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

Секція інформаційно-комунікаційних технологій

ВИПУСКНА РОБОТА

на тему:

«Інформаційна технологія генерації підробних зображень»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Кузіков Б.О.**

Студента групи ІН – 61 **Надточій Ю.О.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-61 спеціальності «Інформатика» денної форми навчання Надточій Юлії Олександрівни.

Тема: " **Інформаційна технологія генерації підробних зображень.** "

Затверджена наказом по СумДУ
№_____ от _____2020 р.

Зміст пояснювальної записки: 1) інформаційний огляд літературних джерел та постановка завдання; 2) розробка алгоритму та реалізація програми.

Дата видачі завдання "_____"_____2020 р.

Керівник випускної роботи _____ Кузіков Б.О.

Завдання прийняв до виконання _____ Надточій Ю.О.

РЕФЕРАТ

Записка: 41 стор., 23 рис., 1 додаток, 23 джерела.

Об’єкт дослідження – додаток swap-face.

Мета роботи - дослідження і розробка додатку з генерації підробних зображень на мові Python.

Результати - розроблена програма swap-face із використанням мови програмування Python, та її бібліотек. Програма здатна компілюватись на будь якій OS де э Python, або на Windows розповсюджуватись скомпільованим додатком

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ГЕНЕРАЦІЇ ПІДРОБНИХ ЗОБРАЖЕНЬ

ЗМІСТ

ВСТУП

Перегляд контенту, зокрема розважального, серед якого відео, фото, та gif-анімація давно став частиною нашого повсякденного життя. Все більше людей починають свій ранок з перегляду відео на Youtube, постів в Instagram та Twitter і протягом усього дня залишаються активними користувачами соціальних мереж і, як наслідок споживачами контенту. Відомі сайти, канали на Youtube, Rutube, Pikabu та блогери ведуть активну боротьбу за увагу користувачів, адже більше підписників та переглядів – це більше зароблених коштів. В епоху інформаційної доступності все важче привернути увагу та здивувати розбещеного, необмеженою кількість джерел для споживання контенту, користувача. Тому набувають популярності так звані технології створення підробного контенту. Підробні фотографія та відео створюються з різною метою: для жарту, підняття соціальних проблем або введення людей в оману. Та якою б не була мета, незмінним залишаться те, що подібні твориння користуються небувалою популярність серед інтернет користувачів.

Різке підвищення інтересу до технологій створення підробного контенту відбулося в кінці 2017 року після того як користувач з нікнеймом deepfakes (лягло в основу для назви технології) опублікував на сайті Reddit підробні відео з голлівудськими зірками. А також після появи в 2018 підробного відео з колишнім президентом США Бараком Обамою.

В кінці 2018 року автори «прогнозу песиміста» з Bloomberg назвали deepfake-відео однією з проблем, здатною серйозно вплинути на майбутнє, в тому числі і на глобальну політичну ситуацію в світі. У 2019 deepfakes-відео привернули увагу експертів з IT-сфери та чи не вперше за історію боротьби з фейками проблему стали намагатися активно вирішувати і вчені, і великі технологічні компанії. Нещодавно Facebook оголосив про конкурс, мета якого створити алгоритми, що дозволяють автоматично відрізняти фейковий відео від справжніх. Коли технологічна компанія витрачає мільйони на боротьбу з конкретною технологією, це говорить тільки про одне: сама технологія змушена

буде розвиватися і ставати кращою, щоб обійти найсучасніші способи захисту [1].

Хтось може сказати, що дана технологія приносить більше шкоди ніж користі. Але сама по собі технологія не може бути хорошою чи поганою - все залежить від того, як людина їх використовує. У мережі багато розважальних роликів з підробними обличчями. Є приклади використання deepfakes і в рекламі. Бекхем знявся в соціальному ролику про небезпеки малярії. Ті ж технології допомогли йому заговорити на 9 різних мовах: носії вимовляли текст, а штучний інтелект підбудовував його під артикуляцію футболіста. У майбутньому застосування даних технологій може вивести галузі реклами та кіноіндустрії на якісно новий рівень. Наприклад, можливість заміни акторів відразу під час перегляду реклами. Натиснути паузу і вибрати зі списку осіб того, хто подобається найбільше. Або замінити їх на близьких і знайомих. Це буде зовсім інший рівень емоційного зв'язку. Можливість зіграти головну роль у улюбленому фільми, або підібрати всіх акторів фільму на власний смак.

Метою дипломного проекту є розробка програми яка на основі двох фотографій буде створювати підробне зображення, замінюючи обличчя людей.

Завдання дослідження:

- провести аналіз сучасних інформаційних технологій генерації підробних зображень;
- проаналізувати системи розробки підробних зображень;
- розробити прогограму для заміни обличчя між двома фотографіями.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

В даній роботі під інформаційними технологіями генерації підробних зображень буде йти мова про створення підробного відео та фото шляхом заміни обличь людей на інші. У технології трансферу обличь поки що немає сталої назви, і в багатьох джерелах в межах однієї і тієї ж публікації її можуть називати і як face swap, і як deepfakes.

Термін «face swap», імовірно, прийшов в мову в 2000-х роках з появою в графічних редакторах функцій, які дозволяли користувачам трансформувати осіб на зображеннях (іноді вживали терміни «face replacement», «face morphing»), а також з наукових робіт. Але сфера застосування була дуже вузькою, тому в 2017 році, коли інтернет підірвали неправдиві відео за участю відомих артистів, технологію стали називати deepfake - по нікнейму користувача Reddit, який ці ролики публікував. І це слово стало широковідомим і змогло легко посунули термін, якому на той момент було півтора десятка років, і стало вживатися нарівні з ним [2].

Deepfake – методика синтезу зображення, заснована на штучному інтелекті. Вона використовується для з'єднання і накладення існуючих зображень і відео на вхідні зображення або відеоролики. У переважній більшості випадків, для створення таких відео використовують генеративно-змагальні нейромережі (GAN). Одна частина алгоритму вчиться на реальних фотографіях певного об'єкта і створює зображення, буквально «змагаючись» з другою частиною алгоритму, поки та не почне плутати копію з оригіналом [3][4].

Основною функцією face swap є пошук особи на зображенні донора (зображення з якого береться особа) і зображенні приймача (то куди буде передано особа з зображення донора) для їх обміну.

1.1 Історія розвитку технологій створення підробних зображень

Наукові дослідження, пов'язані з глибокими підробками, знаходяться переважно в області комп'ютерного зору, підгалузі інформатики. Першим значним проектом була програма Video Rewrite, опублікована в 1997 році, в якій змінювалися існуючі відеозаписи людини, що говорить, щоб зобразити її вимовляючою слова, що містяться в інший звуковій доріжці. Це була перша система, яка повністю автоматизувала цей вид маніпуляції обличчям, вона використовувала методи машинного навчання, для встановлення зв'язку між звуками, виробленими одним суб'єктом відео, і формою обличчя іншого суб'єкта.

Термін deepfakes почали використовувати, для даних технологій, після того, як користувач з однойменним нікнеймом в 2017 року виклав на Reddit створені ним підробні відео із залученими облич знаменитостей. Так як серед відео були провокаційні, що порушували особисті права людей технологія отримала шквал критики.

Сучасні академічні проекти були спрямовані на створення більш реалістичних відеороликів і вдосконалення методів. Програма Face2Face [5], опублікована в 2016 році, змінює реальному часі відеозапис людини так, щоб її обличчя наслідувало вирази обличчя іншої людини. В якості основного дослідницького вкладу проект перераховує перший метод для відтворення виразів обличчя в режимі реального часу з використанням камери, яка не фіксує глибину, що дозволяє виконувати цю техніку з використанням звичайних споживчих камер. Програма «Synthesizing Obama», опублікована в 2017 році, змінює відеозапис колишнього президента Барака Обами, щоб зобразити, як він вимовляє слова, що містяться в окремій звуковій доріжці. Проект перераховує в якості основного дослідницького вкладу свою фотореалістичну техніку для синтезу форм рота з аудіо.

У серпні 2018 року дослідники з Каліфорнійського університету в Берклі опублікували статтю про введення в гру штучного танцю, який може створити враження майстерності в танцях за допомогою штучного інтелекту. Цей проект розширює застосування глибоких підробок для всього тіла; попередні роботи були присвячені голові або частинам обличчя.

На даний момент йде активна розробка комерційних проектів, що дозволяють звичайним користувачам створювати власті підробні відео та фото. Серед таких: FakeApp, Faceswap, Zao. Японська компанія AI DataGrid зробила глибоку підробку всього тіла, здатну створити людину з нуля. Вони збираються застосовувати дану технологію в модній індустрії. У березні 2020 року було запущено перший мобільний додаток для створення відео зі знаменитостями – Deepfake Impressions.

1.2 Огляд існуючих рішень

На даний момент існують різні варіанти вирішення даного питання, як правило вони представляються онлайн сервісами, їх приклади:

- FakeApp - програма створена користувачем з Reddit під ім'ям deepfakes, яка дозволяє за допомогою методів навчання ШІ включати фотографії осіб на відео за допомогою алгоритму створення масок для заміни осіб на вхідному відео. Результат може бути неймовірним і дуже реалістичним, і розробник підтверджує, що він все ще працює над поліпшенням результатів, щоб зробити програмне забезпечення більш простим у використанні. В основі додатку лежать відкриті інструменти - бібліотеки Keras і TensorFlow і машинне навчання. Суттєвим недоліком програми є тє, що для отримання якісної підробки необхідно досить довго навчати нейромережу, а також для навчання необхідна бібліотека з набором зображень людини під різними кутами.

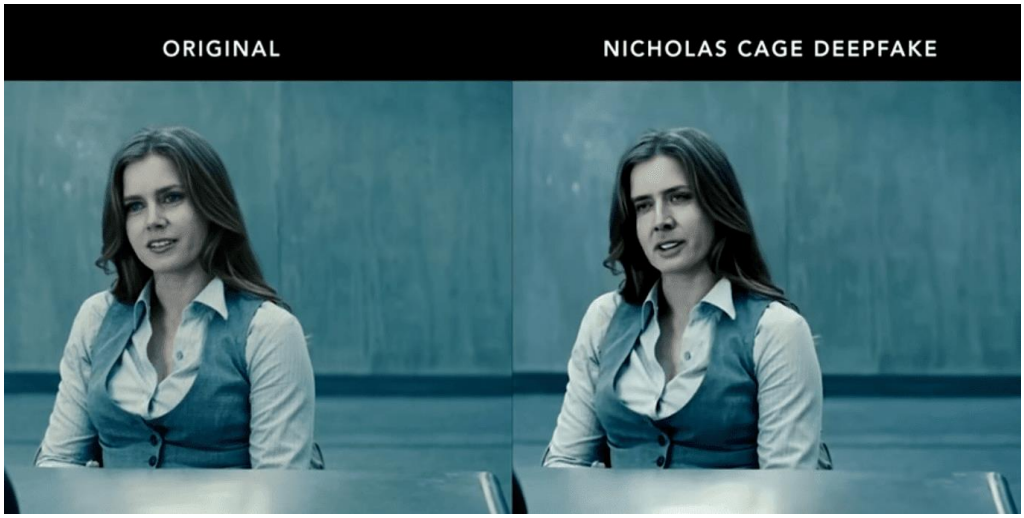

Рисунок 1.1 – Приклад роботи програми FakeApp [6]

- FaceApp - мобільний додаток для iOS і Android, розроблений російською компанією Wireless Lab, який використовує технологію нейронної мережі для автоматичної генерації дуже реалістичних перетворень осіб на фотографіях. Додаток може трансформувати обличчя, зробити його усміхненим, зістарити або омолодити і навіть змінити стать моделі.

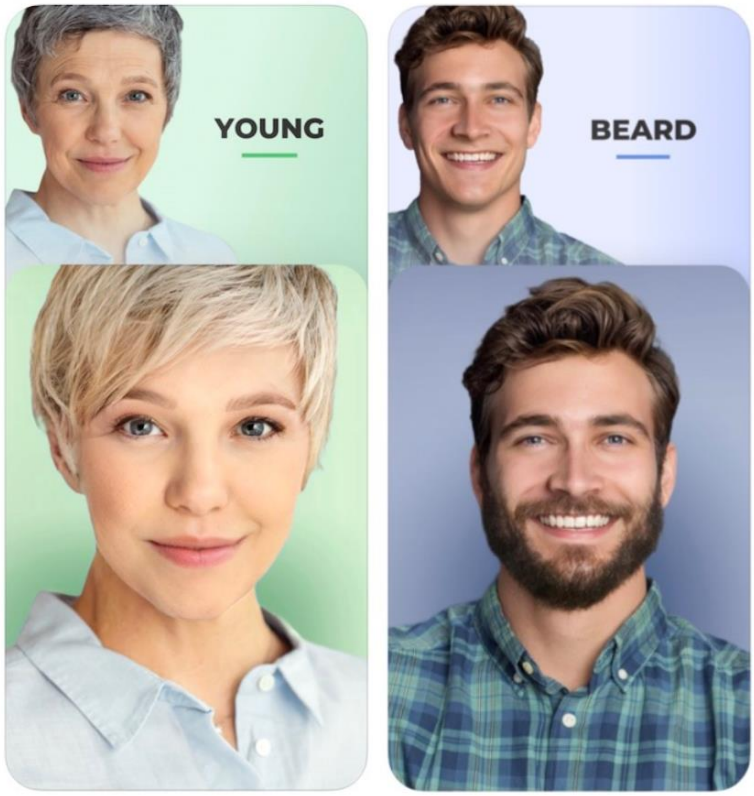

Рисунок 1.2 – Приклад роботи програми FaceApp [7]

- Doublicat – додаток для GIF-файлів розроблений українським стартапом RefaceAI. Основне його призначення полягає в тому, щоб замінювати в різних анімованих GIF-файлах обличчя персонажів на зображення користувача. За словами розробників програми, саме зображення видаляється з серверів відразу після його обробки. Однак при цьому зберігається уявлення рис обличчя. На відміну від інших додатків для заміни осіб, вони не використовують CGI або 3D-моделювання. Технологія заснована тільки на AI, в його основі лежить метод машинного навчання GAN (Generative Adversary Networks).

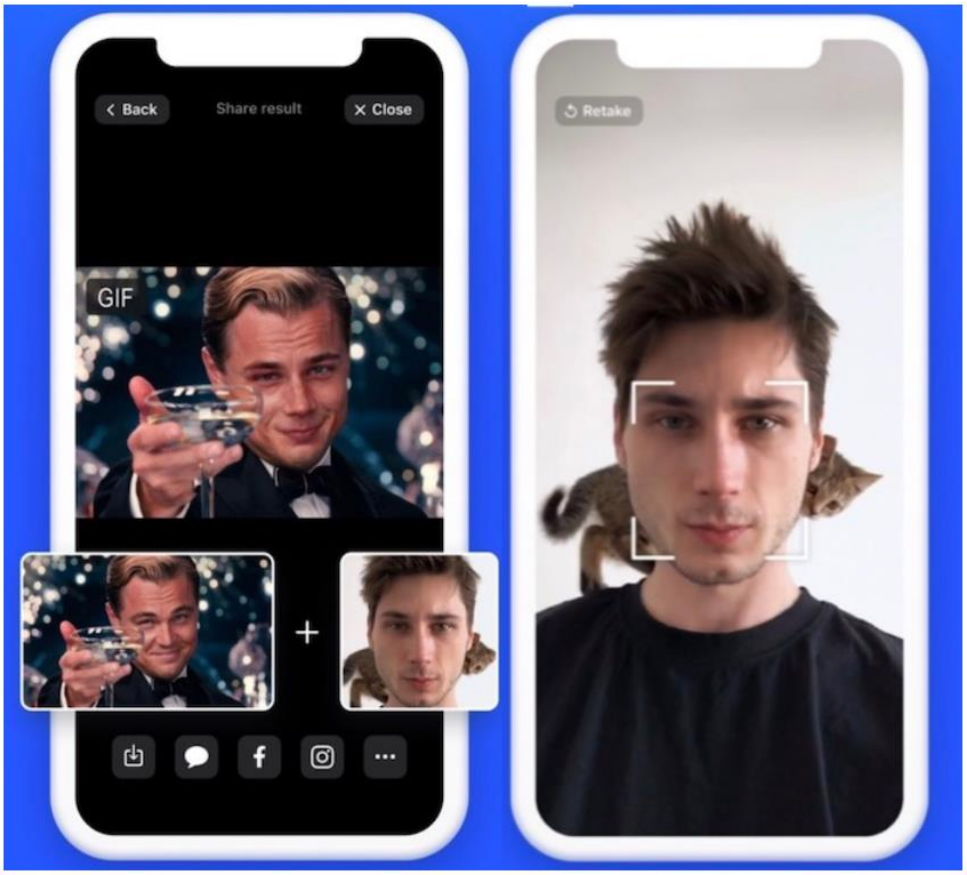

Рисунок 1.3 – Приклад роботи програми Doublicat [8]

● Zao - свіжа розробка з Китаю, яка засобами штучного інтелекту дозволяє вставити обличчя у фрагмент фільму, на обробку цієї інформації йде близько 10 секунд, але даний показник залежить від кількості користувачів.

Рисунок 1.4 – Приклад роботи програми Zao [9]

Таким чином проаналізувавши популярні існуючі рішення можна стверджувати що всі рішення є програмно складними, вони пропонують оживляти фотографії або підставляти фото під відео.

Для кожного такого рішення будуть потрібні великі обчислювальні потужності і виділені сервера, що і є їх мінусом.

1.3 Існуючі підходи до заміни обличь

Існуючі алгоритми заміни обличчя можна умовно поділити на три категорії: основані на заміні, основані на моделях та основані на навчанні.

Метод на основі заміни зазвичай замінює область обличчя в опорному зображенні на вхідну область обличчя, а потім застосовує деякі методи обробки зображень для посилення реалістичності синтезованого зображення. Бітук та ін. [10] побудували набір зображень обличь з різними виразами, формами та поставами, і алгоритм міг автоматично вибирати найбільш схоже опорне обличчя з набору зображень для заміни. Хоча цей метод може зменшити вплив відмінностей на вираз обличчя, форму та поставу, застосування алгоритму обмежене, оскільки користувачі не можуть самостійно вибирати улюблене

опорне зображення обличчя. Махаджан та ін. [11] використали маску зображення рис обличчя, а покрите маскою зображення потім було прикріплено до опорного зображення. Але обрана маска, яке може охоплювати лише риси обличчя, не здатне зберегти зморшки, текстури, кольори шкіри та деформації м'язів. Крім того, цей метод вимагає від користувачів виконання інтерактивних інструкцій під час роботи, що значно знижує практичність роботи алгоритму.

У модельному підході [12] використовується двовимірна або тривимірна параметрична модель характеристик для відображення обличчя людини, параметри та особливості добре відрегульовані для вхідного зображення. Потім реконструкція обличчя виконується на еталонному зображенні на основі результату коригування параметрів моделі. Рання робота, яку представили Бланц, Шербаум та ін. [13] використовувала 3D-модель для оцінки форми обличчя та постави, що поліпшило недолік незадовільних показників синтезу завдяки освітленості та перспективі. Однак для отримання кращого результату алгоритм вимагає введення та ініціалізації 3D-моделі вручну, що, безсумнівно, ускладнює роботу та отримання даних цим алгоритмом. Ван та ін. [14] запропонували алгоритм, заснований на активній уявній моделі (AAM). Використовуючи добре навчену AAM, заміна обличчя реалізується в два етапи: підбір моделі та складових компонент. Але для цього методу потрібно вказати ROI для обличчя вручну та певну кількість зображень обличчя для навчання моделей. Лін та ін. [15] представив метод побудови тривимірної моделі на основі лобового зображення обличчя для вирішення різних перспектив опорного та вхідного зображення. Але реконструйована модель не відображає особливостей оригінального обличчя і вимагає занадто багато часу для обчислення.

У більшості алгоритмів, що ґрунтуються на навчанні, опорне зображення перетворюється на синтезоване шляхом навчання генеративної нейронної мережі, яка містить інформацію вхідного зображення. Коршунова та ін. [16] запропонували модель, засновану на згортковій нейронній мережі, яка може змінити опорну зображення на вхідне, зберігаючи поставу, вираз і освітленість

першого. Хоча цей метод [16] має деякі переваги у реалістичності, він потребує великої кількості навчальних даних та великої кількості обчислень, а навчена мережа працює лише для однієї єдиної людини. Крім того, для отримання генерації синтетичних зображень обличчя також може бути використана технологія gan (генеративні змагальні мережі) [17]. Цей метод замінює латентне представлення місця на обличчі, а потім реконструює все зображення обличчя за допомогою розділеної моделі gan. Однак для цієї моделі потрібно створити великий набір зображень обличчя з мітками розбору рівня пікселів, при цьому трохи погіршується якість синтезованого зображення.

Перш за все, підхід на основі заміни простий і швидкий, але чутливий до зміни постави та перспективи. Модельний метод може ефективно вирішити перспективну проблему; проте зазвичай потрібно збирати тривимірні дані обличчя, і надійність - це не те, що потрібно задовольнити. Підхід, заснований на навчанні, може створити цілком реальний і природний синтетичний образ обличчя, вимагаючи, як правило, великої кількості даних для навчання та накладнає більше обмежень щдо вхідних та опорних зображень. На основі всебічного врахування характеристик вищезгаданих трьох методів для реалізації було обрано алгоритм оснований на заміні.

1.4 Алгоритм створення підробних зображень на основі заміни

Суть алгоритма на основі заміни полягає в тому, щоб замінити область обличчя в цільовому зображенні на обличчя вхідного зображення. В запропонованому алгоритмі для досягнення кращих результатів застосовуються інші широко використовувані алгоритми, такі як виявлення обличчя на основі гістограми орієнтованих градієнтів, виявлення орієнтирів обличчя, сегментація області обличчя, викривлення обличчя та безшовне клонування. Далі всі ці методи будуть описані більш детально.

1.4.1 Пошук обличчя

Для пошуку на зображеннях обличчя буде використовуватися зв’язка гістограми спрямованих градієнтів (HOG) та методу опорних векторів (SVM).

Для обчислення HOG-структури розглядається кожен окремий піксель і його оточення. Виділяється на скільки кожен піксель темніший ніж пікселі, що примикають до нього. Повторюється цей процес для кожного пікселя, в результаті кожен піксель замінюється стрілкою (вектором). Стрілка є градієнтом, який показує потік від світла до темряви по всьому зображенню. Далі зображення розбивається на квадрати по 16×16 пікселів, і вказується головний напрямок пікселів, що входять в цей квадрат. Напрямок градієнта в квадраті шукається шляхом підсумовування всіх напрямків градієнтів пікселів в ньому. Таким чином, більш сильні градієнти вносять більшу вагу, а ефекти невеликих випадкових градієнтів, що утворилися через шум, зменшуються. Ця гістограма дає картину домінантної орієнтації цього осередку. Спільність таких градієнтів створює HOG структуру. Для знаходження особи на зображеннях потрібно знаходити HOG-структури, схожі на відому HOG-структуру, отриману з групи осіб. Для цього використовується класифікація дескрипторів за допомогою системи навчання з учителем SVM (Метод опорних векторів). SVM утворює 2 кластера, розділених кордоном, і для кожного ковзного вікна вираховується HOG-структура, і якщо ця HOG структура потрапить в кластер шуканих об'єктів, то програма знайшла об'єкт, а якщо в інший кластер, то програма просто перевіряє далі. Приклад HOG-структури, зображений на рисунку 1.5.

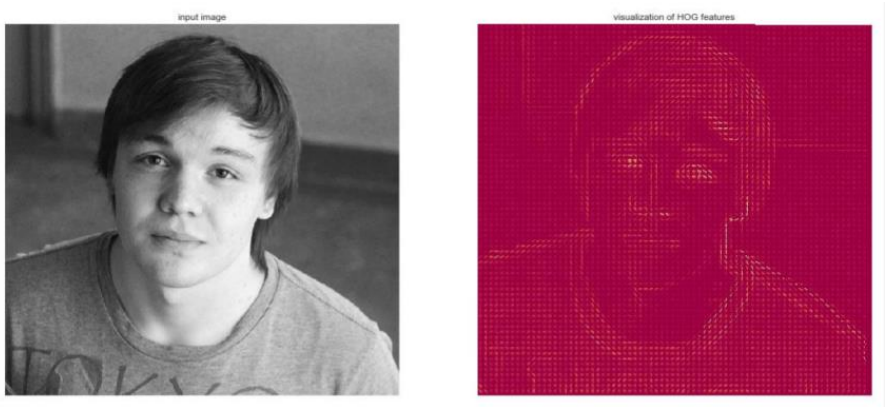

Рисунок 1.5 – Приклад роботи HOG-дескриптора.

1.4.2 Виділення лицьових маркерів

Модель активної форми – це статистична модель форми об'єкта, яка ітеративно деформується, щоб відповідати об'єкту присутньому на новому зображенні. Форма обмежена моделлю розподілу точок статистичної форми моделі, щоб модель могла змінюватися тільки в межах розмічених прикладів з навчальної вибірки. Форму об'єкта представляє безліч точок (контрольованих формою моделі), кожна з яких відповідає за своє місце на об’єкті, у нашому випадку – на обличчі людини. Перед використанням модель повинна бути навчена на безлічі заздалегідь розмічених зображень. Кожна мітка має свій номер і визначає характерну точку, яку повинна буде знаходити модель під час адаптації до нового зображення. Приклад подібної розмітки показаний на рисунку 1.6.

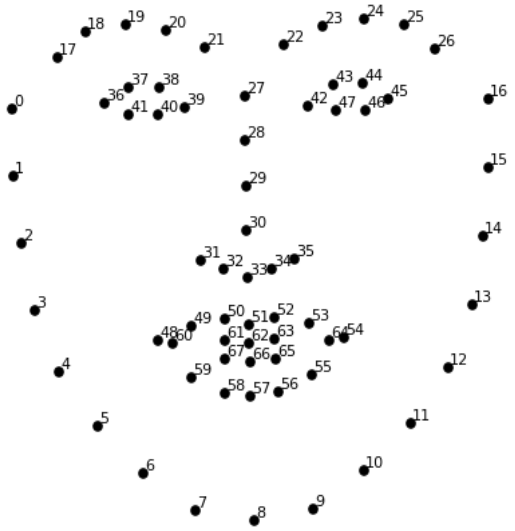

Рисунок 1.6 – Модель активної форми обличчя людини на 68 міток [18]

У представленому прикладі на зображенні відзначені 68 міток, що утворюють форму моделі активного зовнішнього вигляду. Ця форма позначає зовнішній контур обличчя, контури рота, очей, носа, брів. Даний характер розмітки дозволяє в подальшому визначити різні параметри особи за її зображенню, які можуть бути використані для подальшої обробки іншими алгоритмами.

Мета алгоритму моделі активної форми - зіставити модель з новим зображенням. Алгоритм складається з двох почергових дій:

- Пошук на зображенні навколо кожної точки кращій позиції для даної точки;
- Оновлення параметрів моделі шляхом максимальної відповідності умовам з новими знайденими позиціями.

Щоб знайти кращу позицію для кожної точки можна шукати чіткі краї, а можна поєднати статистичну модель з тим, що очікується для даної точки. Для обчислення кращої позиції для кожного орієнтира точки оригінальний метод передбачає використання відстані Махаланобіса, відстань у евклідовому просторі, що узагальнює поняття евклідової відстані. Визначається формулою:

$$d(X, Y; S) = \sqrt{(X - Y)^T S^{-1} (X - Y)}$$

де $X = (x_1 \dots x_N)$ – багатомірний вектор, від якого береться відстань;
$Y = (y_1 \dots y_N)$ – середні значення множини, до якої рахується відстань;
S – матриця коваріацій множини.

1.4.3 Тріангуляція Делоне

На множині точок на площині задана тріангуляція, якщо деякі пари точок з'єднані ребром, будь-яка кінцева грань в отриманому графі утворює трикутник, ребра не перетинаються, і граф максимальний за кількістю ребер.

Тріангуляція Делоне - тріангуляція для заданої множини точок на площині, при якій для будь-якого трикутника всі крапки з за винятком точок, які є його вершинами, лежать поза окружністю, описаною навколо трикутника. Для заданої множини точок, в якому ніякі 4 точки не перебувають на одному колі, існує рівно одна тріангуляція Делоне. Тріангуляція Делоне для двовимірного простору зображена на рисунку 1.7.

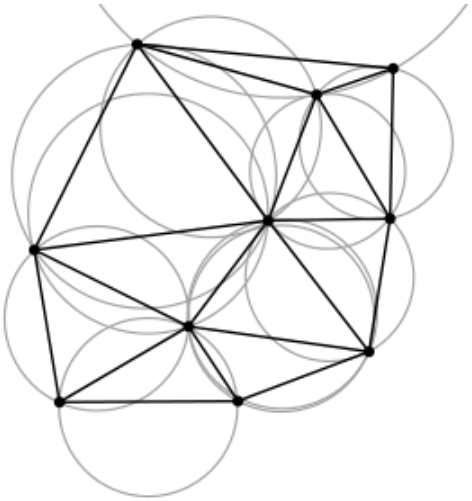

Рисунок 1.7 – Тріангуляція Делоне для двовимірного простору [19]

Існує багато алгоритмів для знаходження тріангуляції Делоне для набору точок. Найбільш очевидний (але не найефективніший) - почати з будь-якої тріангуляції і перевірити, чи містить окружність будь-якого трикутника іншу точку. Якщо це так, необхідно перевернути краї і продовжувати, поки не з'являться трикутники, коло яких містить точку.

Не можна просто вирізати обличчя з вхідного зображення і помістити його в кінцеве зображення, так як вони мають різний розмір і перспективу. Також не можна відразу змінити його розмір і перспективу, тому що обличчя втратить початкові пропорції. Саме тому на основі моделі активної форми будується тріангуляція Делоне для множини точок на площині. При розділенні обличчя на трикутники, можна просто поміняти місцями кожен трикутник, і таким чином він збереже пропорції, а також буде відповідати виразами нового обличчя, таким як, посмішка, закриті очі або відкритий рот.

1.4.4 Афінні перетворення

Афінні перетворення - це перетворення, де зберігаються точки, прямі та площини. Крім того, набори паралельних ліній залишаються паралельними після перетворення. Афінні перетворення не обов'язково зберігають кути між лініями або відстані між точками, хоча зберігаються відносини відстаней між точками, що лежать на одній прямій. Це перетворення масштабує, виконує скіс, поворот і

зрушення координат для вершин шару. Для цього методу потрібно, як мінімум, три зв'язку зсуву.

Рішення завдання перетворення зводиться до знаходження коефіцієнтів системи рівнянь. В разі афінного перетворення, система рівнянь виглядає наступним чином:

$$\begin{cases} x' = a_0 + a_1 x + a_2 y \\ y' = b_0 + b_1 x + b_2 y \end{cases}$$

де:

x,y - координати в початковій системі координат (відомі); x',y' - координати в кінцевій системі координат (відомі); $a_0, a_1, a_2, b_0, b_1, b_2$ - невідомі коефіцієнти.

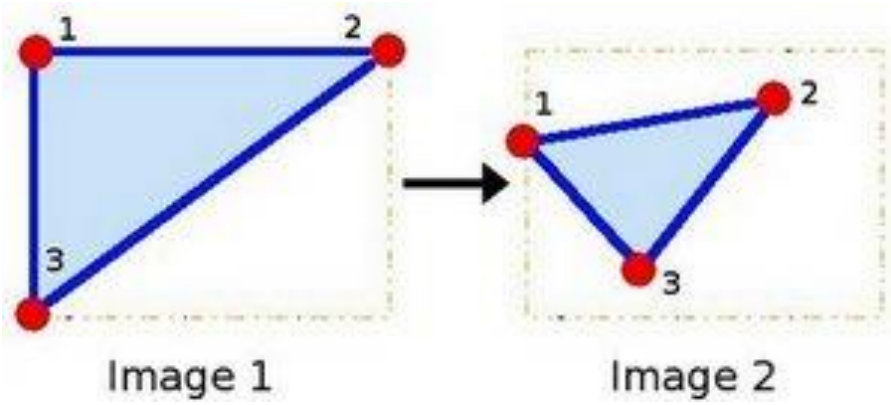

Рисунок 1.8 – Афінне перетворення трикутника [20]

1.4.5 Безшовне клонування

Після перенесення обличчя, зображення виглядає неприродньо, межі вставленого зображення помітно через різницю в освітленості та відтінку шкіри. Для вирішення даної проблеми буде використовуватися метод безшовного клонування. Даний метод працює з градієнтами зображень замість інтенсивностей, що допоможе досягти більш реалістичних результатів.

Після безшовного клонування інтенсивність результуючого зображення в замаскованій області не співпадає з інтенсивністю вхідною інтенсивністю в тій же області. Замість цього градієнт результуючого зображення в межах маски приблизно такий же, як градієнт вхідної області в цих межах. Крім того,

інтенсивність результуючого зображення на кордоні замаскованої області така ж, як інтенсивність цільового зображення. Це досягається шляхом вирішення рівняння Пуассона.

Рівняння Пуассона — неоднорідне еліптичне рівняння в частинних похідних другого порядку, загального виду:

$$\Delta\varphi = f,$$

де Δ— оператор Лапласа, φ — невідома функція, f — довільна функція, що не залежить від невідомої [21].

ПОСТАНОВКА ЗАДАЧІ

Нехай дано два зображення з обличчями людей: одне з них вхідне– з якою буде братися обличчя для заміни; інше цільове – куди буде перенесене обличчя з вхідного зображення.

Необхідно побудувати функціонал, який би дозволяв перенести обличчя з вхідного зображення на цільове.

Для вирішення поставленого завдання необхідно розпізнати обличчя та виділити лицьові маркери на вхідному та цільовому зображенні. Вирізати опуклу маску обличчя з вхідного зображення. Зробити тріангуляцію обох зображень, афінними перетвореннями привести обличчя вхідного зображення до цільового. Вставити обличчя з вихідного зображення на цільове з застосуванням безшовного клонування.

У результаті має бути створений десктопний додаток, в який можна буде завантажити дві фотографії, цільову та вхідну і на виході роботи програми отримати підробне зображення, в якому обличчя з вхідного зображення буде перенесене на цільове. Повинна бути можливість збереження створеного зображення. Додаток має бути максимально простим і легким у використанні.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ СТВОРЕННЯ ПІДРОБНИХ ЗОБРАЖЕНЬ

2.1 Вибір мови програмування

У якості мови для розроблення додатку була обрана мова Python.

Python - об'єктно-орієнтована мова загального призначення, яка розроблена з метою підвищення продуктивності програміста.

На Python можна писати будь-якого роду розширення, використовувати в іграх як мову для штучного інтелекту, активно вбудовувати в інші додатки. Мова Python вже стала стандартом в проектуванні мов, і багато нових мов створюються орієнтуючись на Python і використовують його конструкції.

Основні переваги Python:

- Легко читати та вивчати: Python - це проста мова для читання та вивчення. У ній немає складних синтаксисів, як в інших мовах високого рівня, таких як C або C ++. Завдяки меншій складності Python дозволяє чіткіше мислити та зосереджуватись на побудові логіки.
- Підвищення продуктивності: простота мови та великі бібліотеки роблять програмістів більш продуктивними, ніж такі мови, як Java та C ++. При написанні меншої кількості коду можна виконати більше завдань.
- Інтеграція з іншими мовами: у Python є такі бібліотеки, як Cython та Jython, які дозволяють інтегруватись з іншими мовами, такими як C, C ++ та Java, щоб увімкнути кросплатформенний розвиток. Це одна з головних переваг Python, оскільки жодна мова не є досконалою, а іноді для розвитку потрібні різноманітні мовні функції, які неможливо отримати однією мовою.
- Об’єктно-орієнтована: ця мова підтримує як процедурні, так і об'єктно-орієнтовані парадигми програмування. Хоча функції допомагають нам у використанні коду, класи та об'єкти дозволяють нам моделювати реальний світ. Клас дозволяє інкапсулювати дані та функції в одне ціле.

- Безкоштовно та з відкритим кодом: Python постачається за затвердженою OSI ліцензією з відкритим кодом. Це робить його вільним у використанні та розповсюдженні. Ви можете завантажити вихідний код, змінити його і навіть поширити свою версію Python.
- Переносність: у багатьох мовах, таких як C / C ++, вам потрібно змінити код, щоб запустити програму на різних платформах, але не в Python. Ви пишете лише один раз і запускаєте його де завгодно. Однак слід бути обережним, щоб не включати будь -які функції, що залежать від системи .
- Розширені бібліотеки: Python завантажується з великою бібліотекою, і він містить код для різних цілей, таких як регулярні вирази, створення документації, тестування одиниць, веб-браузери, бази даних, електронна пошта, маніпулювання зображеннями тощо. Отже, не потрібно писати повний код для цього вручну.

Python продовжує розвиватись навіть зараз і не планує згасати ще довго. У недавніх численних аналізах ii рейтингах мова займає лідируючі позиції. За статистикою DOU мова перебуває на п'ятому місці та займає третю позицію серед мов орієнтованих на веб-технології.

2.2 Використання бібліотек

Для програмної реалізації будуть використовуватись такі бібліотеки:

NumPy - модуль притримує open-source ідеї, він надає математичні та числові операції, і надає їх у вигляді пре-скомпільованих, швидких функцій. Дані функції об'єднуються в високорівневі пакети. Даний модуль здатний зрівнятися з функціоналом MatLab. Для NumPy існує велика онлайн база для допомоги, вона включає гайди і туторіали. Даний модуль значно розширює масиви, рядки, стовпці. [22]

Для того щоб розпізнати обличчя, буде використовуватися система контрольних точок, це буде робити два модуля OpenCV + DLib. Метою

розстановки точок на фотографії не являються фінальної задачі, це тільки інструмент. За допомогою контрольних точок відслідковуються рухові одиниці.

Таким чином OpenCV - бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. В даному проекті застосовується для роботи з зображеннями.

Dlib - містить алгоритми машинного навчання, так само штучного інтелекту і різні інструменти для роботи із зображеннями обличь. На dlib поширюється тип ліцензії Boost, що дозволяє використовувати її в комерційних проектах, це сприяло її широкому розповсюдженню. Багато бібліотек і фреймворків, заточені під роботу із зображеннями обличь, використовують dlib [23].

PyQt5 - це набір Python бібліотек для створення графічного інтерфейсу на базі платформи Qt5 від компанії Digia. Бібліотека Qt є однією з найпотужніших бібліотек GUI (графічного інтерфейсу користувача). PyQt5 реалізований у вигляді набору python-модулів. Ця бібліотека має понад 620 класів і 6000 функцій і методів. Це мультиплатформенна бібліотека, яка працює на всіх основних операційних системах, в тому числі Unix, Windows і Mac OS.

PyQt також включає в себе Qt Designer (Qt Creator) - дизайнер графічного інтерфейсу користувача. Програма генерує Python код з файлів, створених в Qt Designer. Це робить PyQt дуже корисним інструментом для швидкого прототипування. Крім того, можна додавати нові графічні елементи управління, написані на Python, в Qt Designer.

2.3 Опис програмної реалізації

2.3.1 Загальний алгоритм реалізації методу

- Виявлення обличчя

Для початку на цільовому та вхідному зображеннях за допомогою гістограми орієнтованих градієнтів (HOG) необхідно знайти обличчя.

а) б)

Рисунок 2.1 – Виявлення обличчя за допомогою гістограми орієнтованих градієнтів: а) цільове зображення; б) вхідне зображення

- Виявлення лицьових орієнтирів

На даному етапі за допомогою моделі активної форми, на кожному з зображень знаходтся 68 унікальних лицьових орієнтирів (визначні місця у зображенні людського обличчя).

а) б)

Рисунок 2.2 – Виявлення лицьових орієнтирів: а) цільове зображення; б) вхідне зображення

- Пошук меж обличчя

Опуклий корпус - це сукупність точок, визначених як найменший опуклий багатокутник, який охоплює всі точки набору. Це означає, що для заданого набору точок опуклий корпус є підмножиною цих точок таким, що всі задані точки знаходяться всередині підмножини. Щоб знайти межу обличчя в зображенні, необхідно знайти опуклий корпус для отриманиї 68 лицьових орієнтирів вхідного зображення.

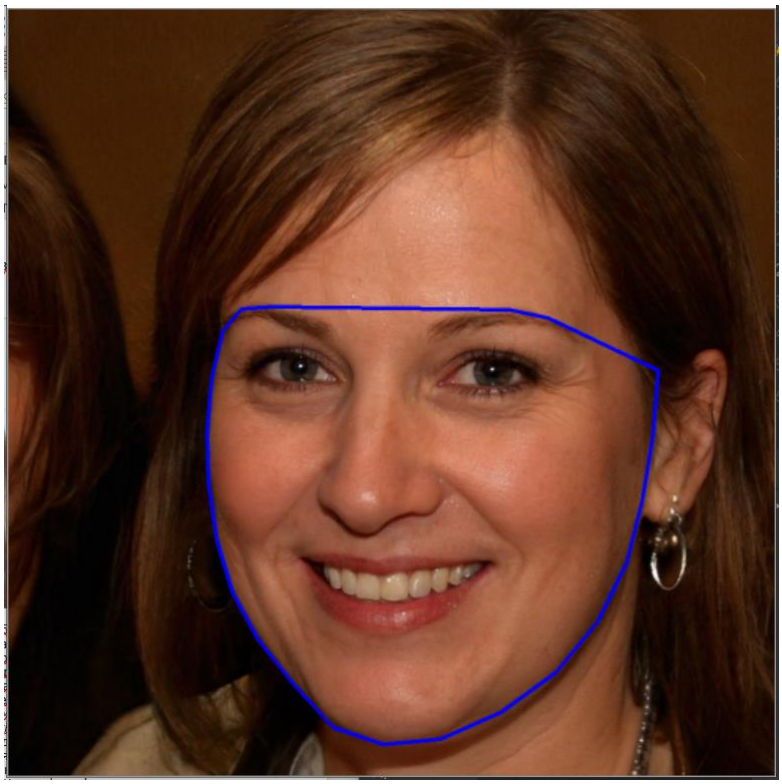

Рисунок 2.3 – Межі обличчя вхідного зображення

- Приведення вхідного обличчя до цільового

Для початку необхідно за допомогою тріангуляції Делоне розбити обличчя цільового та вхідного зображення на трикутники.

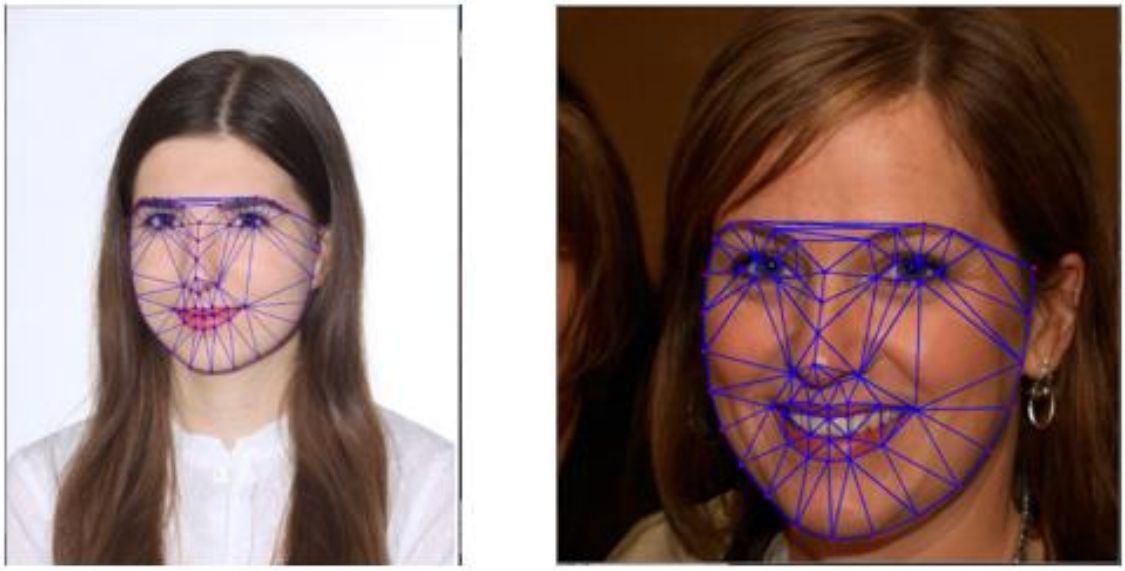

Рисунок 2.4 – Тріангуляція Делоне: а) цільове зображення; б) вхідне зображення

За допомогою афінних перетворень необхідно кожний трикутник вхідного зображення перетворити на відповідний трикутник цільового зображення і об’єднати всі отримані трикутники в нове зображення.

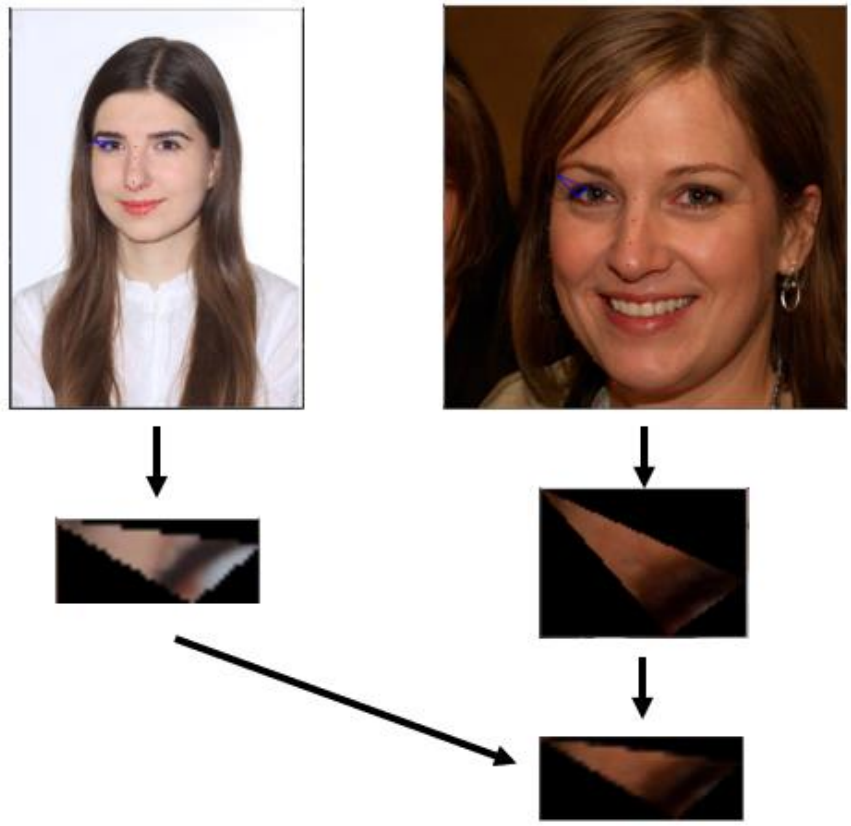

Рисунок 2.5 – Приклад афінного перетворення для зображень

Рисунок 2.6 – Результат афінного перетворення для всього обличчя: а) цільове зображення; б) обличчя отримане в результаті перетворень; в) вхідне зображення

- Перенесення обличчя

Для підвищення реалістичності зображення після перенесення обличчя необхідно застосувати безшовне клонування, яке ґрунтується на вирішенні рівняння Пуассона.

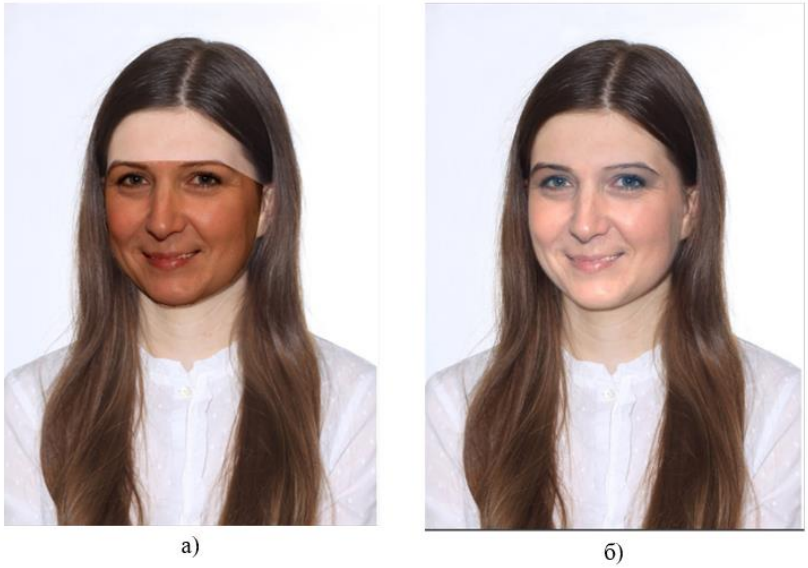

Рисунок 2.7 – Застосування безшовного клонування обличчя: а) зображення без використання безшовного клонування; б) зображення з застосуванням безшовного клонування

2.3.2 Інтерфейс програми

Для реалізації технології створення підробних зображень було розроблено десктопний додаток. Простий і зрозумілий інтерфейс дозволить користувачам різного рівня знань з легкістю користуватися програмою. Інтерфейс програми можна побачити на рис. 2.8 – 2.12.

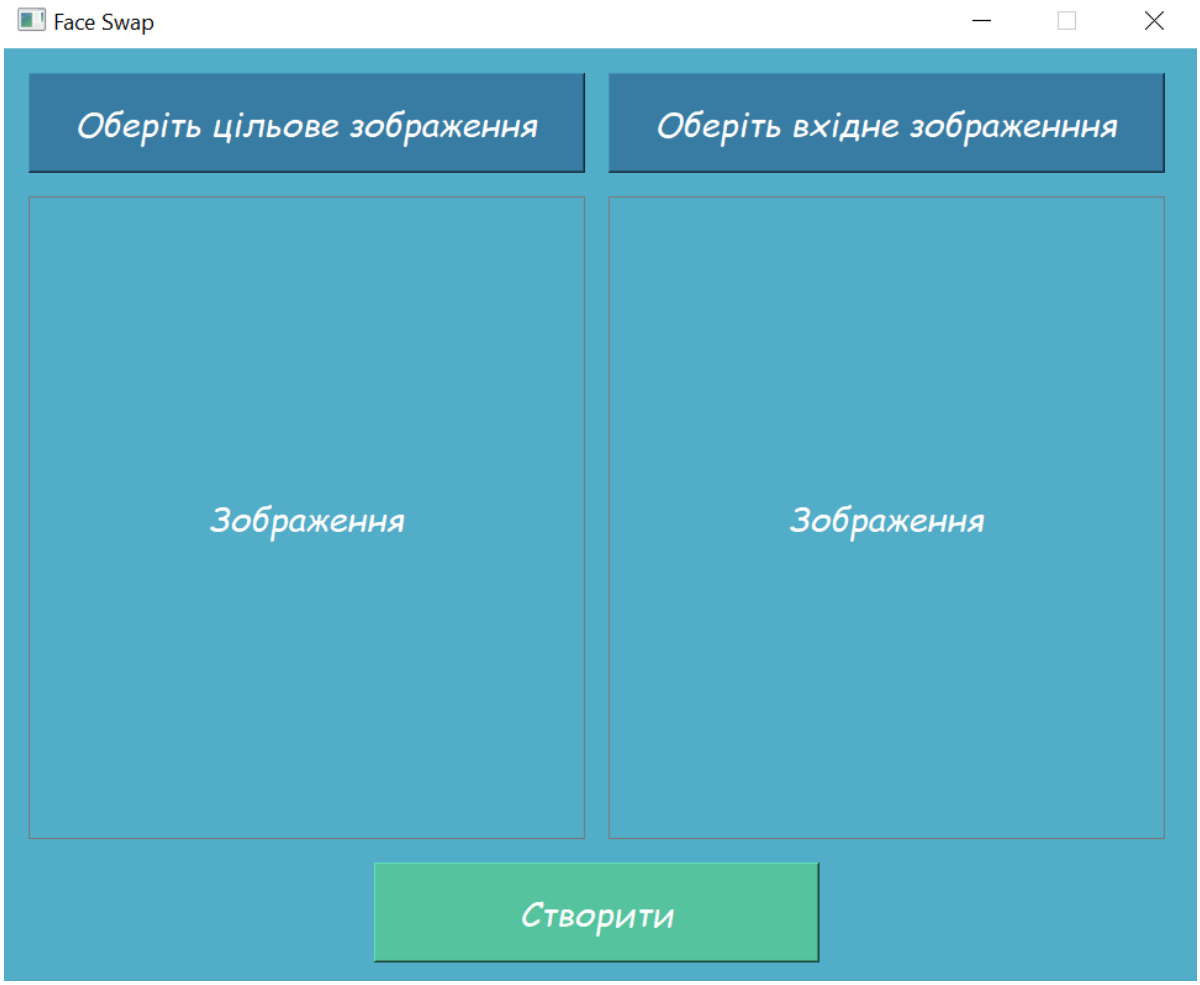

Рисунок 2.8 – Головне вікно програми

На рис. 2.8 зображене головне вікно програми. На даному вікні знаходяться кнопки для вибору цільового та вхідного зображення. При натисканні на дані кнопки відкривається діалогове вікно у якому можна обрати необхідні зображення (рис. 2.9).

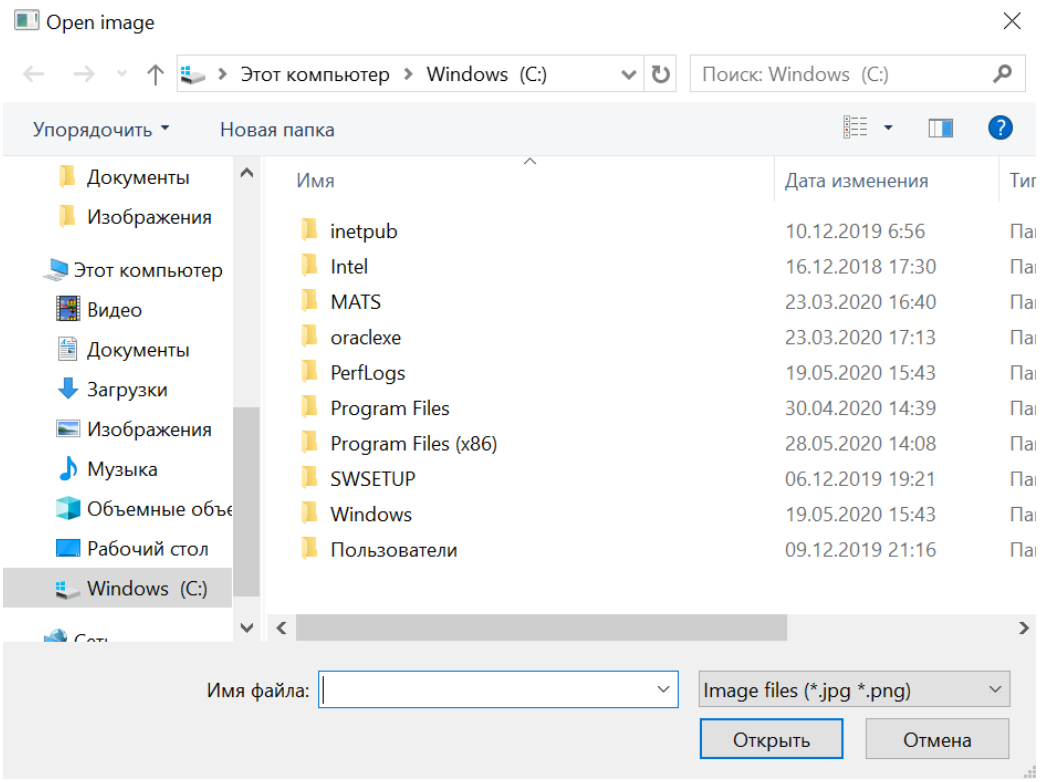

Рисунок 2.9 – Діалогове вікно для вибору зображення

Після вибору зображень вони відображаються у додатку (рис. 2.10).

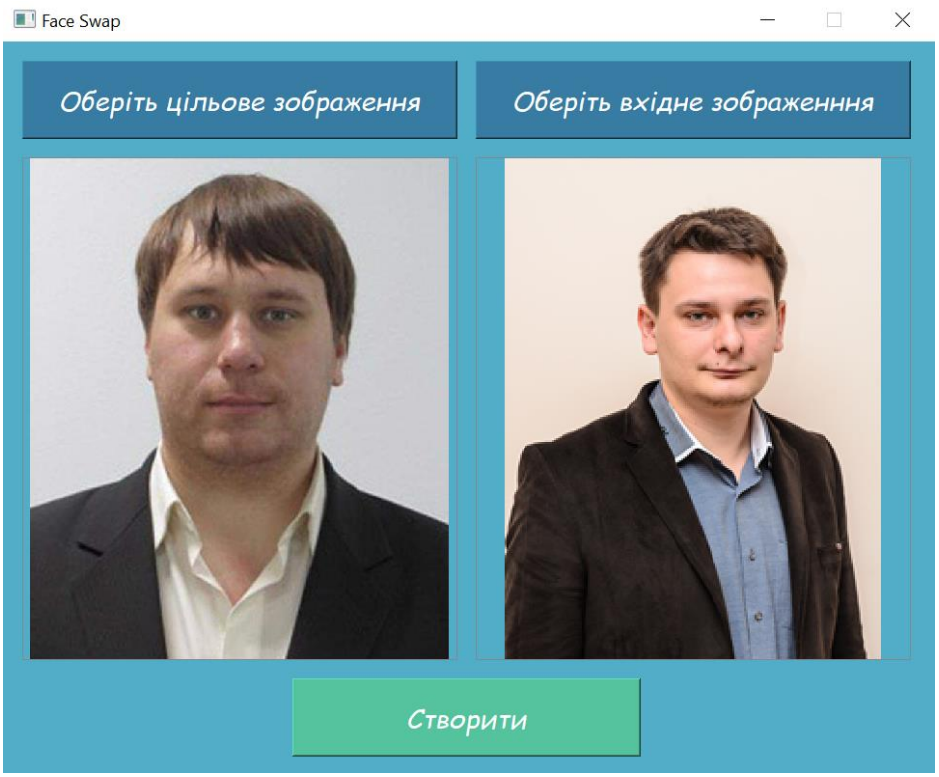

Рисунок 2.10 – Головне вікно після вибору зображень

Після вибору зображень, для запуску процесу створення підробного зображення необхідно натиснути на кнопку створення нового зображення. Результат роботи програми відобразиться у новому вікні (рис. 2.11).

Рисунок 2.11 – Вікно з результатом роботи програми

Якщо необхідно зберегти результат роботи, після натискання на кнопку зберігання відкриється діалогове вікно для збереження зображення (рис. 2.12). Необхідно вибрати місце та назву для збереження зображення.

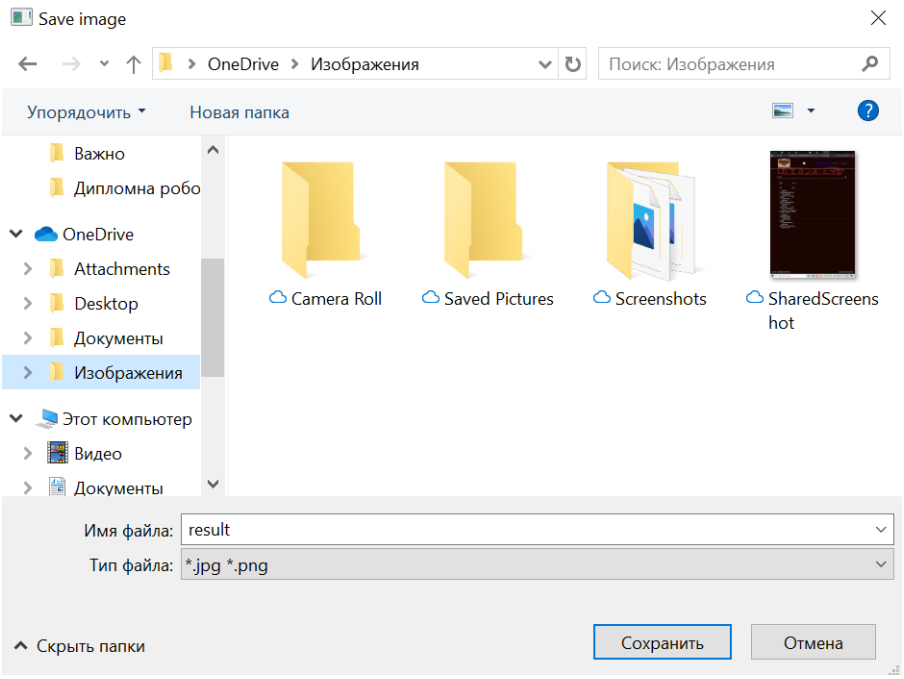

Рисунок 2.12– Діалогове вікно для збереження зображення

2.4 Результат роботи програми

На вхід програмі подається два зображення вхідне та цільове. Вхідне зображення – те з я кого буде братися обличчя для переносу. Цільове зображення – те на яке ми будемо переміщати обличчя з вхідного зображення. Результатом роботи програми буде кінцеве зображення, яке представляє собою цільове зображення зі зміненим обличчям. Код програми наведений у додатку А.

Для демонстрації працездатності програми наведемо декілька прикладів:

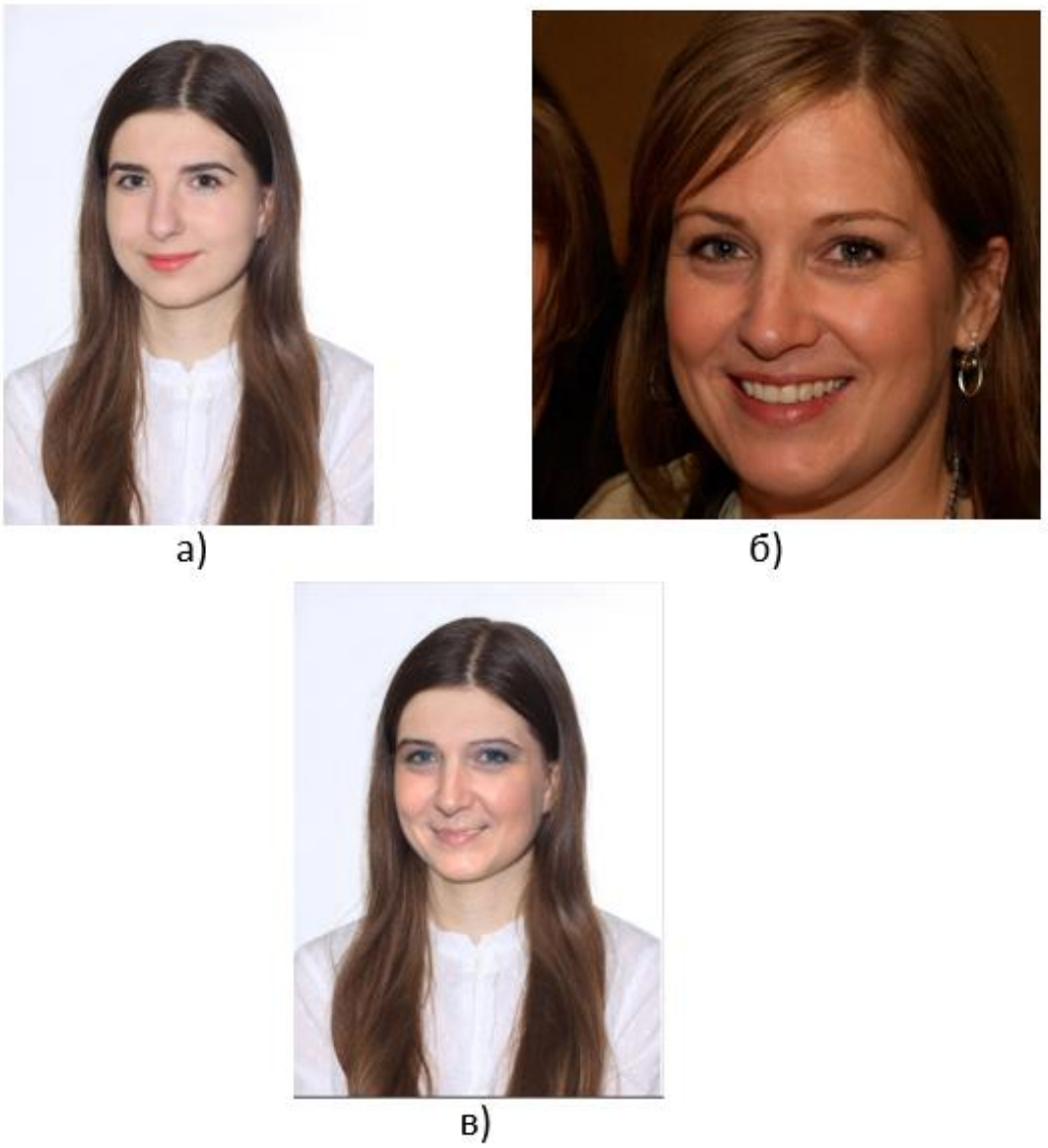

Рисунок 2.13 – Приклад 1: а) цільове зображення; б) вхідне зображення; в) результат роботи програми

Рисунок 2.14 – Приклад 2: а) цільове зображення; б) вхідне зображення; в) результат роботи програми

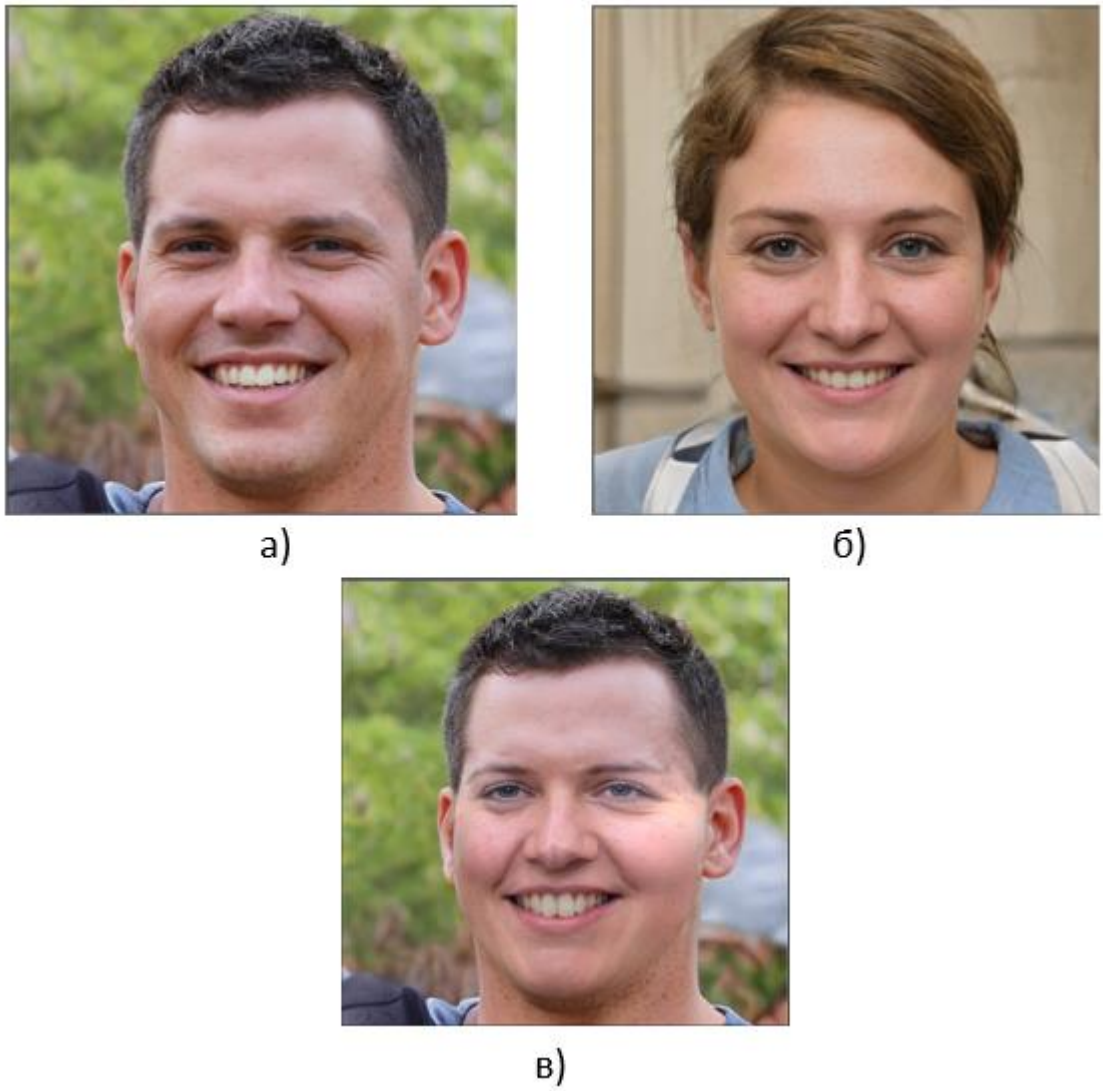

Рисунок 2.15 – Приклад 3: а) цільове зображення; б) вхідне зображення; в) результат роботи програми

ВИСНОВКИ

На підставі вивчення та аналізу існуючих технологій створення підробних зображень було розроблено метод, що відповідає поставленій задачі, а саме створенню підробних зображень шляхом заміни обличь людей.

Метод використовує тріангуляцію Делоне та афінні перетворення, що дає можливість частково наблизити вираз вхідного обличчя до цільового.

На якість створених зображень вливає початкова якість використаних фотографій. Також кращі результати можуть були отримані при використанні фотографій зі схожими ракурсами та виразами обличь

Було створено десктопний додаток, яких дозволяє на основі двох вибраних фотографій створити нове підробне зображення зі зміненим обличчям людини. Реалізована можливість збереження створеного зображення.

Подальша оптимізація можлива зі збільшенням кількості ключових точок на обличчі або застосуванням 3d маски замість тріангуляції з афінними перетвореннями. Також покращити результат можливо з використанням додаткових методів злиття фотографій, що враховують освітленість та колір шкіри і можуть допомогти зробити перехід між фотографіями більш плавним.

СПИСОК ЛІТЕРАТУРИ

1. Deepfake: как видеофейки стали серьезной угрозой и объединили общество [Електронний ресурс] URL: https://www.stopfake.org/ru/deepfake-kak-videofejki-stali-sereznoj-ugrozoj-i-obedinili-obshhestvo/
2. Как делают deepfake-видео [Електронний ресурс] URL: https://vc.ru/ml/94457-kak-delayut-deepfake-video-i-pochemu-luchshe-govorit-face-swap
3. Deepfake [Електронний ресурс] URL: https://ru.wikipedia.org/wiki/Deepfake
4. Github - shaoanlu/faceswap-gan: A denoising autoencoder + adversarial losses and attention mechanisms for face swapping. [Електронний ресурс]. URL:https://github.com/ shaoanlu/faceswap-GAN. (Accessed on 05/11/2019)
5. Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2016. Face2face: Real-time face capture and reenactment of rgb videos. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2387– 2395.
6. [Зображення] URL: https://2.bp.blogspot.com/-OjVr9tUB-zk/Wnbzwr12qjI/AAAAAAAAAAc/WJLifxbgPt0q_oOW5VRevuzBUdt334hhgCLcBGAs/s1600/Untitled-min.png
7. [Зображення] URL: https://new-science.ru/wp-content/uploads/2020/03/5151-6.jpg
8. [Зображення] URL: https://new-science.ru/wp-content/uploads/2020/03/5151-5.jpg
9. [Зображення] URL: https://www.iphones.ru/wp-content/uploads/2019/09/rs_1024x759-171218105305-1024-titanic-jack-tuxedo65-1241x710.jpg
10. D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. K. Nayar, “Face swapping: automatically replacing faces in photographs,” *ACM Transactions on Graphics*, vol. 27, no. 3, article no. 39, 2008.
11. S. Mahajan, L.-J. Chen, and T.-C. Tsai, “SwapItUp: A face swap application for privacy protection,” in *Proceedings of the 31st IEEE International Conference on*

Advanced Information Networking and Applications, AINA 2017, pp. 46–50, Taipei, Taiwan, March 2017.

12. Y. Nirkin, I. Masi, A. T. Tuăn, T. Hassner, and G. Medioni, "On face segmentation, face swapping, and face perception," in *Proceedings of the 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, pp. 98–105, Xi'an, China, May 2018.
13. V. Blanz, K. Scherbaum, T. Vetter, and H.-P. Seidel, "Exchanging faces in images," *Computer Graphics Forum*, vol. 23, no. 3, pp. 669–676, 2008.
14. H. X. Wang, C. H. Pan, H. F. Gong, and H. Y. Wu, "Facial image composition based on active appearance model," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008*, pp. 893–896, Las Vegas, Nev, USA, 2008.
15. Y. Lin, S. Wang, Q. Lin, and F. Tang, "Face swapping under large pose variations: A 3D model based approach," in *Proceedings of the 2012 13th IEEE International Conference on Multimedia and Expo, ICME 2012*, pp. 333–338, Melbourne, Victoria, Australia, July 2012.
16. I. Korshunova, W. Shi, J. Dambre, and L. Theis, "Fast face-swap using convolutional neural networks," in *Proceedings of the 16th IEEE International Conference on Computer Vision, ICCV 2017*, pp. 3697–3705, Venice, Italy, October 2017.
17. R. Natsume, T. Yatagawa, and S. Morishima, "RSGAN: face swapping and editing using face and hair representation in latent spaces," 2018, https://arxiv.org/abs/1804.03447.
18. [Зображення] URL: https://zhuanlan.zhihu.com/p/104299394
19. [Зображення] URL: https://upload.wikimedia.org/wikipedia/commons/d/db/Delaunay_circumcircles_vectorial.svg
20. [Зображення] URL: https://zhuanlan.zhihu.com/p/133301967
21. Рівняння Пуассона [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/%D0%A0%D1%96%D0%B2%D0%BD%D1%8F%

D0%BD%D0%BD%D1%8F_%D0%9F%D1%83%D0%B0%D1%81%D1%81%D0%BE%D0%BD%D0%B0

22. Шолле Ф. "Глубокое обучение на Python" 2018 рік : 35-50 с.

23. О. Райли "Введение в машинное обучение с помощью Python" 2019 рік : 67 с.

ДОДАТОК А

```
import cv2
import numpy as np
import dlib

def find_index(array):
    i = None
    for n in array[0]:
        i = n
        break
    return i

input_photo = cv2.imread("my2.jpg")
gray_input_photo = cv2.cvtColor(cv2.UMat(input_photo),
cv2.COLOR_BGR2GRAY)
maska_input_photo = np.zeros_like(gray_input_photo)
target_photo = cv2.imread("my4.jpg")
gray_target_photo = cv2.cvtColor(cv2.UMat(target_photo),
cv2.COLOR_BGR2GRAY)

height, width, channels = target_photo.shape

target_photo_new_countenance = np.zeros((height, width, channels), np.uint8)

# Input photo
input_countenances = detector_of_countenance(gray_input_photo)
for countenance in input_countenances:
    countenance_marks = predictor_of_shape(gray_input_photo, countenance)
    input_countenance_points = []
    for n in range(0, 68):
        x = countenance_marks.part(n).x
        y = countenance_marks.part(n).y
        input_countenance_points.append((x, y))

    input_points_array = np.array(input_countenance_points, np.int32)
    input_convex_hull = cv2.convexHull(input_points_array)
    # cv2.polylines(img, [countenance_convex_hull], True, (255, 0, 0), 3)
```

```
    cv2.fillConvexPoly(maska_input_photo, input_convex_hull, 255)

    input_countenance_photo = cv2.bitwise_and(input_photo, input_photo,
mask=maska_input_photo)

    # Тріангуляція Делоне
    rectangle = cv2.boundingRect(input_convex_hull)
    del_triangulation = cv2.Subdiv2D(rectangle)
    del_triangulation.insert(input_countenance_points)
    triangles = del_triangulation.getTriangleList()
    triangles = np.array(triangles, dtype=np.int32)

    indexes_triangles = []
    for n in triangles:
        triangle_point_1 = (n[0], n[1])
        triangle_point_2 = (n[2], n[3])
        triangle_point_3 = (n[4], n[5])

        index_triangle_point_1 = np.where((input_points_array ==
triangle_point_1).all(axis=1))
        index_triangle_point_1 = find_index(index_triangle_point_1)

        index_triangle_point_2 = np.where((input_points_array ==
triangle_point_2).all(axis=1))
        index_triangle_point_2 = find_index(index_triangle_point_2)

        index_triangle_point_3 = np.where((input_points_array ==
triangle_point_3).all(axis=1))
        index_triangle_point_3 = find_index(index_triangle_point_3)

        if index_triangle_point_1 is not None and index_triangle_point_2 is not None
and index_triangle_point_3 is not None:
            triangle = [index_triangle_point_1, index_triangle_point_2,
index_triangle_point_3]
            indexes_triangles.append(triangle)

# Target photo
target_countenances = detector_of_countenance(gray_target_photo)
for countenance in target_countenances:
```

```
    countenance_marks = predictor_of_shape(gray_target_photo, countenance)
    target_countenance_points = []
    for n in range(0, 68):
        x = countenance_marks.part(n).x
        y = countenance_marks.part(n).y
        target_countenance_points.append((x, y))

    target_points_array = np.array(target_countenance_points, np.int32)
    target_convex_hull = cv2.convexHull(target_points_array)

lines_space_mask = np.zeros_like(gray_input_photo)
lines_space_new_countenance = np.zeros_like(target_photo)
# Triangulation of both countenances
for tr_ind in indexes_triangles:
    input_tr_point1 = input_countenance_points[tr_ind[0]]
    input_tr_point2 = input_countenance_points[tr_ind[1]]
    input_tr_point3 = input_countenance_points[tr_ind[2]]
    input_triangle = np.array([input_tr_point1, input_tr_point2, input_tr_point3],
np.int32)

    input_rectangle = cv2.boundingRect(input_triangle)
    (x, y, w, h) = input_rectangle
    cropped_input_tr = input_photo[y: y + h, x: x + w]
    cropped_input_tr_mask = np.zeros((h, w), np.uint8)

    input_points_array = np.array([[input_tr_point1[0] - x, input_tr_point1[1] - y],
                                   [input_tr_point2[0] - x, input_tr_point2[1] - y],
                                   [input_tr_point3[0] - x, input_tr_point3[1] - y]], np.int32)

    cv2.fillConvexPoly(cropped_input_tr_mask, input_points_array, 255)

    cv2.line(lines_space_mask, input_tr_point1, input_tr_point2, 255)
    cv2.line(lines_space_mask, input_tr_point2, input_tr_point3, 255)
    cv2.line(lines_space_mask, input_tr_point1, input_tr_point3, 255)
    lines_space = cv2.bitwise_and(input_photo, input_photo,
mask=lines_space_mask)

    target_tr_point1 = target_countenance_points[tr_ind[0]]
    target_tr_point2 = target_countenance_points[tr_ind[1]]
```

```
    target_tr_point3 = target_countenance_points[tr_ind[2]]
    target_triangle = np.array([target_tr_point1, target_tr_point2, target_tr_point3],
np.int32)

    target_rectangle = cv2.boundingRect(target_triangle)
    (x, y, w, h) = target_rectangle

    cropped_target_tr_mask = np.zeros((h, w), np.uint8)

    target_points_array = np.array([[target_tr_point1[0] - x, target_tr_point1[1] - y],
                                    [target_tr_point2[0] - x, target_tr_point2[1] - y],
                                    [target_tr_point3[0] - x, target_tr_point3[1] - y]], np.int32)

    cv2.fillConvexPoly(cropped_target_tr_mask, target_points_array, 255)

    # Warp triangles
    input_points_array = np.float32(input_points_array)
    target_points_array = np.float32(target_points_array)
    matrix = cv2.getAffineTransform(input_points_array, target_points_array)
    transformed_tr = cv2.warpAffine(cropped_input_tr, matrix, (w, h))
    transformed_tr = cv2.bitwise_and(transformed_tr, transformed_tr,
mask=cropped_target_tr_mask)

    target_new_countenance_rect_area = target_photo_new_countenance[y: y + h, x: x
+ w]
    target_new_countenance_rect_area_gray =
cv2.cvtColor(target_new_countenance_rect_area, cv2.COLOR_BGR2GRAY)
    _, mask_triangles_designed =
cv2.threshold(target_new_countenance_rect_area_gray, 1, 255,
cv2.THRESH_BINARY_INV)
    transformed_tr = cv2.bitwise_and(transformed_tr, transformed_tr,
mask=mask_triangles_designed)

    target_new_countenance_rect_area = cv2.add(target_new_countenance_rect_area,
transformed_tr)
    target_photo_new_countenance[y: y + h, x: x + w] =
target_new_countenance_rect_area

target_countenance_mask = np.zeros_like(gray_target_photo)
```

```
target_photo_mask = cv2.fillConvexPoly(target_countenance_mask,
target_convex_hull, 255)
target_countenance_mask = cv2.bitwise_not(target_photo_mask)

target_head_without_countenance = cv2.bitwise_and(target_photo, target_photo,
mask=target_countenance_mask)
final_photo = cv2.add(target_head_without_countenance,
target_photo_new_countenance)

(x, y, w, h) = cv2.boundingRect(target_convex_hull)
center_target_countenance = (int((x + x + w) / 2), int((y + y + h) / 2))

final_corrected_photo = cv2.seamlessClone(final_photo, target_photo,
target_photo_mask, center_target_countenance, cv2.NORMAL_CLONE)

cv2.imwrite("result.jpg", final_corrected_photo)
cv2.waitKey(0)
cv2.destroyAllWindows()
```