

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики і радіоелектроніки
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра системного аналізу та обчислювальної математики
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра
(ступінь вищої освіти)

на тему Обробка зображень методами машинного навчання

Виконав: студент(ка) 4 курсу, групи
КНТ 827-СП

Спеціальності 124 – Системний аналіз
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Інтелектуальні технології та прийняття
рішень в складних системах»

Яценко М.В.
(прізвище та ініціали)

Керівник

Рябенко А.Є.
(прізвище та ініціали)

Рецензент

В.Р. Савченко С.В.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут інформатики і радіоелектроніки
Факультет комп'ютерних наук і технологій
Кафедра системного аналізу та обчислювальної математики
Ступінь вищої освіти бакалавр
Спеціальність 124 - Системний аналіз
(код і найменування)
Освітня програма (спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри Г.В. Корніч
« » 2020 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Яценко Марія Володимирівна
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Обробка зображень методами машинного навчання
керівник проєкту (роботи) Рябенко Антон Євгенович, к. т. н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «19» травня 2020 року № 136

2. Строк подання студентом проєкту (роботи) 04 червня 2020 року

3. Вихідні дані до проєкту (роботи) дані, що стосуються машинного навчання для обробки зображень, перелік сучасних мов програмування та метод існуючих рішень.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області та постановка завдання роботи, проектування та розробка програмного додатку, керівництво програмою, керівництво аналізом

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконання завдання
Контроль	Мирошара Д.В. ст. інж.	Д.В.	Д.В.
Відділ інженерії	Рибенко А.Е., доцент	А.Е.	А.Е.
проектів			

7. Дата видачі завдання « 15 » листопада 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Прим.
1	Сформулювати мету та оч. завд. дип. роботи	15.10.2019 - 29.10.2019	
2	Оформлювати літературу та науков. видання	30.10.2019 - 30.11.2019	
3	Розробка графічного інтерфейсу	02.12.2019 - 22.12.19	
4	Розробка програмного додатку	23.12.2019 - 16.01.2020	
5	Тестування та вивч. щодо дип. роботи	07.01.2020 - 15.05.2020	
6	Попередній захист дип. роботи та оф. реценз.	26.05.2020 - 05.06.2020	
7	Захист дипломної роботи	05.06.2020	

Студент(ка)

(підпис)

Керівник проєкту (роботи)

(підпис)

Яценко М.В.
(прізвище та ініціали)

Рябенко А.Є.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи бакалавра: 62 с., 43 рис., 3 табл., 3 додатка, 9 джерел.

Мета дипломної роботи – розробка програми для обробки зображень алгоритмами машинного навчання, а саме – автоматизованого накладання фільтрів на вхідне зображення.

Методи проектування – мова CSharp, інтерфейс програмування додатків Windows Forms, середовище Microsoft Visual Studio 2019.

У результаті створена програма, яка є простою у використанні, крос-платформна, з можливістю редагування конфігурації, яка дає змогу швидко накладати фільтр на зображення, наглядно демонструє використання машинного навчання для обробки зображень.

МАШИННЕ НАВЧАННЯ, ЯДРО СГОРТКИ, ХУДОЖНІЙ СТИЛЬ, СТИЛІЗАЦІЯ, ФІЛЬТРИ ОБРОБКИ, ЗОБРАЖЕННЯ, C# (C SHARP), VISUAL STUDIO, WINDOWS FORMS.

ЗМІСТ

Завдання.....	2
Реферат	4
Вступ.....	7
1 Аналіз предметної області та постановка завдань роботи.....	9
1.1 Опис предметної області	9
1.2 Організація фільтрів.....	9
1.2 Огляд схеми VGG19.....	15
1.3 Опис алгоритмів	19
1.4 Огляд існуючих програмних засобів що вирішують аналогічні завдання ...	21
1.4.1 Instagram.....	21
1.4.2 Samsung and Apple Camera Apps.....	22
1.4.3 Adobe Photoshop	22
1.5 Вибір архітектури розроблюваної системи	24
2 Опис програми.....	26
2.1 Структура та функціонал розроблюваної програми.....	26
2.2 Аналіз програмних засобів.....	26
2.2.1 Обґрунтування використання мови програмування Csharp	26
2.2.2 Обґрунтування використання Windows Forms	28
3 Основні рішення щодо реалізації компонентів системи.....	29
3.1.1 Макет графічного інтерфейсу користувача.....	29
3.3 Особливості реалізації системи	31
4 Керівництво програміста.....	32
4.1 Призначення й умови застосування програми.....	32
4.2 Характеристики програми.....	33
4.3 Підготовка програми до запуску	33
4.4 Вхідні і вихідні дані	34

5 Керівництво оператора	35
5.1 Призначення програми	35
5.2 Умови виконання програми	35
5.3 Встановлення програми.....	35
5.4 Видалення програми	36
5.5 Виконання програми.....	36
Висновки	40
Перелік посилань.....	41
Додаток А 42Специфікація вимог	42
А.1 Основні вимоги до програми що розроблюється.....	42
А.1.1 Функціональні вимоги	42
А.1.2 Вимоги до надійності.....	42
А.1.3 Умови експлуатації	42
А.1.4 Вимоги до складу та параметрів технічних засобів	43
А.1.5 Вимоги до маркування та пакування	43
А.1.6 Вимоги до транспортування та зберігання	44
А.1.7 Вимоги до програмної документації.....	44
Додаток Б 45Текст програми	45
Б.1 Текст програми модулю ml_core	45
Б.1.1 Файл Program.....	45
Б.1.2 Файл AssemblyInfo.cs	45
Б.1.3 Файл Resources.Designer.cs	46
Б.1.4 Файл App.config	47
Б.1.5 Файл BrightnessContrast.cs	47
Б.1.6 Файл ColorBalance.cs	48
Б.1.7 Файл MatricaSvertki.cs	50
Б.1.8 Файл Program.vshost.exe.manifest.....	52
Додаток В Слайди презентації.....	53

ВСТУП

В наш час обробка медіа даних, будь то статичні зображення, відео чи навіть онлайн відео трансляції у реальному часі, є дуже актуальною темою. Обсяги медіа ростуть, необхідність у їхній обробці також. Багато хто з нас використовує засоби обробки зображень та відео у своєму повсякденному житті, ведучи блог у Instagram, Twitter, Facebook чи навіть YouTube, де простота, швидкість та автоматичність процесу обробки медіа робить його зручним та легким, але обмеженим.

На допомогу прийшло спеціалізоване професійне програмне забезпечення, таке як Adobe Photoshop, Sony Vegas Pro та інше ПЗ для обробки відео та зображень. Але використання цих програм потребує певних навичок та має під собою ручну обробку та редагування, що підвищує складність використання та витрачає багато часу.

Засобом автоматизації певних рутинних ручних дій користувача для обробки зображень можуть виступати алгоритми машинного навчання та лінійні алгоритми. За їх допомогою можна автоматизувати такі процеси обробки зображень як комплексне покращення якості, корекція кольору, контрастність, колірний баланс картинки, а також підвищувати різкість і розмивати зображення та багато іншого лише в декілька простих кроків, без необхідності ручного редагування зображень.

Розроблене програмне забезпечення має автоматизувати певний процес обробки зображення, а саме – прості фільтрації растрових зображень, що дозволяє змінювати яскравість, контрастність, колірний баланс картинки, а також підвищувати різкість і розмивати зображення, в декілька кроків, без участі користувача. Таке ПЗ дає змогу перетворити звичайне фото у професійне. Треба лише обрати стиль та натиснути одну кнопку. Таке трансформування зображення може використовуватися у сфері художньої обробки фото, а

особливо – у сфері розваг. Наприклад, редагування фото в історіях чи публікаціях користувачів Instagram у режимі реального часу лише одним рухом.

Також, розроблене ПЗ має бути кросплатформовим та незалежним від інтерфейсної частини, представляти собою ядро обробки медіа даних, абстрагуючись від конкретної реалізації графічного інтерфейсу користувача.

Виходячи із вимог, розроблений трансформатор стилю представляє собою програмне забезпечення, розроблене на CSharp.

Для простоти демонстрації прикладу реалізації трансформатору стилю на практиці в якості фінального зразка рішення з GUI було використано додаток Windows Forms.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ РОБОТИ

1.1 Опис предметної області

У даній роботі аналізуються проблеми, пов'язані із застосуванням алгоритмів машинного навчання для обробки цифрових зображень і ефективних методів їх вирішення; досліджуються сфери застосування алгоритмів машинного навчання на практиці, їх оптимізація та комерційний успіх.

Цифрове зображення – масив даних, отриманий шляхом дискретизації (аналого-цифрового перетворення) оригіналу. Після кодування за допомогою певного алгоритму і запису на носій, цей масив даних стає файлом [1].

Справжня цінність комп'ютерної обробки зображень залежить від того, що можна зробити з зображенням, як тільки воно виявилось в комп'ютері. Існує безліч корисних маніпуляцій з фотографіями, введеними цифровим способом. Якщо знімок зроблений з перетримуванням можна скоротити витримку, зменшивши колірні значення пікселів. При необхідності, червону, зелену і синю компоненти можна змінювати окремо, щоб отримати найкращий колірний баланс. У розпливчастих зображеннях можна збільшити різкість, і, навпаки, чіткі, контрастні зображення можна розмити, імітуючи ефект пом'якшуючих фотофільтри.

1.2 Організація фільтрів

Математична морфологія – це теорія і техніка аналізу і обробки геометричних структур, заснована на теорії множин, топології і випадкових функціях, в основному застосовується в обробці цифрових зображень.

- Фільтр «Яскравість / контрастність»

При розмиванні перерозподіляються кольору в зображенні і пом'якшуються різкі кордону, в той час як при збільшенні різкості підкреслюються відмінності між квітами суміжних пікселів і виділяються непомітні деталі.

Ядро розмивання, складається із сукупності коефіцієнтів, кожен з яких менше 1, а їх сума становить 1. Це означає, що кожен піксель поглине щось з квітів сусідів, але повна яскравість зображення залишиться незмінною. (Якщо сума коефіцієнтів більше ніж 1, яскравість збільшиться; якщо менше ніж 1, яскравість зменшиться.) У ядрі різкості центральний коефіцієнт більше 1, а оточений він негативними числами, сума яких на одиницю менше центрального коефіцієнта. Таким чином збільшується будь-який існуючий контраст між кольором пікселя і квітами його сусідів[2].

Для зміни яскравості на N відсотків використовується наступна формула:

$$I = I + N \cdot 128 / 100 \quad (1.1)$$

де I - відповідно R, G, B канали кожної точки зображення.

Зменшення контрастності на N відсотків була обрана формула:

$$I = (I \cdot (100 - N) + 128 \cdot N) / 100 \quad (1.2)$$

Для збільшення контрастності на N відсотків була обрана формула:

$$I = (I \cdot 100 - 128 \cdot N) / (100 - N) \quad (1.3)$$

Якщо нове I не влучає у діапазон 0..255 - то його слід урізати.

- Фільтр «Баланс кольору»

У фотографії та обробки зображень, колірний баланс є глобальна регулювання інтенсивності кольорів (як правило, червоного, зеленого і синього основних кольорів). Важлива метою даної коригування є надання певних кольорів - особливо нейтральних кольорів - правильно. Отже, загальний метод іноді називають баланс сірого, нейтральний баланс або баланс білого. Колірний баланс змінює загальну суміш кольорів в зображенні і використовується для корекції кольору. Узагальнені варіанти колірного балансу використовуються для корекції інших, ніж нейтральні кольори або навмисно змінити їх ефект [2].

Дані зображення, отримані за допомогою датчиків - або плівки або електронні датчики зображення - повинні бути перетворені з отриманих значень до нових значень, які підходять для відтворення кольору або дисплея. Деякі аспекти процесу збору і відображень роблять корекцію такого кольору істотно - в тому числі, що датчики збору не збігаються з датчиків в людському оці, що властивості середовища відображення повинні бути враховані, і що умови навколишнього середовища обзоа $\sim d$ придбання відрізняються від умов перегляду дисплея.

Операції колірного балансу в популярних редагування зображень додатків зазвичай працюють безпосередньо на червоний, зелений і синій канал піксельних значень, без відносно будь-якого кольору зондування або відтворення моделі.

Для зміни колірного балансу по одному з каналів R, G, B на N відсотків слід обчислити нове значення колірного каналу за формулою:

$$I = I + N \cdot 128 / 100 \quad (1.4)$$

- Фільтри «Підвищити різкість» і «Розмити»

Ці фільтри реалізуються на основі ядра згортки. Елемент зображення отримує нове значення на основі групи елементів, що примикають до даного.

Область примикання є квадратна матриця, розмірність якої збігається з розміром обраного ядра згортки, і центром в оброблюваному елементі.

Згортка (англ. Convolution) - це операція, що показує «схожість» однієї функції з відбитою і зрушеною копією іншого. Поняття згортки узагальнюється для функцій, визначених на групах, а також заходів [2].

Для обчислення значення використовується матриця, звана ядром згортки. Зазвичай ядро згортки є квадратною матрицею $n * n$, де n - непарне, однак ніщо не заважає зробити матрицю прямокутною. Під час обчислення нового значення обраного пікселя ядро згортки як би «прикладається» своїм центром до даного пікселя. Навколишні пікселі так само накриваються ядром. Далі вираховується сума, де складовими є твори значень пікселів на значення комірки ядра, що накрила даний піксель. Сума ділиться на суму всіх елементів ядра згортки. Отримане значення як раз і є новим значенням обраного пікселя. Якщо застосувати згортку до кожного пікселя зображення, то в результаті вийде якийсь ефект, що залежить від обраного ядра згортки [3].

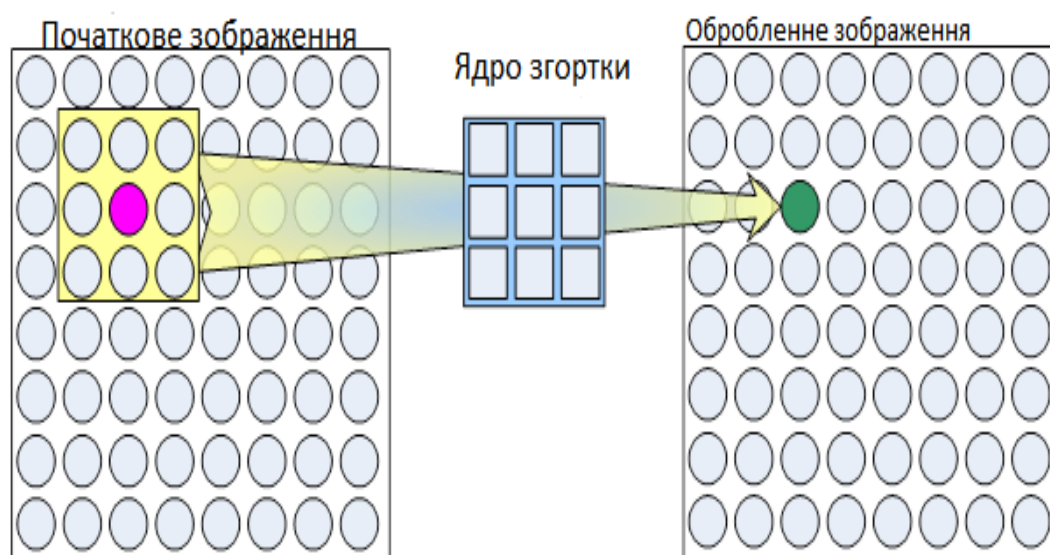


Рисунок 1.1 – Ядро згортки

Ядро згортки є матрицею розміром 3x3, 5x5, 7x7 і т.д., на якій визначена деяка функція. Ядро згортки називається вікном, а задана на ньому функція - ваговий або функцією вікна. Кожному елементу вікна відповідає число, зване ваговим множником. Сукупність усіх вагових множників і становить вагову функцію. Непарні розміри вікна необхідні для однозначного визначення центрального елемента.

По суті, ядро згортки є фільтром, який дозволяє посилити або послабити компоненти зображення. Фільтрація здійснюється переміщенням вікна фільтра по зображенню. Вагова функція в процесі переміщення залишається незмінною. У кожному положенні вікна відбувається операція згортки - лінійна комбінація значень елементів зображення:

$$\begin{vmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{vmatrix} \begin{vmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \\ k_7 & k_8 & k_9 \end{vmatrix} \sum_{i=1}^9 p_i K_i = p'_5$$

де p_i - елементи області примикання,

k_i - вагові множники,

p'_5 - нове значення пікселя.

Рисунок 1.2 – Операція згортки

При кожному положенні вікна вагова функція поелементно множиться на обсяг пропонованих пікселів вихідного зображення і твори підсумовуються. Отримана сума називається відгуком фільтра і присвоюється тому пікселя нового зображення, який відповідає положенню центру вікна.

Результат обробки пікселя записується в відповідну клітинку тимчасової матриці такого ж розміру, як і вихідне зображення. Запис в окрему тимчасову матрицю необхідна для того, щоб виключити вплив вже оброблених пікселів на ще не оброблені.

Обов'язково слід згадати про граничні умови. Наприклад, у верхнього лівого пікселя не існує «сусідів» зліва і зверху, отже, нам нема на що множити коефіцієнти матриці.

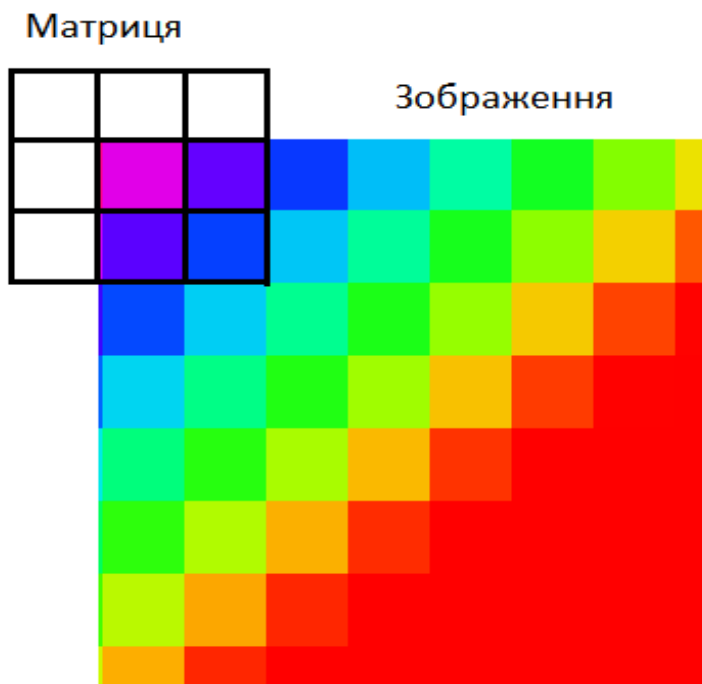


Рисунок 1.3 – Обробка пікселя

Для вирішення цієї проблеми потрібно створення проміжного зображення. Ідея в тому, щоб створювати тимчасове зображення з розмірами:

$$(width + 2 \cdot gap / 2, height + 2 \cdot gap / 2) \quad (1.5)$$

Де $width$ і $height$ - ширина і висота фільтруемого зображення, а gap - розмірність матриці згортки. У центр зображення копіюється вхідна картинка, а краю заповнюються крайніми пікселями зображення. Розмиття застосовується до проміжного буфера, а потім з нього витягується результат [4].

При використанні алгоритму збільшення різкості підкреслюються відмінності між суміжних пікселів і виділяються непомітні деталі. В ядрі різкості центральний коефіцієнт більше 1, а оточений він негативними числами, сума яких на одиницю менше центрального коефіцієнта. Таким чином, збільшується будь-який існуючий контраст між кольором пікселя і кольором його сусідів. При обробці кожного пікселя в зображенні використовується ядро різкості розмірами 3×3 :

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Рисунок 1.4 – Ядро різкості розмірами 3×3

При використанні алгоритму розмиття в зображенні перерозподіляються кольору і пом'якшуються різкі кордону. Ядро згладжування складається з сукупності коефіцієнтів, кожен з яких менше 1, а їх сума становить 1. Це означає, що після фільтрації кожен піксель поглине щось з квітів сусідів, але повна яскравість зображення залишиться незмінною. При обробці зображення використовується наступне ядро згортки [5]:

$$\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Рисунок 1.5 – Ядро згортки при обробці зображення

1.2 Огляд схеми VGG19

Проблематика, що порушується даним алгоритмом відноситься до галузі комп'ютерного зору та зветься «не фотореалістичний рендерінг». Алгоритми що там застосовуються, опрацьовують зображення безпосередньо на рівні кожного окремого пікселя вхідного зображення. Використання ж ЗНМ пошуку об'єктів на зображенні дає змогу виконувати ті ж самі маніпуляції на рівні абстракцій «текстура» та «зміст» зображення.

Існує готова 19-ти шарова архітектура ЗНМ – VGG19-Network, яка використовується у сфері комп'ютерного зору. Вона складається з 16 згорткових та 5 субдискретизаційних шарів (рис. 1.6).

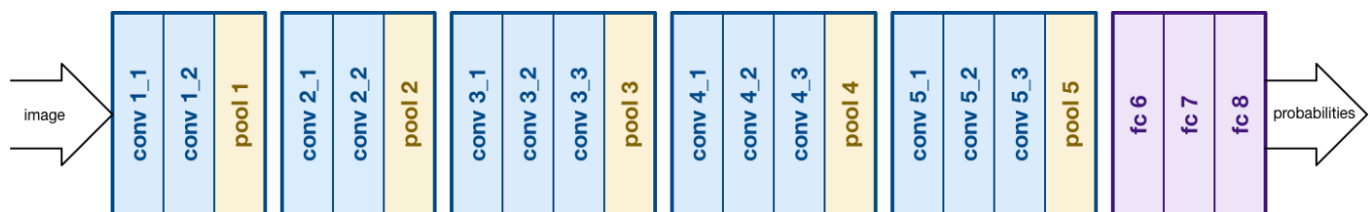


Рисунок 1.6 – Загальна схема VGG19

Саме така мережа використовується під час трансформації стилю але з різницею у правилах пулу субдискретизаційних шарів. Замість максимізації відгуку використовується його усереднення. Така зміна показує кращі результати для нашої конкретної задачі [5].

Кожен шар такої мережі представляє собою фільтр-банк деталей зображення, рівень абстракції якого стає складнішим та вищим з кожним наступним шаром мережі. Шар з N_l чітких фільтрів має N_l карт (наборів деталей) зображення, кожна розміром M_l , де M_l це добуток висоти карти на її ширину. Отже відгуки шару l можуть бути збережені у матриці $F^l \in R^{N_l \times M_l}$ де F_{ij}^l – активація i -го фільтру в позиції j шару l . Для того щоб відновити зміст зображення на кожному шарі, ми використовуємо градієнтний спуск зображення з білим шумом щоб знайти інше зображення, але яке співвідноситься з відгуками змісту оригіналу (матриця F^l). Отже, нехай маємо оригінальне зображення $\vec{p}$ та зображення що відновлюється $\vec{x}$ та їх відповідні карти P^l та F^l для шару l , які і представляють собою зміст зображень. Ми можемо розрахувати середньоквадратичну похибку між змістом зображень таким чином:

$$\lambda_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 \quad (1.6)$$

Похідна функції похибки для шару l :

$$\frac{\partial \lambda_{content}}{\partial F_{ij}^l} = \begin{cases} (F^l - P^l)_{ij}, & F_{ij}^l > 0 \\ 0, & F_{ij}^l < 0 \end{cases} \quad (1.7)$$

Виходячи з чого, ми можемо розрахувати градієнт для $\vec{x}$ стандартним методом зворотного поширення помилки. Цю операцію треба повторювати до отримання мінімуму помилки, а отже – відновленого зображення $\vec{x}$ (рис. 1.3, Content Reconstructions).

Для побудови стильової репрезентації зображення ми використовуємо кореляції між різними картами для кожного шару мережі. Ці кореляції можна представити матрицею Грама $G^l \in R^{N_l \times N_l}$, де G_{ij}^l – скалярний добуток векторів карт i та j шару l :

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \quad (1.8)$$

Спосіб отримання відновленого стилю зображення (його текстури) аналогічний відновленню змісту – градієнтний спуск зображення з білим шумом шляхом мінімізації середньоквадратичної похибки відстані між значеннями матриці Грама для оригінального зображення й генеруємого. Отже, нехай $\vec{a}$ – оригінальне зображення та $\vec{x}$ – зображення що генерується та A^l і G^l їх відповідні стильові репрезентації. Функція втрат для окремого шару виглядає так:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \quad (1.9)$$

Тоді вираз для загальної похибки має вигляд:

$$\lambda_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^L \omega_l E_l \quad (1.10)$$

Де ω_l – ваговий фактор для кожного окремого шару.

Похідна функції похибки для шару l має вигляд:

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} ((F^l)^T (G^l - A^l))_{ji}, & F_{ij}^l > 0 \\ 0, & F_{ij}^l < 0 \end{cases} \quad (1.11)$$

Виходячи з цього, можемо розрахувати градієнти E_l стандартним методом зворотного поширення помилки. Цю операцію треба повторювати до отримання мінімуму помилки, а отже – відновленої текстури $\vec{x}$ (рис. 1.3, Style Reconstructions) [5].

Для отримання зображення що комбінує в собі різні текстури і стиль, достатньо лише поєднати попередні обчислення між собою з використанням вагових факторів (акцентів) для текстури та змісту. Отже, нехай $\vec{p}$ – фотографія, а $\vec{a}$ – стиль (художній малюнок). Тоді функція мінімізації помилки для результуючого зображення (функція відновлення), що поєднує зміст фотографії та стиль малюнку має вигляд:

$$\lambda_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \lambda_{content}(\vec{p}, \vec{x}) + \beta \lambda_{style}(\vec{a}, \vec{x}) \quad (1.12)$$

Де α та β – відповідні вагові фактори або акценти.

Виходячи з останнього виразу та вищевикладеного опису можемо зробити такі висновки щодо програмної реалізації цього алгоритму:

- а) необхідно обрати зручне програмне середовище (фреймворк) для роботи з ЗНМ та об'єктами лінійної алгебри;
- б) необхідна програмна репрезентація VGG19 мережі з можливістю доступу до представлень карт усіх її шарів для обчислення останнього виразу.

1.3 Опис алгоритмів

- Зміна контрастності

У функцію зміни контрастності передається значення кольору поточного пікселя, поточний стан повзунка (*poz*) зміни контрастності і максимальне значення, яке він може приймати (*length*). Обчислюється кількість відсотків:

$$N = (100 / \textit{length}) * \textit{poz} \quad (1.13)$$

1) ЯКЩО $N < 0$, то нове значення кольору кожного каналу розраховується (N береться за модулем) по формулі (1.2)

ІНАКШЕ за формулою (1.3).

2) Контролюємо переповнення змінних:

ЯКЩО $I < 0$, ТО $I = 0$,

ЯКЩО $I > 255$, ТО $I = 255$,

де I - відповідно R , G , B пікселя.

3) return значення кольору пікселя.

- Зміна колірного балансу

У функцію зміни колірного балансу за відповідним каналу передається значення кольору поточного пікселя, поточний стан повзунка (*poz*) зміни колірного балансу і максимальне значення, яке він може приймати (*length*).

1) Обчислюється кількість відсотків:

$$N = (100 / \textit{length}) * \textit{poz} \quad (1.14)$$

2) За формулою (1.4) для відповідного колірний каналу розраховується нове значення.

3) Контролюємо переповнення змінних:

ЯКЩО $I < 0$, ТО $I = 0$,

ЯКЩО $I > 255$, ТО $I = 255$,

де I - відповідний (R , G або B) канал кольору пікселя.

4) return значення кольору пікселя.

- Зміна яскравості

У функцію зміни контрастності передається значення кольору поточного пікселя, поточний стан повзунка (poz) зміни контрастності і максимальне значення, яке він може приймати ($length$).

1) Обчислюється кількість відсотків:

$$N = (100 / length) * poz \quad (1.15)$$

2) ЯКЩО $N < 0$, то нове значення кольору кожного каналу розраховується (N береться за модулем) по формулі (1.2)

ІНАКШЕ за формулою (1.3).

3) Контролюємо переповнення змінних:

ЯКЩО $I < 0$, ТО $I = 0$,

ЯКЩО $I > 255$, ТО $I = 255$,

де I - відповідно R , G , B пікселя.

5) return значення кольору пікселя.

- Підвищення різкості і розмиття

У функцію передаються значення ширини і висоти редагованого зображення, матриця пікселів цього зображення, розмірність матриці згортки і сама матриця.

1) Створення тимчасової матриці розширеного зображення і її заповнення.

2) Застосування ядра згортки для кожного пікселя, при цьому контролюємо переповнення змінних (0 ... 255 для кожного каналу). Нові значення кольорів пікселів заносяться в окрему матрицю ($newpixel$)[5].

3) Повернення функцією матриці newpixel.

1.4 Огляд існуючих програмних засобів що вирішують аналогічні завдання

1.4.1 Instagram

Instagram – соціальна мережа, що базується на обміні фотографіями, дозволяє користувачам робити фотографії, застосовувати до них фільтри, а також поширювати їх через свій сервіс і низку інших соціальних мереж. Є одним із найпопулярніших сервісів у мистецтві айфонографії. Instagram робить фотографії в квадратній формі — як камери Kodak Instamatic і Polaroid. Більшість же мобільних фоторедакторів використовує співвідношення сторін 3:2.

Додаток сумісний зі специфікацією iPhone, iPad і iPod на IOS 4,0 і вище, а також зі смартфонами на Android 4.2.2 і вище з підтримкою OpenGL ES 2, а також доступний для смартфонів, на яких встановлена Windows Phone 8 (8.1). Поширюється воно через App Store, Google Play, Windows Marketplace відповідно [6].

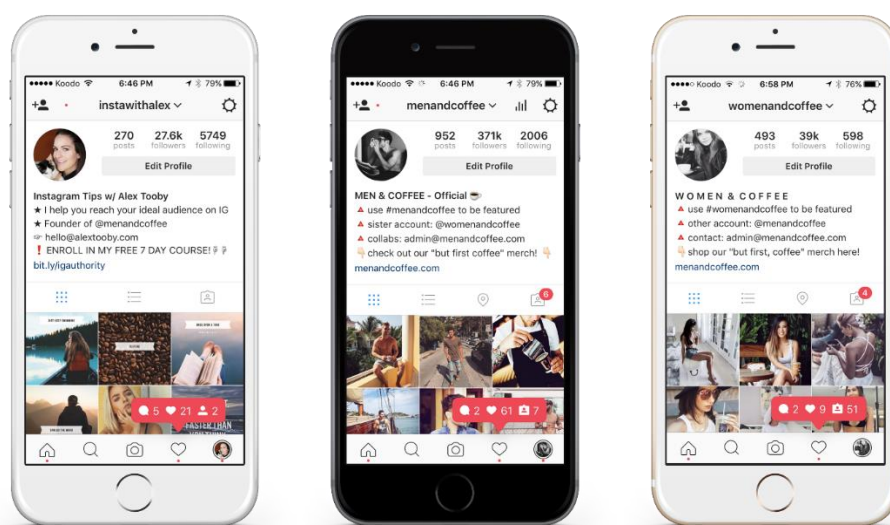


Рисунок 1.7 – Скріншоти стіни користувача Instagram

Основний функціонал Instagram, що використовує машинне навчання для обробки зображень та відео в реальному часі – режим Instagram Stories та Instagram Post. Функція «Редагування» обрати різні фільтри редагування, якими кожен день користуються мільйони користувачів.

1.4.2 Samsung and Apple Camera Apps

Вбудовані у смартфони додатки «Камера» у сучасних версіях Android та iOS (рис. 1.8) використовують машинне навчання для розпізнавання облич, масок, покращення зображення, видалення (затемнення) фону та багато іншого функціоналу, що працює в автоматичному режимі. Власне – автоматичність є головним здобутком алгоритмів машинного навчання.

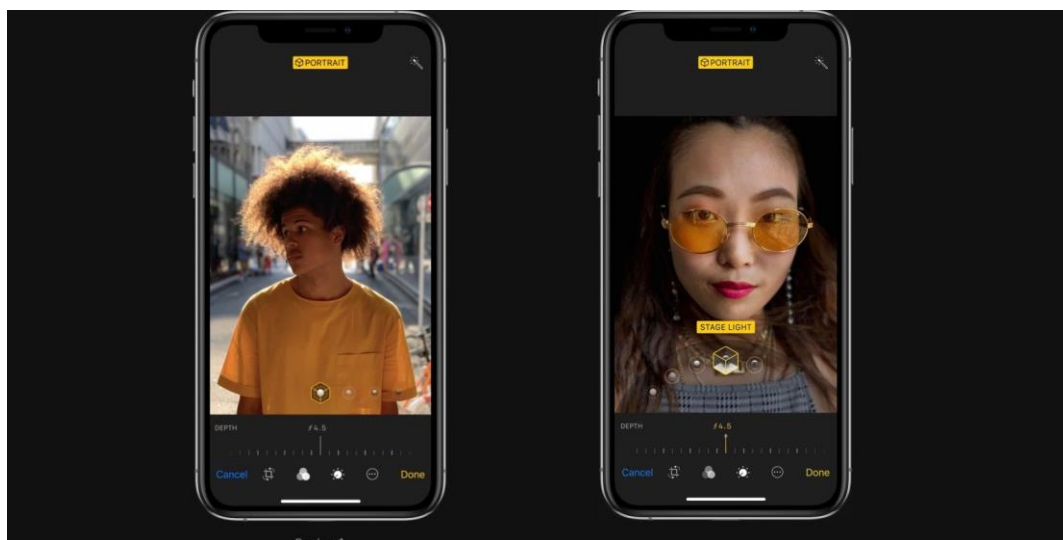


Рисунок 1.8 – iOS Camera App у режимі «Художній портрет»

1.4.3 Adobe Photoshop

Adobe Photoshop - багатофункціональний графічний редактор, розроблений і поширюваний фірмою Adobe Systems. В основному працює з растровими зображеннями, проте має деякі векторні інструменти. Продукт є лідером ринку в області комерційних засобів редагування растрових зображень і найбільш відомою програмою фірми Adobe.

В даний час Photoshop доступний на платформах macOS, Windows і iPadOS. Для Windows Phone і Android доступна спрощена версія програми під назвою Adobe Photoshop Touch. Також існує версія Photoshop Express для Windows Phone 8 і 8.1. У 2014 році в США проходило бета-тестування потокової версії продукту для Chrome OS. Ранні версії редактора були перенести під SGI IRIX, але офіційна підтримка припинена починаючи з третьої версії продукту. Для версій 8.0 та CS6 можливий запуск під Linux за допомогою альтернативи Windows API – Wine [7].

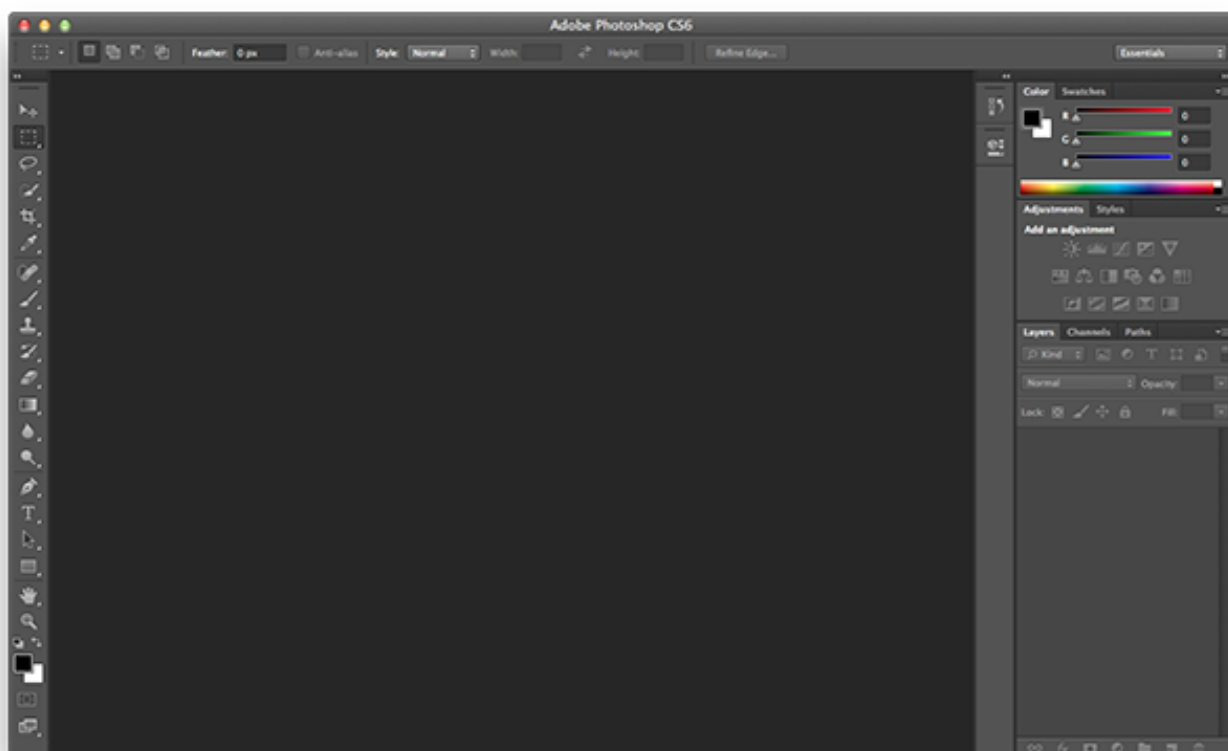


Рисунок 1.9 – Функціонал Adobe Photoshop

1.5 Вибір архітектури розроблюваної системи

Так як розроблювана система перш за все є програмним засобом трансформації стилю, для неї може бути використана будь-яка архітектура, адже в основі лежить алгоритм і засоби його обробки, які є однаковими для будь-якої платформи – Web, Desktop, Mobile чи Embedded. Тому розроблений ПЗ має бути цільним ядром машинного навчання, яке має програмний інтерфейс (API) розроблений таким чином, аби забезпечити виконання наступної логіки роботи: запуск з вхідним зображенням, обробка, завершення з вихідним трансформованим зображенням.

Готове ПЗ можна розмістити на будь-якій платформі, додавши до нього додаткове програмне забезпечення управління – сервер, плагін або додаток-інтерфейс.

Клієнт-серверна архітектура підійде для організації на Web-платформі. Для цього необхідно створити середовище у вигляді контейнеру Docker з необхідною операційною системою, розмістити на ньому трансформатор стилю, сервер, що буде здійснювати керування трансформатором через його програмний інтерфейс, та веб-сторінку для доступу та управління користувачем. Звичайно, хостинг має підтримувати зовнішні статичні адреси. Можливо також поєднати трансформатор стилю та сервер в одному процесі, проте це зменшить загальну гнучкість системи. Загальна схема такої архітектури наведена нижче.

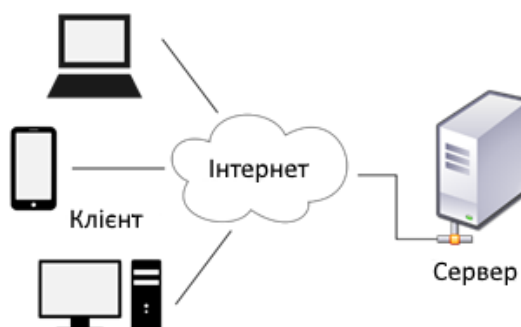


Рисунок 1.10 – Загальна схема клієнт-серверної архітектури

Наступний варіант платформи – Desktop та Mobile. З архітектурної точки зору вони є однаковими. Обидві мають під собою високорівневу розробку під певну операційну систему. Так само як і у Web-платформі, трансформатор стилю буде запускатися як дочірній процес основного додатку, якій і буде містити в собі графічний інтерфейс користувача та всю необхідну бізнес-логіку. Така архітектура називається міжпроцесною взаємодією.

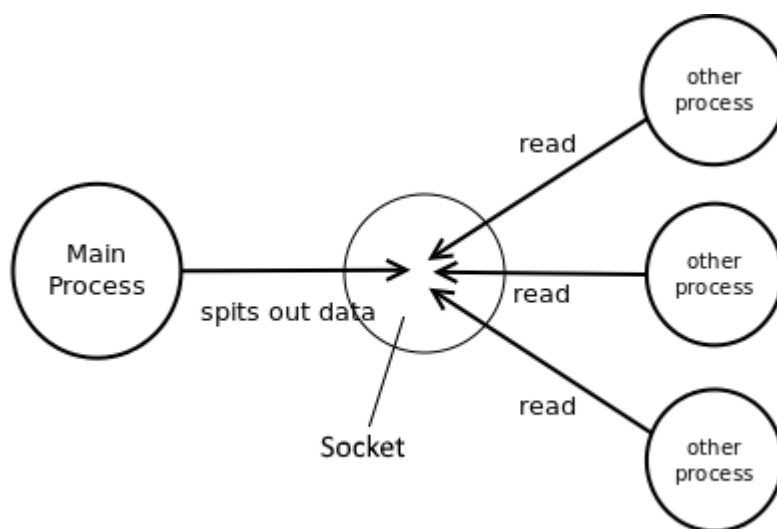


Рисунок 1.11 – Загальна схема міжпроцесної взаємодії

Остання платформа – це Embedded. Вона має під собою розробку системи на найнижчому апаратному рівні. У цьому також все зводиться до міжпроцесної взаємодії, за виключенням того що ресурси дуже обмежені разом з інструментарієм розробки, проте продуктивність такої системи є найвищою.

Гарним рішенням буде використання апаратного прискорення для трансформатора стилю, будь то потужний графічний процесор або будь-який спеціалізований AI-прискорювач.

Виходячи з вимог кросплатформенності розроблюваного трансформатору стилю та простоти реалізації готового програмного засобу, що демонструє його роботу, було обрано Desktop платформу для розробки. Готове програмне забезпечення матиме власний програмний інтерфейс та буде запускатися дочірнім процесом GUI-додатку. Цільовою операційною системою було обрано Windows.

2 ОПИС ПРОГРАМИ

2.1 Структура та функціонал розроблюваної програми

Для побудови призначеного для користувача інтерфейсу застосовується графіч-ська підсистема Windows Forms. Під час виконання програми використовуються класи Form1, Form2, Form3, BrightnessContrast, ColorBalance, Filter, які включають такі методи:

- а) Brightness - метод, що змінює яскравість поточної точки.
- б) Contrast - метод, що змінює контрастність поточної точки.
- в) ColorBalance_R - метод, що змінює колірний баланс точки по каналу R.
- г) ColorBalance_G - метод, що змінює колірний баланс точки по каналу G.
- д) ColorBalance_B - метод, що змінює колірний баланс точки по каналу B.
- е) matrix_filtration - метод, що виконує фільтрацію зображення на основі ядра згортки.

2.2 Аналіз програмних засобів

2.2.1 Обґрунтування використання мови програмування Csharp

CSharp - об'єктно-орієнтована мова програмування. Відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML [8].

Переїнявши багато від своїх попередників - мов C ++, Delphi, Модула, Smalltalk і, особливо, Java - C #, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C # на відміну від C ++ не підтримує множинне успадкування класів (між тим допускається множинна реалізація інтерфейсів). Серед основних її переваг можна назвати такі:

- а) чистий синтаксис (для виділення блоків слід використовувати відступи);
- б) переносність програм (що властиве більшості інтерпретованих мов);
- в) стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- г) можливість використання Sharp в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- д) стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDE;
- е) зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- ж) відкритий код (можливість редагувати його іншими користувачами).

CSharp має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови CSharp може бути розширений функціями та типами даних, розробленими на C чи C ++ (або на іншій мові, яку можна викликати із C). CSharp також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

2.2.2 Обґрунтування використання Windows Forms

Windows Forms - інтерфейс програмування додатків (API), що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Причому керований код - класи, що реалізують API для Windows Forms, що не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C #, C ++, так і на VB.Net, J # і ін.

З одного боку, Windows Forms розглядається як заміна більш старої і складної бібліотеки MFC, спочатку написаної на мові C ++. З іншого боку, WF не пропонує парадигму, порівнянну з MVC. Для виправлення цієї ситуації і реалізації даної функціональності в WF існують сторонні бібліотеки. Однією з найбільш використовуваних подібних бібліотек є User Interface Process Application Block, випущена спеціальною групою Microsoft, що займається прикладами і рекомендаціями, для безкоштовного скачування. Ця бібліотека також містить вихідний код і навчальні приклади для прискорення навчання. Microsoft Visual Studio - лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight [9].

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1.1 Макет графічного інтерфейсу користувача

Макет побудовано на основі шаблону «Головного Вікна» програми. На ньому розміщується віджет з панелями, кожна вкладка якої відповідає за свій потік виконання (рис. 3.1). Нижче наведені відповідні зображення розробленого макету:

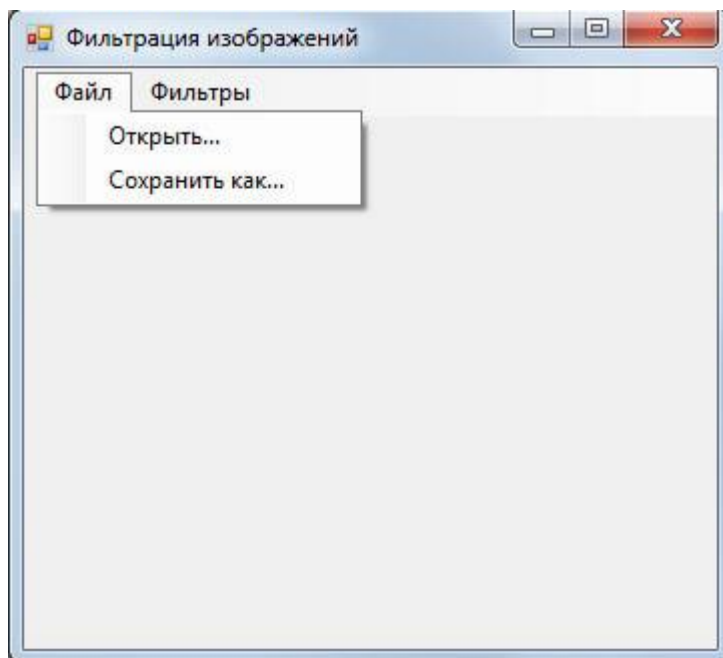


Рисунок 3.1 – Макет головного меню програми

У вкладці «Файл» знаходяться кнопки для відкриття і збереження зображення. У вкладці «Фільтри» представлені інструменти для фільтрації завантаженого зображення: «Яскравість / контрастність» (рис 3.2), «Баланс кольору» (рис 3.3), «Підвищити різкість», «Розмити». При виклику перших двох відкриваються нові вікна, в яких знаходяться слайдери для зміни відповідних компонентів фільтруемого зображення і кнопки «Застосувати», щоб зберегти зміни.

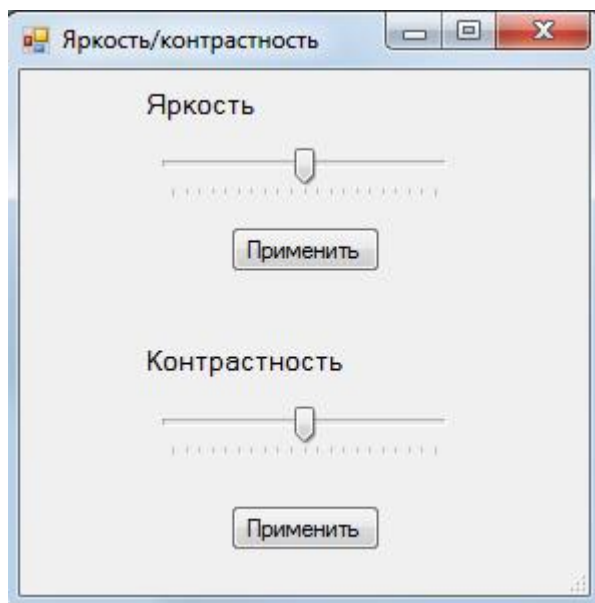


Рисунок 3.2 – Фільтр «Яскравість/Контрастність»

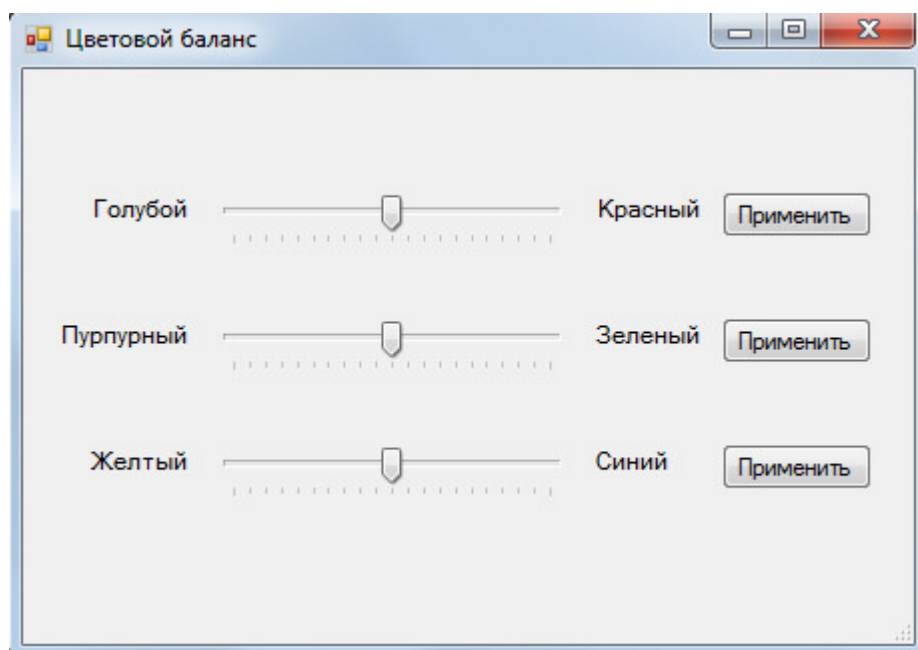


Рисунок 3.3 – Фільтр «Баланс кольору»

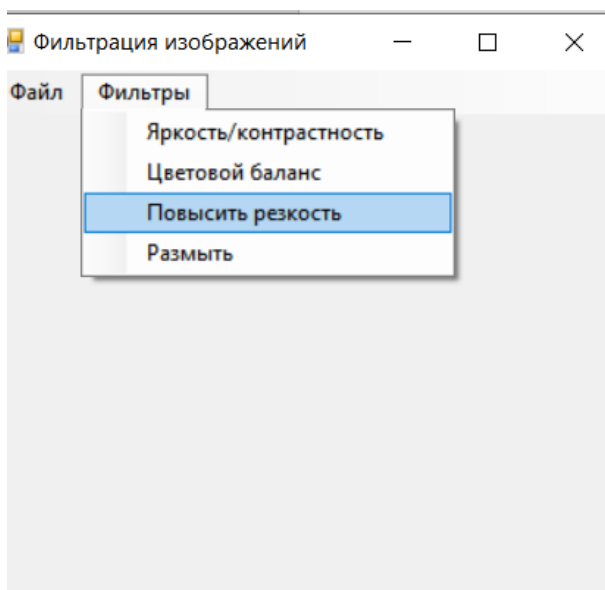


Рисунок 3.4 – Фільтри «Підвищити різкість» та «Розмити»

3.3 Особливості реалізації системи

Основними особливостями реалізації системи є:

- а) автономність фільтратора зображень (він надає простий та зручний програмний інтерфейс, керуюче ПЗ може бути будь-яким);
- б) використання мови програмування CSharp;
- в) використання інтерфейсу Windows Forms;
- г) використання Qt Framework для створення зручного та стилізованого GUI;
- д) використання фільтр зображень в якості дочірнього процесу кінцевою програмної реалізації для користувача;
- е) можливість трансформації як зображень (підтримуються формати PNG, JPG, BMP) та можливість одночасного перегляду вхідних та вихідних даних для наочності;
- ж) вхідними та вихідними даними є зображення.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Призначенням програмного продукту є надання можливості кінцевому користувачу накладати різні фільтри обробки зображень, з можливістю швидкого перегляду отриманих результатів. За фільтрацію зображень відповідає накладання стилю відповідає метод, що виконує фільтрацію зображення на основі ядра згортки - `matrix_filtration`, а за інтерфейсну частину – Windows Forms, що є звичайним клієнтським програмним засобом. Так як кінцева реалізація системи має під собою поєднання цих двох модулів, надалі вони будуть розглядатися разом як одне ціле.

Характеристики комп'ютера, необхідні для стабільної та швидкої роботи програми повинні задовольняти вимогам зазначеним в таблиці 4.1:

Таблиця 4.1 – Системні вимоги

Складова комп'ютера	Характеристика
CPU	Intel або AMD 4-core, 3.0GHz або краще
GPU	NVIDIA GeForce 750ti, 2Gb VRAM або краще
RAM	8Gb або більше
Простір на диску	150Mb
Додаткове ПЗ	Visual Studio 2019
Операційна система	Windows 10, macOS High Sierra, Linux (Ubuntu, CentOS, RedHat)

4.2 Характеристики програми

Після запуску програми відобразиться головне вікно програми.

Робота програми організована у вигляді віконного режиму, з використанням таких компонентів: панель обрання фото та зберігання фото, віджет зі списком фільтрів та переглядач зображень.

Конфігурація програми здійснюється шляхом внесення змін до файлу конфігурації `.NETFramework,Version=v4.5.AssemblyAttributes.cs` у форматі `cs`. Його зміст викладено у таблиці 4.2:

Таблиця 4.2 – Зміст файлу конфігурації `.NETFramework,Version=v4.5.AssemblyAttributes.cs`

Поле	Значення
<code>Program.Properties.Resources.resources</code>	Шлях до зміни кольору
<code>Program.exe</code>	Шлях до запуску програми
<code>Program.pdb</code>	Опис кольорів
<code>Program.Form2.resources</code>	Макет Windows Forms
<code>Program.exe.config</code>	Конфігурація програми
<code>Program.Form1.resources</code>	Макет Windows Forms
<code>Program.Form3.resources</code>	Макет Windows Forms
<code>Program.csproj.GenerateResource.cache</code>	Генератор кольорів

4.3 Підготовка програми до запуску

Щоб підготувати систему до роботи необхідно виконати наступні кроки:

- а) обрати комп'ютер що відповідає системним вимогам (табл. 4.1);

- б) встановити Visual Studio 2019
- г) розмістити клієнтський модуль на диску з супроводжуваними бібліотеками та файлом конфігурації (подається в архіві);
- д) налаштувати клієнтський модуль для коректної роботи з модулем трансформації шляхом правильного заповнення файлу конфігурації `.NETFramework,Version=v4.5.AssemblyAttributes.cs` описаний у таблиці 4.2.

4.4 Вхідні і вихідні дані

У ролі вхідних даних виступають фільтри та не трансформоване зображення.

У ролі вихідних даних виступають трансформоване зображення.

Підтримувані формати зображення: PNG, JPG, BMP.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Призначенням програмного продукту є надання можливості кінцевому користувачу накладати фільтри обробки на файли зображень, з можливістю швидкого перегляду отриманих результатів.

5.2 Умови виконання програми

Характеристики комп'ютера, необхідні для стабільної та швидкої роботи програми повинні задовольняти вимогам зазначеним в таблиці 5.1:

Таблиця 5.1 – Системні вимоги

Складова комп'ютера	Характеристика
CPU	Intel або AMD 4-core, 3.0GHz або краще
GPU	NVIDIA GeForce 750ti, 2Gb VRAM
RAM	8Gb або більше
Простір на диску	150Mb
Додаткове ПЗ	NVIDIA CUDA 10.0, Visual Studio 2019
Операційна система	Windows 10, macOS High Sierra, Linux (Ubuntu, CentOS, RedHat)

5.3 Встановлення програми

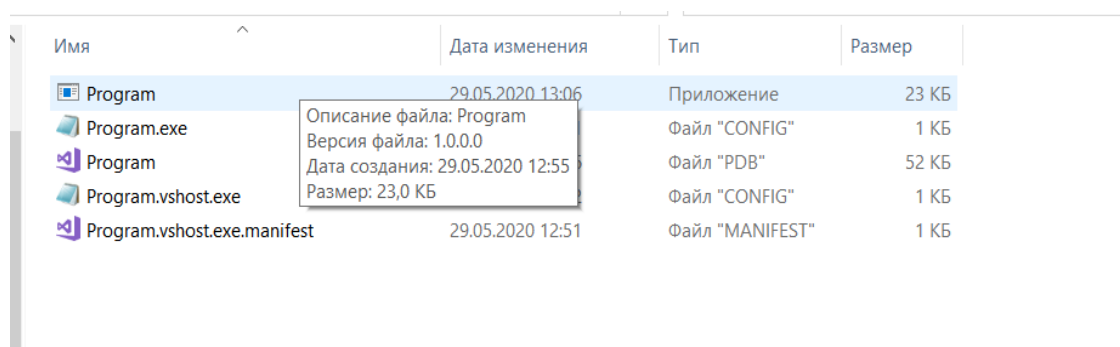
Для розміщення програми на фізичному носію необхідно розпакувати архіви з фільтрами обробки та клієнтським модулем і налаштувати конфігурацію останнього. Після цього програма готова до використання.

5.4 Видалення програми

Для видалення програми з фізичного носія, необхідно видалити розпаковані у файли програми (дивись пункт 5.3).

5.5 Виконання програми

Для запуску програми треба перейти в каталог, до якого було розпаковано клієнтську частину системи та запустити файл Program (рис. 5.1).



Имя	Дата изменения	Тип	Размер
Program	29.05.2020 13:06	Приложение	23 КБ
Program.exe		Файл "CONFIG"	1 КБ
Program		Файл "PDB"	52 КБ
Program.vshost.exe		Файл "CONFIG"	1 КБ
Program.vshost.exe.manifest	29.05.2020 12:51	Файл "MANIFEST"	1 КБ

Рисунок 5.1 – Виконуючий файл Program

Після запуску з'явиться головне вікно програми, де можна обрати потрібний файл (рис. 5.2).

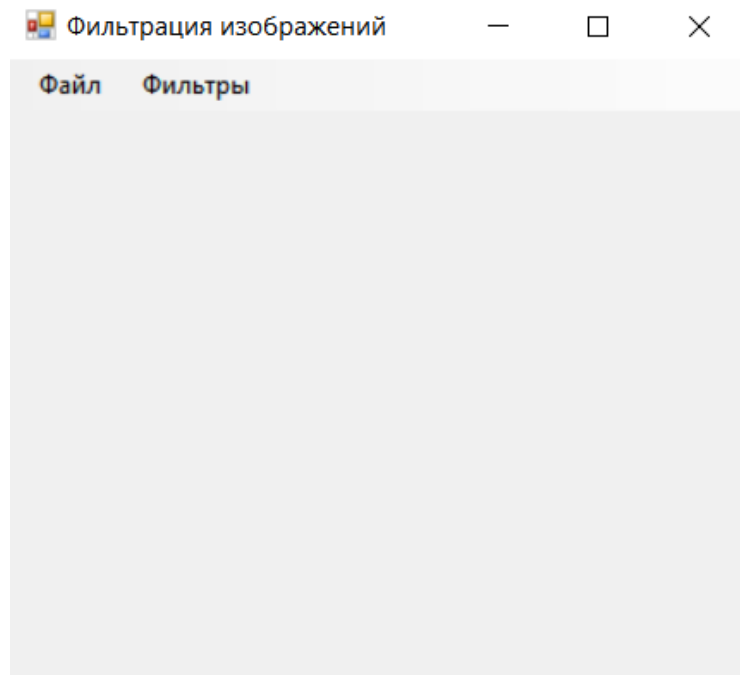


Рисунок 5.2 – Головне вікно програми

Для початку обробки фото, необхідно спочатку обрати фото та відкрити його (рис. 5.3).

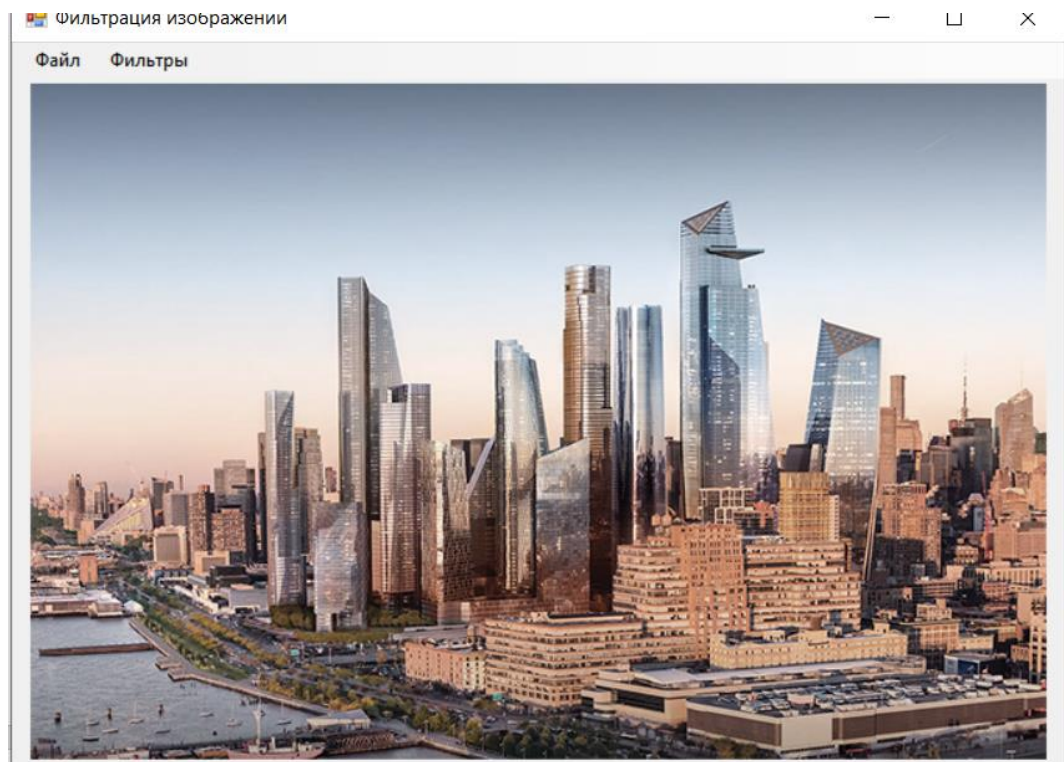


Рисунок 5.3 – Обране зображення

Після чого, для обробки зображення необхідно перейти на відповідну панель, яка містить фільтри обробки (рис. 5.4).

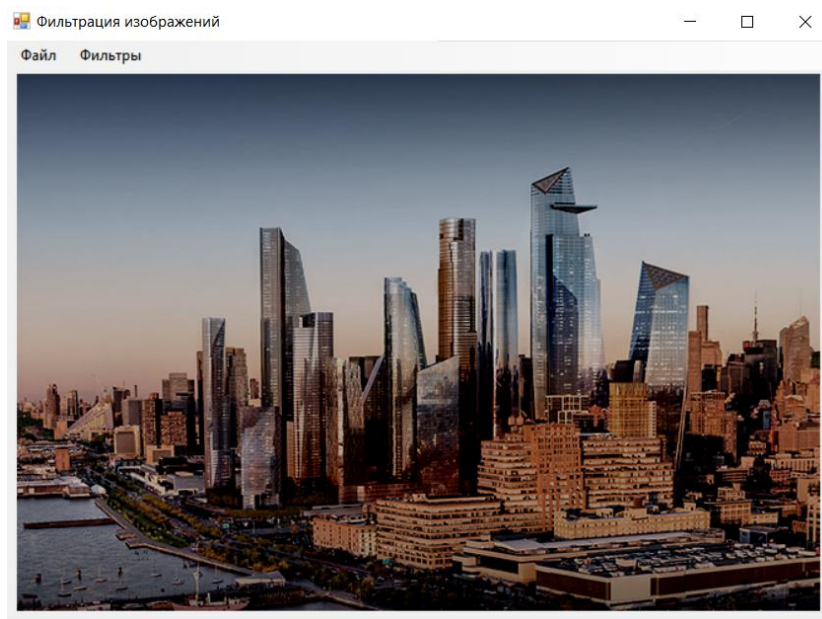


Рисунок 5.4 – Обробка фото

На данному прикладі було застосовано фільтри «Яскравість/Контраснісіність».

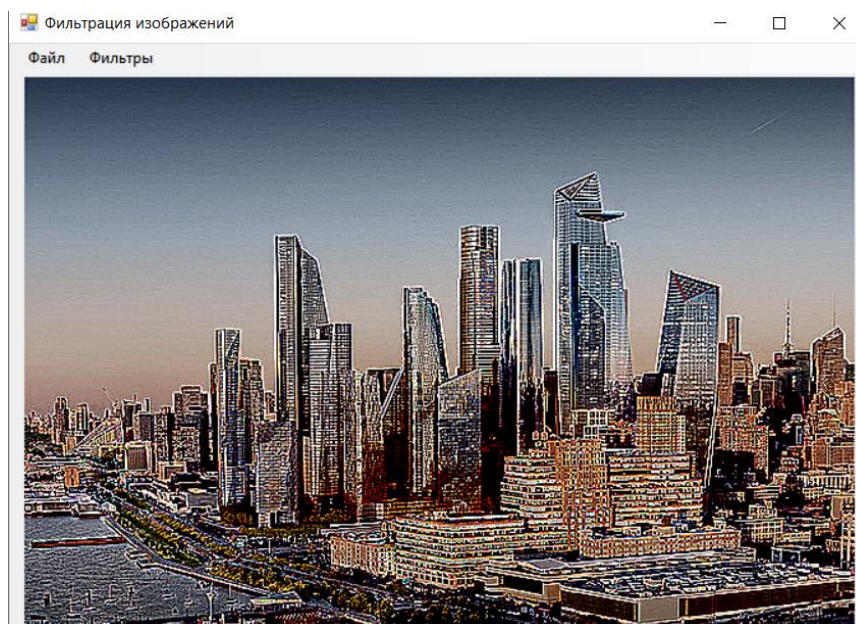


Рисунок 5.5 – Обробка фото

Після обробки фото за допомогою фільтрів «Яскравість/Контрасність» було використано фільтр «Повищення різкості».

Для того щоб використати фільтр «Розмити» завантажемо початкове зображення:

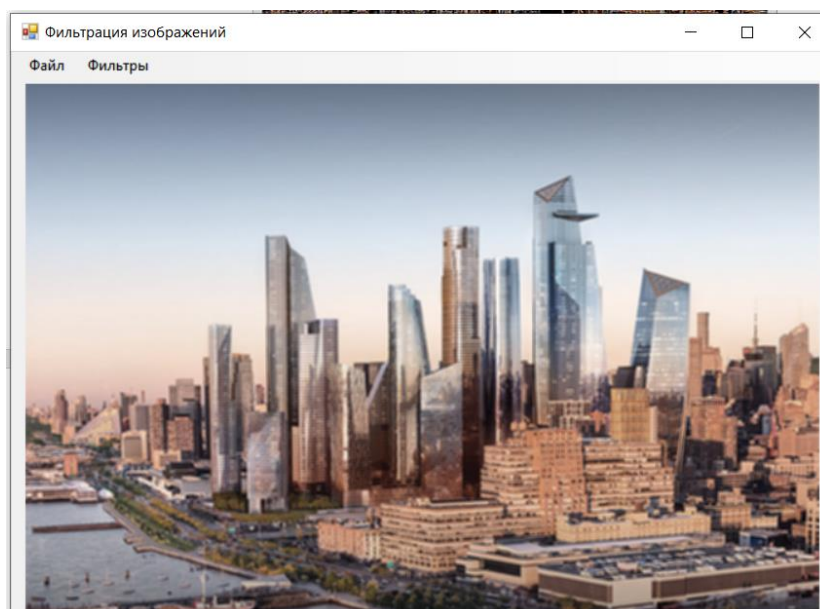


Рисунок 5.6 – Обробка фото

Коли сподобався результат обробки фото треба можна його зберігти та обрати місце зберігання на ПК:

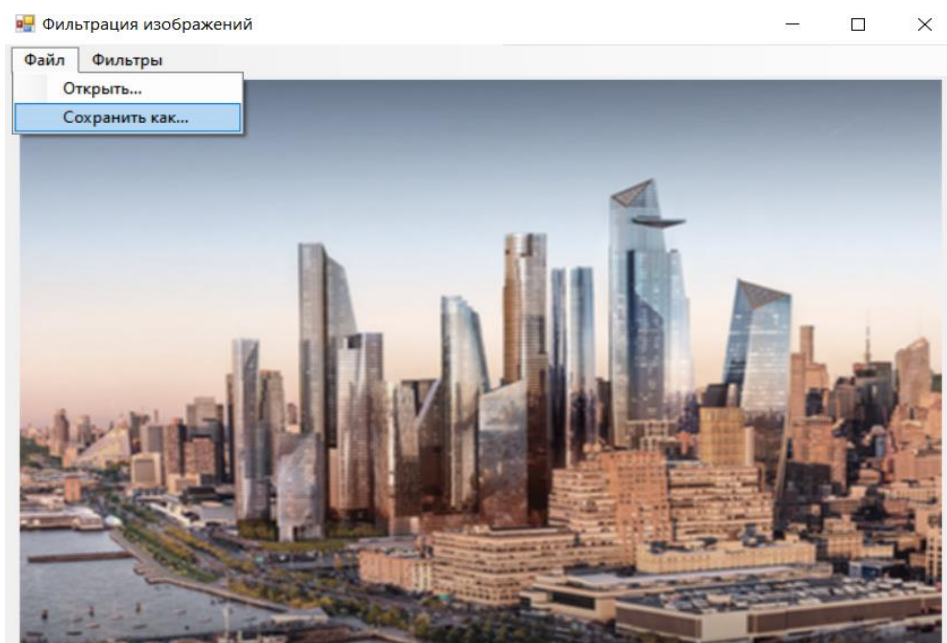


Рисунок 5.7 – Збереження фото

ВИСНОВКИ

В ході виконання дипломної роботи було вирішено такі задачі:

- проаналізовано предметну область, розглянуто структуру методів обробки фото та методи лінійних обчислень над ними;
- проведено аналіз існуючих програмних засобів, що вирішують аналогічні завдання;
- проведено аналіз засобів розробки які були застосовані під час створення розроблюваної системи;
- розглянуто скриптову мову програмування CSharp як швидкий інструмент створення кросплатформових програмних засобів;
- розглянуто можливості графічного інтерфейсу Widows Forms.

У результаті проведеної роботи, за допомогою середовища програмування MS Visual Studio 2019, Widows Forms мови програмування CSharp, було створено програмне забезпечення для обробки зображень.

Розроблена система дозволяє:

- переглядати наявні для фільтри обробки;
- трансформувати фільтрів зображень.

Розроблена система може покращуватися у майбутньому шляхом додавання нових фільтрів, що має значно підвищити швидкість роботи системи.

Також можлива більш загальна реалізація системи шляхом розробки серверу керування модулем трансформації стилів і його графічної інтерфейсної частини у вигляді веб-сторінки. Іншими словами – реалізація у вигляді веб-сервісу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Цифрове зображення Електронний ресурс // Вільна енциклопедія Wikipedia. – Режим доступу: https://uk.wikipedia.org/wiki/Цифрове_зображення.
2. Баланс кольору [Електронний ресурс] – Режим доступу: https://ru.qwe.wiki/wiki/Color_balance.
3. Обробка зображень [Електронний ресурс] – Режим доступу: <http://algolist.ru/graphics/>.
4. Матрична фільтрація [Електронний ресурс] – Режим доступу: <https://habr.com/ru/post/142818/>
5. Фільтрація зображень методом згортки [Електронний ресурс] – Режим доступу: <https://habr.com/ru/post/62738/>.
6. Instagram [Електронний ресурс] – Режим доступу: wikipedia.org/wiki/Instagram
7. Adobe Photoshop [Електронний ресурс] // Вільна енциклопедія Wikipedia. – Режим доступу: https://ru.wikipedia.org/wiki/Adobe_Photoshop
8. C Sharp [Електронний ресурс] // Вільна енциклопедія Wikipedia. – Режим доступу: wikipedia.org/wiki/C_Sharp.
9. Microsoft Visual Studio [Електронний ресурс] // Вільна енциклопедія Wikipedia. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.

ДОДАТОК А Специфікація вимог

А.1 Основні вимоги до програми що розроблюється

А.1.1 Функціональні вимоги

В якості вхідних даних виступають зображення, завантажений користувачем, на який треба накласти фільтр, який також обирає користувач з певного фіксованого набору.

В якості вихідних даних виступають трансформовані зображення, які надаються користувачу.

Інтерфейс програми повинен забезпечувати просте та ефективне спілкування користувача з програмним засобом, надавати можливість візуально бачити набір стилів та вхідні і вихідні зображення.

До основних функцій програми повинно входити:

- а) надавати можливість графічного вибору накладаємого фільтру з фіксованого набору стилів;
- б) здійснення трансформації фільтру обробки зображення шляхом накладання фільтру обраного користувачем;
- в) детальний перегляд вхідних та вихідних даних.

А.1.2 Вимоги до надійності

Для коректної роботи програми користувач повинен слідувати інструкціям з використання програми, викладеним у «Керівництві оператора». Інтерфейс має розроблено спеціальним чином, аби користувач в кожний момент часу знав, що відбувається з програмою та що робити далі, тобто – інтерфейс має бути інтуїтивним. Також перед початком використання програми мають бути виконані вимоги з «Керівництва програміста».

А.1.3 Умови експлуатації

Програма має виконуватися у середовищі операційної системи Windows, бути записана на диск.

Експлуатація програми здійснюється відповідно «Керівництву оператора» та «Керівництву програміста», які містять всю необхідну інформацію для ознайомлення з функціоналом програми та її умовами експлуатації.

А.1.4 Вимоги до складу та параметрів технічних засобів

Для нормальної роботи з програмою необхідна наступна мінімальна конфігурація апаратно-програмного забезпечення комп'ютеру:

- а) процесор Intel або AMD 4-core;
- б) графічний процесор NVIDIA GeForce 750ti, 2Gb VRAM;
- в) оперативна пам'ять 8Gb або більше;
- г) простір на диску 150Mb;
- д) операційна система Windows 10;
- е) монітор;
- ж) клавіатура;
- з) маніпулятор «миша».

А.1.5 Вимоги до маркування та пакування

Програма може поставлятися на флеш-накопичувачі, оптичному диску або розповсюджуватися архівом через мережу Інтернет.

А.1.6 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання аналогічні вимогам, які пред'являються до носія, на якому розповсюджується та зберігається програма.

А.1.7 Вимоги до програмної документації

Програмний засіб повинен поставлятися з «Керівництвом програміста» та «Керівництвом оператора».

ДОДАТОК Б Текст програми

Б.1 Текст програми модулю ml_core

Б.1.1 Файл Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Program
{
    static class Program
    {
        /// <summary>
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Б.1.2 Файл AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

[assembly: AssemblyTitle("Program")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Program")]
[assembly: AssemblyCopyright("Copyright © 2013")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

[assembly: ComVisible(false)]

[assembly: Guid("36e30e36-7ef5-4573-b73d-fabb371538c7")]

[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Б.1.3 Файл Resources.Designer.cs

```

namespace Program.Properties
{

    /// <summary>
    /// </summary>

    [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "4.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class Resources
    {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;

        [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]
        internal Resources()
        {
        }

        /// <summary>
        /// </summary>

        [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager
        {
            get
            {
                if ((resourceMan == null))
                {
                    global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("Program.Properties.Resources",
typeof(Resources).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }

        /// <summary>
        /// </summary>

        [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture
        {
            get
            {
                return resourceCulture;
            }
        }
    }
}

```

```

    }
    set
    {
        resourceCulture = value;
    }
}
}
}

```

Б.1.4 Файл App.config

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
</configuration>

```

Б.1.5 Файл BrightnessContrast.cs

```

namespace Program
{
    class BrightnessContrast
    {
        public static UInt32 Brightness (UInt32 point, int poz, int lenght)
        {
            int R;
            int G;
            int B;

            int N = (100 / lenght) * poz; //кол-во процентов

            R = (int)((point & 0x00FF0000) >> 16) + N * 128 / 100;
            G = (int)((point & 0x0000FF00) >> 8) + N * 128 / 100;
            B = (int)((point & 0x000000FF) + N * 128 / 100);

            if (R < 0) R = 0;
            if (R > 255) R = 255;
            if (G < 0) G = 0;
            if (G > 255) G = 255;
            if (B < 0) B = 0;
            if (B > 255) B = 255;

            point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) |
            ((UInt32)B);

            return point;
        }
    }
}

```

```

public static UInt32 Contrast(UInt32 point, int poz, int lenght)
{
    int R;
    int G;
    int B;

    int N = (100 / lenght) * poz; //кол-во процентов

    if (N >= 0)
    {
        if (N == 100) N = 99;
        R = (int)((((point & 0x00FF0000) >> 16) * 100 - 128 * N) / (100 -
- N));
        G = (int)((((point & 0x0000FF00) >> 8) * 100 - 128 * N) / (100 -
N));
        B = (int)((((point & 0x000000FF) * 100 - 128 * N) / (100 - N));
    }
    else
    {
        R = (int)((((point & 0x00FF0000) >> 16) * (100 - (-N)) + 128 *
(-N)) / 100);
        G = (int)((((point & 0x0000FF00) >> 8) * (100 - (-N)) + 128 * (-
N)) / 100);
        B = (int)((((point & 0x000000FF) * (100 - (-N)) + 128 * (-N)) /
100);
    }

    if (R < 0) R = 0;
    if (R > 255) R = 255;
    if (G < 0) G = 0;
    if (G > 255) G = 255;
    if (B < 0) B = 0;
    if (B > 255) B = 255;

    point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) |
((UInt32)B);

    return point;
}
}

```

Б.1.6 Файл ColorBalance.cs

```

using System;

namespace Program
{
    class ColorBalance
    {
        public static UInt32 ColorBalance_R(UInt32 point, int poz, int lenght)
        {
            int R;
            int G;

```

```

        int B;

        int N = (100 / lenght) * poz;

        R = (int)((point & 0x00FF0000) >> 16) + N * 128 / 100;
        G = (int)((point & 0x0000FF00) >> 8);
        B = (int)(point & 0x000000FF);

        if (R < 0) R = 0;
        if (R > 255) R = 255;

        point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) |
        ((UInt32)B);

        return point;
    }

    public static UInt32 ColorBalance_G(UInt32 point, int poz, int lenght)
    {
        int R;
        int G;
        int B;

        int N = (100 / lenght) * poz; //кол-во процентов

        R = (int)((point & 0x00FF0000) >> 16);
        G = (int)((point & 0x0000FF00) >> 8) + N * 128 / 100;
        B = (int)(point & 0x000000FF);

        if (G < 0) G = 0;
        if (G > 255) G = 255;

        point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) |
        ((UInt32)B);

        return point;
    }

    public static UInt32 ColorBalance_B(UInt32 point, int poz, int lenght)
    {
        int R;
        int G;
        int B;

        int N = (100 / lenght) * poz;

        R = (int)((point & 0x00FF0000) >> 16);
        G = (int)((point & 0x0000FF00) >> 8);
        B = (int)((point & 0x000000FF) + N * 128 / 100);

        if (B < 0) B = 0;
        if (B > 255) B = 255;

        point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) |
        ((UInt32)B);

        return point;
    }
}

```

```
}
```

Б.1.7 Файл MatricaSvertki.cs

```
using System;

namespace Program
{
    struct RGB
    {
        public float R;
        public float G;
        public float B;
    }

    class Filter
    {
        public static UInt32[,] matrix_filtration(int W, int H, UInt32[,] pixel,
int N, double[,] matryx)
        {
            int i, j, k, m, gap = (int)(N / 2);
            int tmpH = H + 2 * gap, tmpW = W + 2 * gap;
            UInt32[,] tmppixel = new UInt32[tmpH, tmpW];
            UInt32[,] newpixel = new UInt32[H, W];

            for (i = 0; i < gap; i++)
                for (j = 0; j < gap; j++)
                {
                    tmppixel[i, j] = pixel[0, 0];
                    tmppixel[i, tmpW - 1 - j] = pixel[0, W - 1];
                    tmppixel[tmpH - 1 - i, j] = pixel[H - 1, 0];
                    tmppixel[tmpH - 1 - i, tmpW - 1 - j] = pixel[H - 1, W - 1];
                }

            for (i = gap; i < tmpH - gap; i++)
                for (j = 0; j < gap; j++)
                {
                    tmppixel[i, j] = pixel[i - gap, j];
                    tmppixel[i, tmpW - 1 - j] = pixel[i - gap, W - 1 - j];
                }

            for (i = 0; i < gap; i++)
                for (j = gap; j < tmpW - gap; j++)
                {
                    tmppixel[i, j] = pixel[i, j - gap];
                    tmppixel[tmpH - 1 - i, j] = pixel[H - 1 - i, j - gap];
                }

            for (i = 0; i < H; i++)
                for (j = 0; j < W; j++)
                    tmppixel[i + gap, j + gap] = pixel[i, j];

            RGB ColorOfPixel = new RGB();
            RGB ColorOfCell= new RGB();
            for (i = gap; i < tmpH - gap; i++)
                for (j = gap; j < tmpW - gap; j++)
```

[illegible]

Б.1.8 Файл Program.vshost.exe.manifest

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <requestedExecutionLevel level="asInvoker" uiAccess="false"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

Програмна реалізація алгоритмів машинного навчання для обробки зображень

Яценко Марія Володимирівна
студентка групи КНТ-827-СП



Активация Windows
Чтобы активировать Windows, перейдите
по ссылке

Рисунок В.1 – Тема роботи

Об'єкт, предмет та мета роботи

Об'єкт розробки: використання фільтрів обробки для трансформації стилю зображення.

Предмет дослідження: алгоритми накладання фільтрів обробки зображення на зображення з використанням ядра сгортки.

Мета роботи бакалавра: розробка програмної системи для обробки стилю зображення.

Активация Windows 2
Чтобы активировать Windows, перейдите
по ссылке

Рисунок В.2 – Об'єкт, предмет та мета роботи

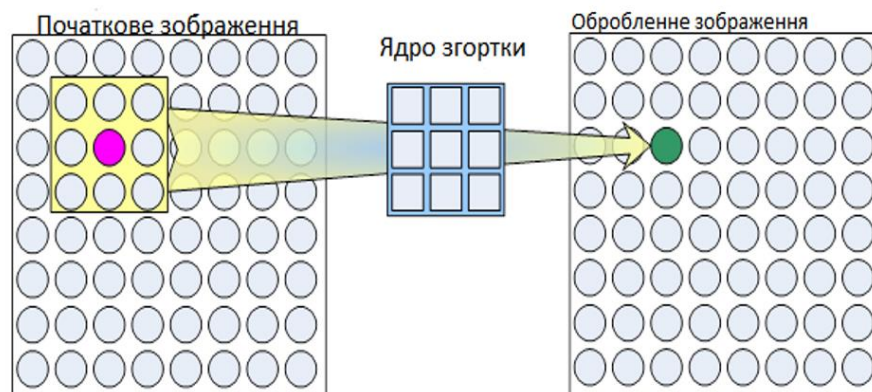
Існуючі програмні засоби що вирішують схожі задачі:

- **Instagram**
- **Apple iOS Camera**
- **Adobe Photoshop**

Активация Windows 3
Чтобы активировать Windows, перейди

Рисунок В.3 – Існуючі програмні засоби що вирішують схожі задачі

Ядро сгортки



Активация Windows 4
Чтобы активировать Windows, перей
"Параметры".

Рисунок В.4 – Ядро сгортки

Обробка пікселя



Активация Windows
Чтобы активировать Windows, г

Рисунок В.5 – Обробка пікселя

Алгоритм збільшення різкості розмірами 3×3

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Алгоритм розмиття зображення

$$\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Активация Windows 6

Рисунок В.6 – Алгоритми збільшення різкості та розмиття зображення

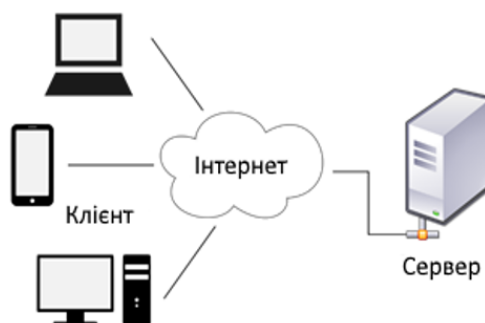
Використані алгоритми:

- Зміна контрастності
- Зміна колірного балансу
- Зміна яскравості
- Підвищення різкості і розмиття

Активация Windows 7
Чтобы активировать Windows, перейди

Рисунок В.7 – Використані алгоритми

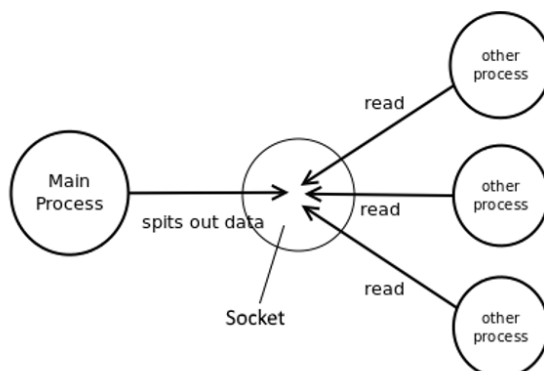
Вибір архітектури розроблюваної системи



Активация Windows 8

Рисунок В.8 – Вибір архітектури розроблюваної системи

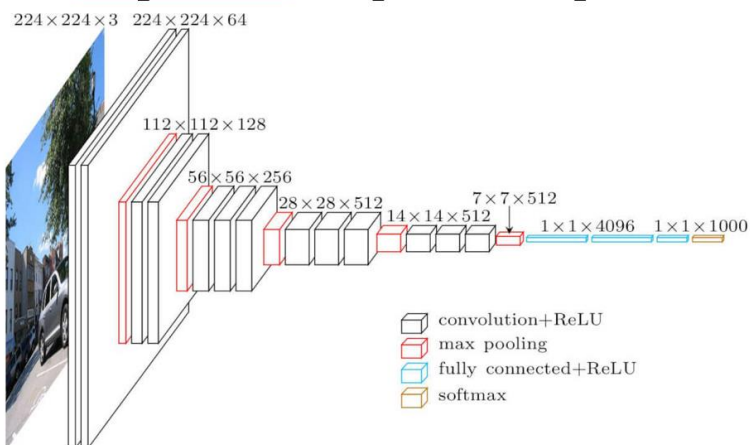
Загальна схема міжпроцесної взаємодії



Активация Windows 9
Чтобы активировать Windows, перейди

Рисунок В.9 – Загальна схема міжпроцесної взаємодії

VGG19-Network – типова архітектура згорткової нейронної мережі



Активация Windows 10
Чтобы активировать Windows, перейдите в
"Параметры".

Рисунок В.10 – VGG19-Network – типова архітектура ЗНМ

VGG19-Network – типова архітектура згорткової нейронної мережі

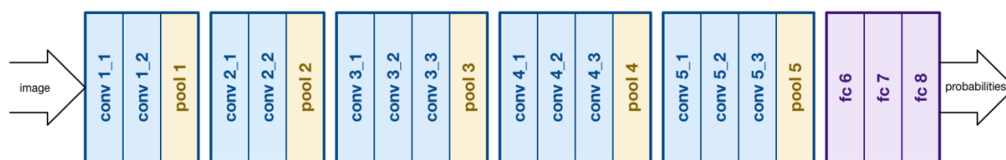


Рисунок В.11 – VGG19-Network – типова архітектура ЗНМ

Функції для обробки субдискретизаційних шарів

- $$\frac{\partial \lambda_{content}}{\partial F_{ij}^l} = \begin{cases} (F^l - P^l)_{ij}, & F_{ij}^l > 0 \\ 0, & F_{ij}^l < 0 \end{cases}$$
 - Похідна функції похибки для шару l ;
- $$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$
 - Функція втрат для окремого шару;
- $$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} ((F^l)^T (G^l - A^l))_{ji}, & F_{ij}^l > 0 \\ 0, & F_{ij}^l < 0 \end{cases}$$
 - Похідна функції похибки для шару l .

Рисунок В.12 – Функції

Архітектура розробленої системи

- Під час виконання програми використовуються класи `Form1`, `Form2`, `Form3`, `BrightnessContrast`, `ColorBalance`, `Filter`, які включають такі методи:
- а) `Brightness` - метод, що змінює яскравість поточної точки.
- б) `Contrast` - метод, що змінює контрастність поточної точки.
- в) `ColorBalance_R` - метод, що змінює колірний баланс точки по каналу R.
- г) `ColorBalance_G` - метод, що змінює колірний баланс точки по каналу G.
- д) `ColorBalance_B` - метод, що змінює колірний баланс точки по каналу B.
- е) `matrix_filtration` - метод, що виконує фільтрацію зображення на основі ядра згортки.

Активация Windows 11
Чтобы активировать Windows, перейдите

Рисунок В.13 – Архітектура розробленої системи

Діаграма прецедентів розробленої системи



Активация Windows 14
Чтобы активировать Windows, перейдите

Рисунок В.14 – Діаграма компонентів розробленої системи

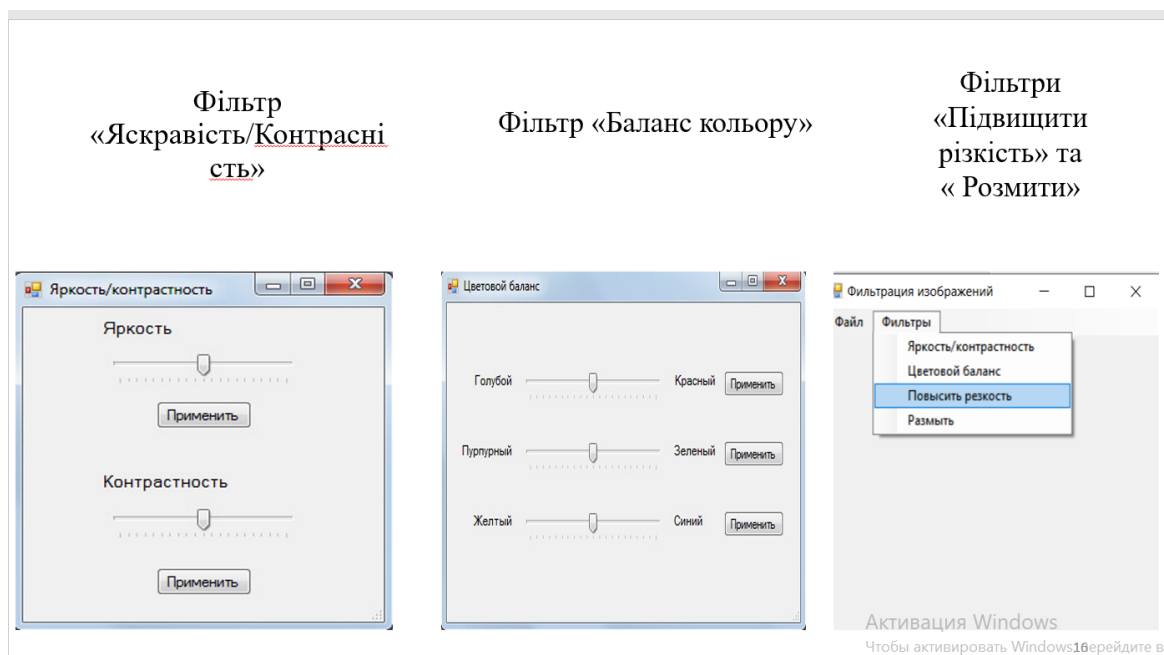


Рисунок В.15 – Функціонал програми

Вхідне зображення

Застосування фільтру «Розмити»

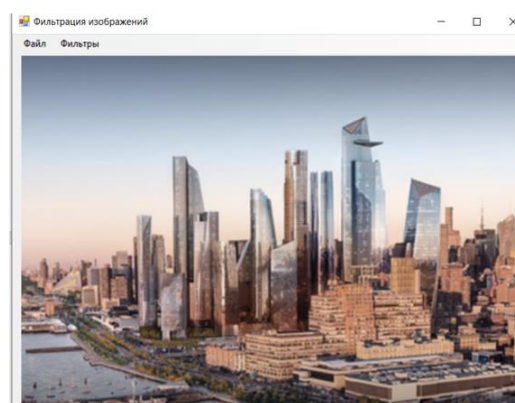
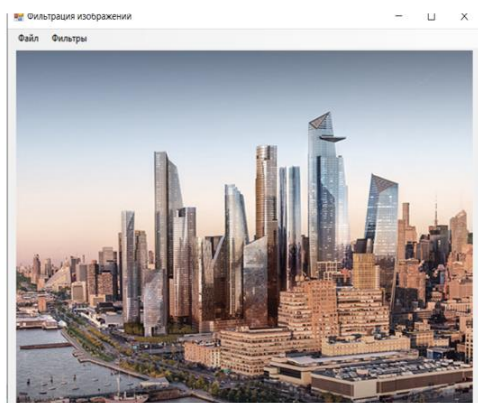


Рисунок В.16 – Застосування фільтрів

Застосування фільтрів «Яскравість/Контрасність»

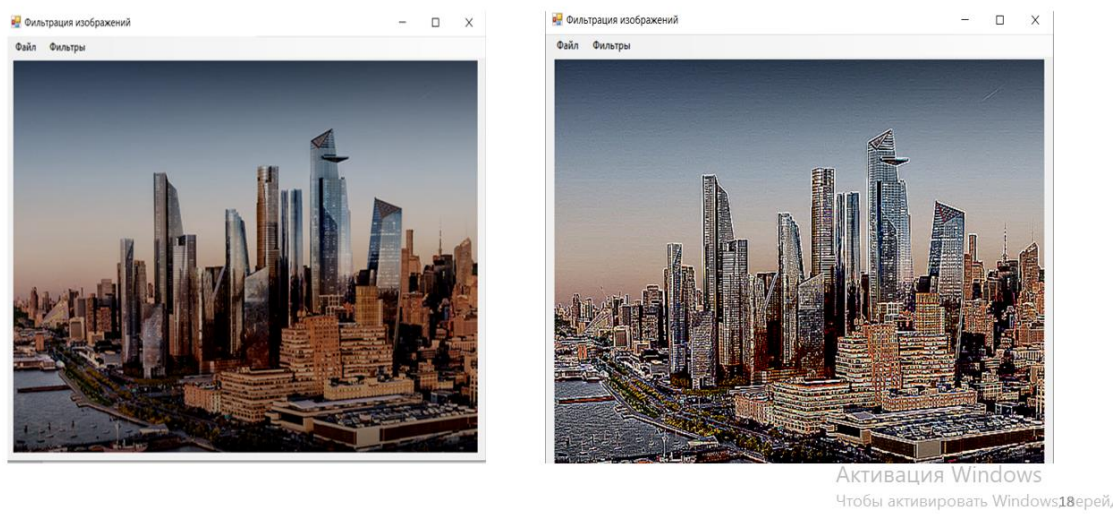


Рисунок В.17 – Застосування фільтрів

Висновки

Під час виконання дипломної роботи бакалавра, було розроблено програмну реалізацію алгоритму ядра сгортки та накладання фільтрів на зображення.

Для цього були поставлені і вирішені такі задачі:

- › аналіз існуючих рішень;
- › аналіз предметної області та алгоритму художнього стилю;
- › аналіз засобів та вибір засобів розробки (CSharp/Windows Forms та Visual Studio);
- › розробка програмної системи для накладання фільтрів на зображення.

Рисунок В.18 - Висновки

Шляхи подальшого розвитку

Під час аналізу розробленого програмного комплексу з'явилися наступні ідеї з його удосконалення як:

- Додавання нових фільтрів та стилей обробки зображення;
- Підвищити швидкість роботи системи;
- перенесення системи на Web-платформу та створення Web-сервісу задля універсальності системи.

Рисунок В.19 - Шляхи подальшого розвитку