приміщення не задовольняє нормам тому потрібно підібрати приміщення більшої площі для комфортної роботи п'яти працівників.

Напруга живлення системного блоку становить 220 В. Потрібна потужність цього пристрою становить 450 Вт, тобто цей пристрій відноситься к класу пристроїв великої потужності.

В пристрої використовується джерело живлення, яке перетворює напругу 220 В у напругу величиною 300 В, необхідну для живлення блоку ключів. На виході блоку ключів отримуємо послідовність імпульсів, які подаються на первинну обмотку силового трансформатора. На вторинній обмотці силового трансформатора формується базові напруги +5 В та +12 В, котрі подаються на блок випрямлячів та стабілізаторів. На виході стабілізатора-випрямляча маємо готові стабілізовані напруги для живлення комп'ютерних вузлів.

Монітор є основним пристроєм виводу інформації. Споживча потужність монітора під час роботи 30 Вт., в режимі очікування 0,5 Вт. Клавіатура і «миша» – пристрої вводу інформації, вони підключаються за допомогою портів USB, та

споживають 2,5 Вт.

Засоби вводу розташовуються на поверхні робочого столу, клавіатура знаходиться по центру, для більшої зручності користувача можна регулювати її положення, встановлювати прямо чи під нахилом до працівника, «миша» розташована згідно основної робочої руки, праворуч або ліворуч від клавіатури.

Стосовно системного блоку, то він повинен бути встановлений в спеціальне відділення комп'ютерного столу, аби мінімізувати контакт користувача з його корпусом. В разі відсутності такого відділу варто розташувати системний блок внизу столу, щоб контакт з ним був якомога меншим.

Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів.

Висота робочої поверхні робочого столу має регулюватися в межах 680 – 800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600 – 1400 мм, глибина 800 – 1000 мм).

Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500 мм, завглибшки (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги – не менше ніж 650 мм.

Основною загрозою здоров'ю оператора ПК являється неправильно облаштоване робоче місце. Для її уникнення необхідно дотримуватись таких правил: робочий стілець має бути підйомне-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим. Регулювання за кожним із параметрів має здійснюватися незалежно, легко і надійно фіксуватися. Крок регулювання елементів стільця має становити: для лінійних розмірів – 15 – 20 мм, для кутових 2 – 5 градусів. Зусилля регулювання має не перевищувати 20 Н.

Для зниження статичного напруження м'язів верхніх кінцівок слід використовувати стаціонарні або змінні підлокітники завдовжки не менше ніж

250 мм, завширшки 50 – 70 мм, що регулюються за висотою над сидінням у межах 230 – 260 мм і відстанню між підлокітниками в межах 350 – 500 мм.

Поверхня сидіння і спинки стільця має бути напівм'якою з нековзним, повітронепроникним покриттям, що легко чиститься і не електризується.

Також у приміщенні, де знаходиться робоче місце оператора ПК, мають місце шуми механічного і аеродинамічного походження, широкосмугові із аперіодичним підсиленням при роботі принтерів, нижче вказані допустимі норми шуму (табл. 4.1).

Таблиця 4.1 – Допустимі норми шуму

Джерело шуму	Рівень шуму, дБА
Жорсткий диск	45
Вентилятор	45
Принтер	55
Сканер	50

Якщо рівень шуму більший ніж допустимі норми, то слід вдатися до низки заходів: облицьовування стін і стелі залу звукопоглинальними матеріалами, зниження шуму в джерелі, правильне планування устаткування і раціональна організація робочого місця оператора. Розраховане значення середнього рівня шуму не перевищує гранично допустимого рівня шуму для робочого місця оператора, тобто в спеціальних заходах зі зниження рівня шуму не має потреби.

Також в приміщенні, де розташоване робоче місце користувача ПК, необхідно дотримуватися певного мікроклімату (вентиляція, вологість повітря, опалення). На сьогоднішній день норми мікроклімату являються наступними (табл. 4.2):

Таблиця 4.2 – Норми мікроклімату

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22 – 24 °С 40 – 60 % до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23 – 25 °С 40 – 60 % 0,1 – 0,2 м/с

Для підтримки допустимих значень мікроклімату та концентрації позитивних та негативних іонів необхідно передбачати установки або прилади зволоження або штучної іонізації, кондиціювання повітря.

Одним із найважливіших факторів, котрі впливають на якість роботи оператора ПК, являється освітленість приміщення, згідно з сьогоднішніми стандартами приміщення, в яких встановлені персональні комп’ютери, повинні мати природне та штучне освітлення відповідно.

Природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природною освітленості (КПО) не нижче ніж 1,5%.

Штучне освітлення в приміщеннях з робочими місцями має здійснюватися системою загального рівномірного освітлення. У разі переважної роботи з документами, допускається застосування системи комбінованого освітлення (крім системи загального освітлення додатково встановлюються світильники місцевого освітлення). Зазначення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300 – 500 лк. Якщо ці значення освітленості неможливо забезпечити системою загального освітлення, допускається використовувати місцеве освітлення. При цьому світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати відблисків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк.

Як джерела світла в разі штучного освітлення мають застосовуватись переважно люмінесцентні лампи типу ЛБ. У разі влаштування відбитого освітлення у приміщеннях, де переважним чином працюють з документами, допускається застосування металогалогенних ламп потужністю 250 Вт. Допускається застосування ламп розжарювання у світильниках місцевого освітлення. Система загального освітлення має становити суцільні або переривчасті лінії світильників, розташовані збоку від робочих місць (переважно ліворуч), паралельно лінії зору працюючих.

Для загального освітлення слід застосовувати світильники серії ЛПО 3б із дзеркальними гратами, що укомплектовані високочастотними пускорегулювальними апаратами (ВЧ ПРА). Допускається застосовувати світильники цієї серії без ВЧ ПРА тільки в модифікації «Кососвітло».

Застосування світильників без розсіювачів та екрануючих грат заборонено. Яскравість світильників загального освітлення в зоні кутів випромінювання від 50 до 90 градусів з вертикаллю в повздовжній та поперечній площинах має становити не більше ніж 200 кд/м^2, захисний кут світильників – не менше ніж 40 градусів. Світильники місцевого освітлення повинні мати відбивач, що просвічує, із захисним кутом, не меншим ніж 40 градусів.

Слід передбачити обмеження прямої блискості від джерел природного та штучного освітлення. При цьому яскравість світлих поверхонь (вікна, джерела штучного освітлення), що розташовані в полі зору повинна бути не більше ніж 200 кд/м^2. Необхідно обмежувати відбиту блискість на робочих поверхнях відносно джерел природного і штучного освітлення. При цьому яскравість відблисків на екрані ВДТ має не перевищувати 40 кд/м^2, а яскравість стелі в разі застосування системи відбитого освітлення – 200 кд/м^2.

Показник осліпленості у разі використання джерел загального штучного освітлення у виробничих приміщеннях має не перевищувати 20, а показник дискомфорту в адміністративно-громадських приміщеннях має бути не більше за 40. Необхідно обмежувати нерівномірність розподілу яскравості в полі зору працюючих з ВДТ. При цьому співвідношення яскравостей робочих поверхонь

має бути не більшим ніж 3:1, а співвідношення яскравостей робочих поверхонь та поверхонь стін, обладнання тощо – 5:1.

Згідно ДСТУ Б В.1.1-36:2016 визначається та приймається ступінь вогнестійкості будинків та їх призначення. У приміщеннях з ПК найбільш вірогідні пожежі класів В, тобто горіння твердих речовин, що супроводжується тлінням або самозайманя електроустановок. Для співробітників дуже важливо виконання елементарних правил пожежної безпеки під час перебування на робочому місці, адже безвідповідальне ставлення може спричинити пожежу.

Необхідно дотримуватися таких правил пожежної безпеки:

– меблі та обладнання необхідно розміщувати таким чином, щоб забезпечувався вільний евакуаційний прохід до дверей виходу з приміщення. Евакуаційні шляхи та виходи необхідно постійно утримувати вільними, нічим не захаращувати;

– електромережі, електроприлади і апаратуру експлуатувати тільки у справному стані з урахуванням вказівок та рекомендацій підприємств-виготовлювачів. У разі виявлення пошкоджень електромереж, вимикачів, розеток та інших електровиробів слід негайно вимкнути їх та вжити необхідних заходів щодо приведення в пожежобезпечний стан;

– документи, папір та інші горючі матеріали слід зберігати на відстані не менше 1 м від електрощитів; 0,5 м від електросвітильників; 0,6 м від сповіщувачів автоматичної пожежної сигналізації та 0,15 м від приладів центрального водяного опалення.

– засоби протипожежного захисту слід утримувати у справному стані.

Також організація повинна встановити в приміщеннях спеціальні засоби:

– кнопки для сповідування о пожежі;

– пожежні крани;

– вогнегасники типу ОУ – 2 чи ОУ – 3.

Усі працівники повинні вміти користуватись вогнегасниками, іншими первинними засобами пожежогасіння, знати місце їх знаходження. Відстань від найбільш віддаленого місця приміщення до місця розташування вогнегасника не

повинна перевищувати 20 м. У випадку коли пожежу не вдалося погасити своїми силами, слід викликати службу пожежної безпеки подзвонивши по номеру «101». Для безпечної евакуації персоналу поруч з дверними отворами, вимикачами, рубильниками слід розміщувати фотолюмінісцентні евакуаційні знаки.

У разі короткого замикання в одному з блоків ПК або при падінні на підлогу системного блоку, принтера, необхідно відключити ПК від електромережі шляхом від'єднання штекер кабелю електроживлення блоку ПК від розетки, доповісти керівнику і викликати фахівця з ремонту комп'ютерів.

Виявлені фактори в першу чергу впливають на здоров'я людини. Тому ми повинні їх усунути, а якщо це неможливо, то зменшити їхню шкідливу дію.

Основною небезпекою є ураження струмом, для цього необхідно регулярно перевіряти справність ПК.

Також для уникнення можливих захворювань в результаті роботи необхідно дотримуватися правил облаштування робочого місця. Якщо працівник проводить забагато часу за ПК, це може викликати погіршення здоров'я, тому варто вести активний спосіб життя, зміцнювати тіло займаючись спортом, основну увагу приділяти спині, для уникнення розвитку сколіозу, і менше навантажувати очі, використовуючи спеціальні монітори та окуляри.

Важливим фактором являється дотримання правил пожежної безпеки та регулярні перевірки стану електрообладнання, наявності справних систем протипожежної безпеки, та засобів для усунення місцевого загорання. Не варто нехтувати і регулярними навчаннями (учбовими тривогами), для того щоб персонал знав, як необхідно поводитися в екстремальних ситуаціях. Це все допоможе зберегти життя працівникам, та уникнути небажаних впливів і травм.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра розглянуто та проаналізовано алгоритм виявлення та локалізації областей клонування в цифрових зображеннях.

Існуючі аналоги вирішення проблеми виявлення клонування мають ряд недоліків. Проте загальним недоліком є обчислювальна складність усіх методів та надто високі часові витрати в роботі їх програмних реалізацій.

За результатами аналізу алгоритму виявлення та локалізації областей клонування в цифрових зображеннях був проведений вибір маркерів для квадратних блоків, які можуть використовуватися для виявлення областей фальсифікації.

Розроблено та програмно реалізовано модифікацію алгоритму виявлення та локалізації областей клонування в цифрових зображеннях з використанням маркерів в інтегрованому середовищі розробки Visual Studio Community на мові C#. Розроблена модифікація дозволила зменшити час роботи алгоритму виявлення та локалізації областей клонування в цифрових зображеннях, що значно спрощує процедуру аналізу цифрових зображень на наявність порушень їх цілісності.

Предметом подальшого розвитку даної роботи має бути вдосконалення модифікованого алгоритму виявлення клонування шляхом розробки системи маркерів для автоматичного вибору виду блоку та його розміру.

ПЕРЕЛІК ПОСИЛАНЬ

1. Что такое фотомонтаж? URL: https://puskart.com/articles/chto-takoe-fotomontazh

2. Хорошко В.А. Чекатков А.А. Методы и средства защиты информации К.: ЮНИОР, 2003. 505 с.

3. Нариманова Е.В. Проверка целостности цифрового сигнала. Донецк: Цифровая типография, 2011. 180 с.

4. Грибунин В.Г. Оков И.Н., Туринцев И.В. Цифровая стеганография. М.: СОЛОН-Пресс, 2002. 272 с.

5. Fridrich J. Methods for Tamper Detection in Digital Images. *Proceedings of the Multimedia and Security Workshop* 1999. P.19—23.

6. Fridrich J. Goljan M.Protection of digital images using self-embedding *Proceedings of Symposium on Content Security and Data Hiding in Digital Media*. 1999. P. 92—96.

7. Fridrich J., Goljan M. Images with self-correcting capabilities. *Proceedings of IEEE International Conference on Image Processing*. 1999. Vol.3. P. 792—796.

8. Yeung M.M. Mintzer F. An Invisible Watermarking Technique for Image Verification. *Proceedings of IEEE International Conference on Image Processing*. 1997. Vol.2. P. 680—683.

9. Wolfgang R.B. Delp E. J. A Watermark for Digital Images. *Proceedings of IEEE International Conference on Image Processing*. 1996. Vol.3. P. 219—222.

10. Dybala B., Jennings B., Letscher D. Detecting filtered cloning in digital images. *ACM Multimedia and Security Workshop.* 2007. P. 43–50.

11. Langille A., Gong M. An efficient match-based duplication detection algorithm. *Canadian Conference on Computer and Robot Vision.* 2006. P.64.

12. Luo W. Huang J., Qiu G. Robust detection of region duplication forgery in digital images. *International Conference on Pattern Recognition*. 2006. P.746–749.

13. Pan X., Lyu S. Region duplication detection using image feature matching. *IEEE Transactions on Information Forensics and Security*. 2010. Vol.5(4). P.857-867.

14. Wang J. Liu G., Li H., Dai Y.,. Wang Z Detection of image region duplication forgery using model with circle block. *International Conference on Multimedia Information Networking and Security*. 2009. P. 25–29.

15. Jing L. Shao Ch. Image Copy-Move Forgery Detecting Based on Local Invariant Feature . *Journal of Multimedia*. 2012. Vol 7, No 1. P.90-97.

16. Shivakumar B.L., Baboo S.S. Detection of Region Duplication Forgery in Digital Images Using SURF. *IJCSI International Journal of Computer Science Issues.* 2011. Vol.8, Is. 4, No 1. P.199-205.

17. Bay H., Tuytelaars T., Van Gool L. J. SURF: Speeded Up Robust Features *ECCV*. 2006. Vol.1. P. 404-417.

18. Lowe, D.G. Distinctive image features from scale-invariant keypoints. URL: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

19. Зорило, В.В. Выявление клонирования как фальсификации цифрового изображения. *Вісник Національного технічного університету «ХПІ». Збірник наукових праць. Тематичний випуск «Системний аналіз, управління та інформаційні технології»*. 2011. № 35. С. 31-38.

20. Лебедева, Е.Ю. Обнаружение клонированных участков изображений в задачах выявления фальсификации. *XII міжнародна науково–практична конференція «Современные информационные и электронные технологии»*. 2011. С. 175.

21. Кобозева А.А., Лебедева Е.Ю. Основы метода выявления клонированных участков изображения, подвергнутых коррекции яркости. *Сучасна спеціальна техніка*. 2013.№3. С.17 – 24.

22. Прэтт У. Цифровая обработка изображений. Москва: Мир, 1982. 280 с.

23. Visual Studio Community. URL: https://programy.com.ua/ru/visual-studio-community.

24. Особенности и преимущества языка C#. URL: https://studbooks.net/2070063/informatika/osobennosti_preimuschestva_yazyka.

Додаток А. Лістинг програмного продукту

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;

namespace Forensic
{
    public partial class Forensic : Form
    {
        private string fileName = "";
        private string typeOfBlocks = "";
        private int sizeOfBlocks = 0;
        private string method = "";

        private PictureBox pb;
        private Bitmap forensicImage;

        private ImageReader blk1, blk2;
        private ImageReader blk11, blk12, blk13, blk14;
        private ImageReader blk21, blk22, blk23, blk24;

        public Forensic(string file, string type, int size,
string how)
        {
            InitializeComponent();

            fileName = file;
```

```
                typeOfBlocks = type;
                label2.Text = typeOfBlocks;
                sizeOfBlocks = size;
                label4.Text = "" + sizeOfBlocks + " x " +
sizeOfBlocks;

                method = how;
                label6.Text = method;
                forensicImage = new Bitmap(fileName);
                panel1.AutoScroll = true;
                pb = new PictureBox();
                pb.Image = forensicImage;
                pb.Width = forensicImage.Width;
                pb.Height = forensicImage.Height;
                pb.SizeMode = PictureBoxSizeMode.Zoom;
                label8.Text = ""+ forensicImage.Height+" x "+
forensicImage.Width;
                panel1.Controls.Add(pb);
            }

            private void button1_Click(object sender, EventArgs e)
            {
                Stopwatch watch = new Stopwatch();
                watch.Start();
                if (method == "cloning")
                    ScanForensics(forensicImage,       typeOfBlocks,
sizeOfBlocks);
                else if (method == "cloning with markers")
                    ScanForensicsMarkers(forensicImage,
typeOfBlocks, sizeOfBlocks);

                watch.Stop();
                TimeSpan ts = watch.Elapsed;
                string                elapsedTime                =
String.Format("{0:00}:{1:00}:{2:00}:{3:00}", ts.Hours, ts.Minutes,
```

```
ts.Seconds, ts.Milliseconds / 10);
                MessageBox.Show("Done! " + elapsedTime, "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            private void InitializeImages(Bitmap forensic, out
ImageReader img, out ImageReader imgNew)
            {
                img = new ImageReader(forensic);
                img.CreateArrayYUV();
                ColorModels.RGBToYUV(img.R, img.G, img.B, img.Y,
img.U, img.V, img.Width, img.Height);

                Bitmap bmp = new Bitmap(img.Width, img.Height);
                using (var g = Graphics.FromImage(bmp))
                    g.Clear(Color.Black);
                imgNew = new ImageReader(bmp);
            }

            private void CreateBlock(int blk_sizeW, int blk_sizeH,
out ImageReader blk)
            {
                Bitmap bmpBlock = new Bitmap(blk_sizeW, blk_sizeH);
                using (var g = Graphics.FromImage(bmpBlock))
                    g.Clear(Color.Black);

                blk = new ImageReader(bmpBlock);
                blk.CreateArrayYUV();
            }

            private void CreateBlock(ref ImageReader blk, int
blk_sizeW, int blk_sizeH)
            {
                Bitmap bmpBlock = new Bitmap(blk_sizeW, blk_sizeH);
                using (var g = Graphics.FromImage(bmpBlock))
```

```
                g.Clear(Color.Black);

                blk = new ImageReader(bmpBlock);
                blk.CreateArrayYUV();
            }

            private void ScanForensics(Bitmap forensicImage, string
type, int blk_size)
            {
                ImageReader img, imgNew;
                InitializeImages(forensicImage,   out   img,   out
imgNew);

                if (type == "square")
                {
                    CreateBlock(blk_size, blk_size, out blk1);
                    CreateBlock(blk_size, blk_size, out blk2);
                }

                progressBar1.Maximum  =  (img.Width/  blk_size)  *
(img.Height/ blk_size);
                progressBar1.Value = 0;

                for (int ii = 0; ii < img.Height; ii += blk_size)
                    for  (int  jj  =  0;  jj  <  img.Width;  jj  +=
blk_size)
                    {
                        if  (ii  +  blk_size  >  img.Height  ||  jj  +
blk_size > img.Width)
                            continue;

                        if (type == "square")
                            Blocks.GetBlockFromImage(img.Y,  blk1.Y,
img.Width, img.Height, jj, ii, blk_size, blk_size);
```

```
                        for (int i = 0; i < img.Height - blk_size;
i++)
                            for (int j = 0; j < img.Width -
blk_size; j++)
                            {
                                if (ii == i && jj == j)
                                    continue;

                                if (type == "square")
                                {
                                    Blocks.GetBlockFromImage(img.Y,
blk2.Y, img.Width, img.Height, j, i, blk_size, blk_size);
                                    CompareBlocks(img,       imgNew,
blk1, blk2, i, j, ii, jj, blk_size, blk_size, type, -1);
                                }
                            }
                        progressBar1.Value ++;
                    }

                Bitmap    result    =    new    Bitmap(imgNew.Width,
imgNew.Height);
                for (int i = 0; i < imgNew.Height; i++)
                    for (int j = 0; j < imgNew.Width; j++)
                        result.SetPixel(j,                          i,
Color.FromArgb(imgNew.R[i   *   imgNew.Width   +   j],   imgNew.G[i   *
imgNew.Width + j], imgNew.B[i * imgNew.Width + j]));

                Image                   image_new                   =
Image.FromHbitmap(result.GetHbitmap());
                ShowImage(image_new);

                String   fileNameNew   =   GetNewFileName(fileName,
type.Substring(0, 1), blk_size);
                image_new.Save(fileNameNew);
```

```
                if (type == "square")
                {
                    if (blk2 != null) blk2.Dispose();
                    if (blk1 != null) blk1.Dispose();
                }

                if (imgNew != null)
                imgNew.Dispose();

                if (img != null)
                    img.Dispose();
            }

            private void ScanForensicsMarkers(Bitmap forensicImage,
string type, int blk_size)
            {
                ImageReader img, imgNew;
                InitializeImages(forensicImage,   out   img,   out
imgNew);

                if (type == "square")
                {
                    CreateBlock(blk_size, blk_size, out blk1);
                    CreateBlock(blk_size, blk_size, out blk2);
                }

                progressBar1.Maximum  =  (img.Width  /  blk_size)  *
(img.Height / blk_size);
                progressBar1.Value = 0;

                for (int ii = 0; ii < img.Height; ii += blk_size)
                    for  (int  jj  =  0;  jj  <  img.Width;  jj  +=
blk_size)
                    {
                        if  (ii  +  blk_size  >  img.Height  ||  jj  +
```

```
blk_size > img.Width)
                                continue;

                            if (type == "square")
                                Blocks.GetBlockFromImage(img.Y,  blk1.Y,
img.Width, img.Height, jj, ii, blk_size, blk_size);
                            for (int i = 0; i < img.Height - blk_size;
i++)
                                for  (int  j  =  0;  j  <  img.Width  -
blk_size; j++)
                                {
                                    if (ii == i && jj == j)
                                        continue;

                                    if (type == "square")
                                    {
                                        bool           found           =
VerificationBlocks(img, type, jj, ii, j, i, blk_size, blk_size);
                                        if (found)
                                        {

Blocks.GetBlockFromImage(img.Y, blk2.Y, img.Width, img.Height, j, i,
blk_size, blk_size);
                                            CompareBlocks(img,   imgNew,
blk1, blk2, i, j, ii, jj, blk_size, blk_size, type, -1);
                                        }
                                    }
                                }
                            progressBar1.Value++;
                        }

                    Bitmap    result    =    new    Bitmap(imgNew.Width,
imgNew.Height);
                    for (int i = 0; i < imgNew.Height; i++)
                        for (int j = 0; j < imgNew.Width; j++)
```

```
                        result.SetPixel(j, i,
Color.FromArgb(imgNew.R[i * imgNew.Width + j], imgNew.G[i *
imgNew.Width + j], imgNew.B[i * imgNew.Width + j]));

                Image image_new =
Image.FromHbitmap(result.GetHbitmap());
                ShowImage(image_new);

                String fileNameNew = GetNewFileName(fileName,
type.Substring(0, 1), blk_size);
                image_new.Save(fileNameNew);

                if (type == "square")
                {
                    if (blk2 != null) blk2.Dispose();
                    if (blk1 != null) blk1.Dispose();
                }

                if (imgNew != null)
                    imgNew.Dispose();

                if (img != null)
                    img.Dispose();
            }

            private bool VerificationBlocks(ImageReader img, string
type, int jj, int ii, int j, int i, int blk_sizeX, int blk_sizeY)
            {
                if (type == "square")
                {
                    byte point11 = img.Y[ii * img.Width + jj];
                    byte point12 = img.Y[i * img.Width + j];
                    byte point21 = img.Y[ii * img.Width + jj +
blk_sizeX-1];
                    byte point22 = img.Y[i * img.Width + j +
```

```
blk_sizeX-1];
                    byte  point31  =  img.Y[(ii  +  blk_sizeY-1)  *
img.Width + jj];
                    byte  point32  =  img.Y[(i  +  blk_sizeY-1)  *
img.Width + j];
                    byte  point41  =  img.Y[(ii  +  blk_sizeY-1)  *
img.Width + jj + blk_sizeX-1];
                    byte  point42  =  img.Y[(i  +  blk_sizeY-1)  *
img.Width + j + blk_sizeX-1];

                    if (point11 == point12 && point21 == point22 &&
point31 == point32 && point41 == point42)
                        return true;
                    else
                        return false;
                }
                return false;
            }

            private void CompareBlocks(ImageReader img, ImageReader
imgNew, ImageReader blk1, ImageReader blk2, int i, int j, int ii,
int jj, int blk_sizeX, int blk_sizeY, string type, int SubBlockNum)
            {
                ImageReader blk = null;
                double cor = Statistics.Corralation(blk1.Y, blk2.Y,
blk_sizeX, blk_sizeY);
                if (cor == 1)
                {
                    CreateBlock(blk_sizeX, blk_sizeY, out blk);

                    if (type == "square")
                    {
                        Blocks.GetBlockFromImage(img.R,       blk.R,
img.Width, img.Height, j, i, blk_sizeX, blk_sizeY);
                        Blocks.GetBlockFromImage(img.G,       blk.G,
```

```
img.Width, img.Height, j, i, blk_sizeX, blk_sizeY);
                        Blocks.GetBlockFromImage(img.B,       blk.B,
img.Width, img.Height, j, i, blk_sizeX, blk_sizeY);

                        Blocks.PutBlockToImage(blk.R,      imgNew.R,
img.Width, img.Height, jj, ii, blk_sizeX, blk_sizeY);
                        Blocks.PutBlockToImage(blk.R,      imgNew.R,
img.Width, img.Height, j, i, blk_sizeX, blk_sizeY);

                        Blocks.PutBlockToImage(blk.G,      imgNew.G,
img.Width, img.Height, jj, ii, blk_sizeX, blk_sizeY);
                        Blocks.PutBlockToImage(blk.G,      imgNew.G,
img.Width, img.Height, j, i, blk_sizeX, blk_sizeY);

                        Blocks.PutBlockToImage(blk.B,      imgNew.B,
img.Width, img.Height, jj, ii, blk_sizeX, blk_sizeY);
                        Blocks.PutBlockToImage(blk.B,      imgNew.B,
img.Width, img.Height, j, i, blk_sizeX, blk_sizeY);
                    }
                }

                if (blk != null)
                    blk.Dispose();
            }

            private  string  GetNewFileName(string  fileName,  string
type, int size)
            {
                String temp = fileName;
                String new_str = "";
                int len = temp.Length;
                if (method == "cloning with markers")
                    new_str = temp.Substring(0, len - 4) + "_cor_"
+ type + size + "_mark" + temp.Substring(len - 4);
                else
```

```
                    new_str = temp.Substring(0, len - 4) + "_cor_"
+ type + size + temp.Substring(len - 4);

                return new_str;
            }

            private void ShowImage(Image image)
            {
                panel2.AutoScroll = true;

                pb = new PictureBox();
                pb.Image = image;
                pb.Width = image.Width;
                pb.Height = image.Height;
                pb.SizeMode = PictureBoxSizeMode.Zoom;

                panel2.Controls.Add(pb);
            }
        }
    }
```