Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно–аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

ПРОГРАМНЕ МОДЕЛЮВАННЯ МЕТОДІВ ОЖИВЛЕННЯ ЗОБРАЖЕНЬ У МУЛЬТИМЕДІЙНИХ СИСТЕМАХ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ–21–2
Льолін Д. В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо–професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Гороховатський В. О.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно–аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо–професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Льоліну Данилу Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмне моделювання методів оживлення зображень у мультимедійних системах

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 25 травня 2025 р.
3. Вихідні дані до роботи науково–методична та науково–технічна література, матеріали конференцій, дані інтернет–мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV, мова програмування Python, редактор коду Microsoft Visual Studio.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз методів нейромережевого оживлення зображень обличчя.
2. Огляд алгоритмів нейромережевого моделювання міміки з використанням FOMM та MediaPipe.
3. Розробка системи передачі міміки на тривимірну модель обличчя в середовищі Blender через shape keys.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність задачі, постановка задачі, методи генерації міміки за допомогою FOMM, обробка ключових точок через MediaPipe, передача параметрів в Blender.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25–10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25–14.04.25	
4	Аналіз технічних засобів	15.04.25–20.04.25	
5	Розробка методу	21.04.25–27.04.25	
6	Програмна реалізація	28.04.25–11.05.25	
7	Оформлення пояснювальної записки	12.05.25–20.05.25	
8	Перевірка на нормоконтроль	21.05.25–01.06.25	
9	Перевірка на плагіат	21.05.25–01.06.25	
10	Рецензування	21.05.25–01.06.25	
11	Підготовка презентації та доповіді	21.05.25–18.06.25	
12	Занесення роботи в електронний архів	02.06.25–18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25–18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Гороховатський В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 66 с., 4 табл., 53 рис., 43 джерела.

АНІМАЦІЯ ОБЛИЧЧЯ, НЕЙРОННІ МЕРЕЖІ, FOMM, BLENDER, МІМІКА, КЛЮЧОВІ ТОЧКИ, МОДЕЛЮВАННЯ, ОЖИВЛЕННЯ.

У цій роботі ми розглянули спосіб оживлення моделі обличчя з участю нейромережі First Order Motion Model. В основі методу лежить передача міміки з відео, де людина рухає обличчям, на зображення і тривимірну модель у Blender.

Розроблено програму, яка вміє обробляти відео, визначати ключові точки на обличчі і переносити їх на тривимірну модель як анімацію. Усе вище сказане працює через Shape Keys у Blender та дозволяє робити обличчя живим без якогось складного рігінгу.

Результати роботи показали, що обличчя може реалістично рухаєтися, майже як у справжньому відео. Такий підхід можна застосувати в ігровій розробці, фільмах, онлайн–аватарах та в інших сферах нашого життя.

FACE ANIMATION, NEURAL NETWORKS, FOMM, BLENDER, FACIAL EXPRESSIONS, KEY POINTS, MODELING, ANIMATION.

In this work, we have considered a method for animating a face model using the First Order Motion Model neural network. The method is based on the transfer of facial expressions from video, where a person moves his or her face, to an image and a 3D model in Blender.

A program has been developed that can process video, identify key points on the face and transfer them to a 3D model as an animation. All of the above works through Shape Keys in Blender and allows you to make your face look alive without any complicated rigging.

The results of the work showed that the face can move realistically, almost like in a real video. This approach can be used in game development, movies, online avatars and other areas of our lives.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КТ – ключова точка (Keypoint) – опорна точка на зображенні обличчя.

FOMM – First Order Motion Model – модель нейронної мережі для оживлення зображень шляхом передачі міміки.

Shape Key – елемент Blender, що дозволяє створювати деформовані форми сітки для анімації обличчя.

IDE – інтегроване середовище розробки програмного забезпечення, що поєднує текстовий редактор, компілятор і засоби запуску скриптів у єдиному інтерфейсі. Наприклад, Visual Studio, VS Code, PyCharm

MediaPipe – фреймворк для виділення ключових точок обличчя в реальному часі.

UDP Socket – спосіб передачі даних між програмами (наприклад, з Python до Blender) через інтернет-протокол. Blender Python API – набір інструментів для управління анімацією, shape keys та об’єктами сцени у Blender за допомогою Python.

Driving Video – відео, що містить міміку, яка буде перенесена на інше зображення або модель.

VR – віртуальна реальність; технологія, що дозволяє користувачу зануритися у штучно створене середовище, яке може реагувати на його дії в реальному часі

Source Image – статичне зображення, яке буде оживлено на основі руху з driving video.

ВСТУП

Сучасні технології комп’ютерного зору швидко розвиваються у нашому столітті. Цей розвиток зачепив багато різних галузей від розважальної індустрії до кіно та мистецтва. Одним із таких цікавих напрямів є оживлення зображень та обличчя. Таких підхід дозволяє зробити зображення схожим на живе та реалістичне, схожим на обличчя в реальному світі

Завдяки усім досягненням у сфері глибокого навчання та розвитку спеціалізованих нейромереж з’явилася можливість здійснювати такі анімації автоматично. Однією з відомих моделей для таких задач є First Order Motion Model (FOMM), яка дозволяє переносити рухи з відео на статичне зображення з високою точністю. Ця технологія стала поштовхом до створення нових способів створення реалістичної міміки [9-12].

У межах цієї роботи було зосереджено увагу на перенесенні міміки з відео на тривимірну модель обличчя, попередньо завантажену до графічного середовища Blender. Центральною складовою методу є використання ключових точок обличчя, що відповідають за відстеження рухів окремих його частин – таких як очі, брови, губи тощо. Обчислення положення цих точок здійснюються за допомогою нейромережі FOMM, яка аналізує відео та генерує оживленне фото, на якому рухи відповідають рухам на оригінальному обличчі.

Актуальність дослідження обумовлюється як зростаючим попитом на автоматизовані інструменти в області комп’ютерної графіки, так і широким практичним застосуванням подібних систем у сфері створення анімованих персонажів, цифрових аватарів, віртуальних ведучих та інших мультимедійних систем. Інтеграція таких інструментів дозволяє істотно скоротити витрати часу та ресурсів на створення анімації, відкриваючи нові можливості для розробників і дизайнерів у сфері візуального контенту.

.

1 МЕТОДИ НЕЙРОМЕРЕЖЕВОГО ОЖИВЛЕННЯ ЗОБРАЖЕНЬ ОБЛИЧЧЯ

1.1 Аналіз програмних засобів для анімації обличчя

Прикладні задачі та застосування анімації обличчя – це напрям у комп’ютерному зорі, який має справу з передачею виразів, міміки та жестів з одного обличчя на інше, з відео на зображення. Такі технології вже знайшли застосування в кіноіндустрії, створенні віртуальних аватарів, відеозв’язку, у геймінгу та віртуальній реальності. У поєднанні з нейромережами цей процес спрощується і прискорюється [1], відкриваючи можливість для створення реалістичних анімацій навіть без професійного рігінгу чи захоплення рухів у студійних умовах.

Серед прикладних задач можна виділити такі як оживлення фотографій історичних або нерухомих персонажів, передачу міміки в реальному часі для стримів або VR-середовищ, автоматизоване створення відеоконтенту для медіа

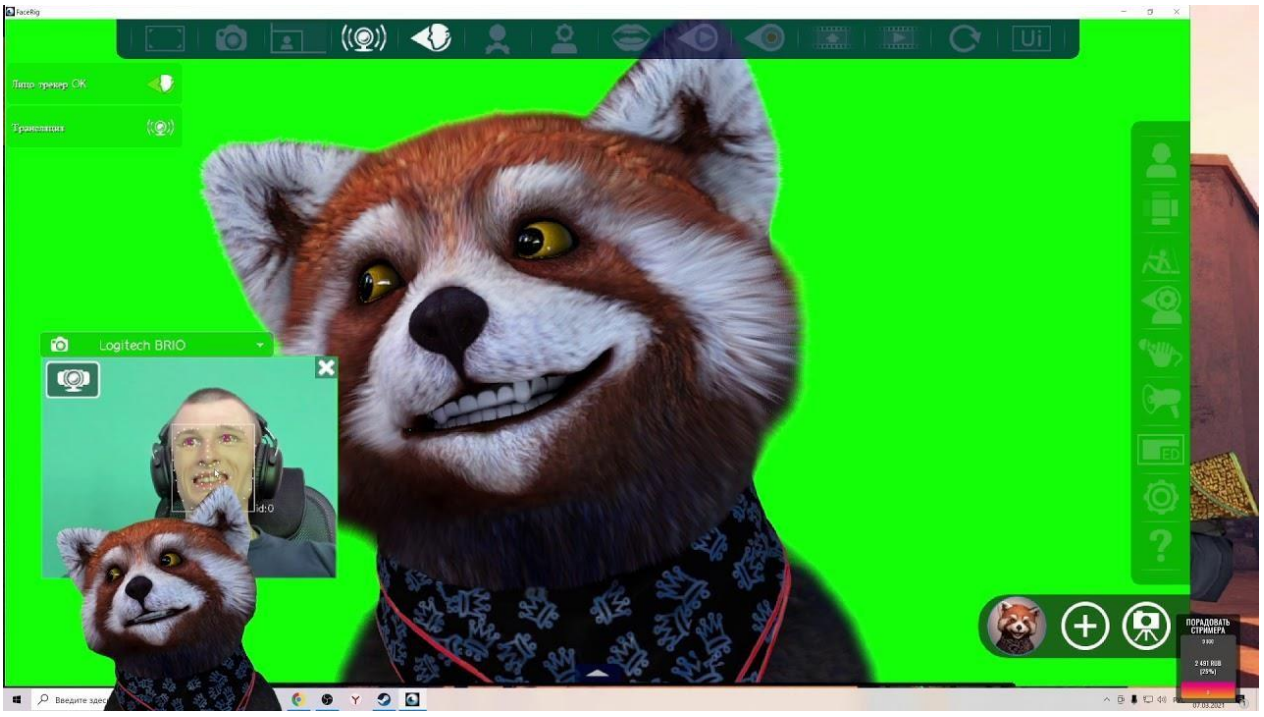


Рисунок 1.1 – Передача міміки в реальному часі для стримів

Завдяки розвитку нейромережевих технологій стало можливим автоматично оживляти зображення без потреби у фізичному захопленні руху чи складному рігінгу. Однією з найвідоміших архітектур для цього є First Order Motion Model (FOMM), запропонована у 2019 році [1]. Blender використовується як середовище для анімації міміки.

Blender – це безкоштовне програмне забезпечення з відкритм кодом [2] для створення та редагування тривимірної графіки, яке підтримує повноцінну систему ріггінгу, анімації, обробки геометрії та Python-скриптів. Саме тому він широко використовується у задачах візуалізації, включаючи анімацію міміки на тривимірних моделях обличчя. Однією з ключових функцій Blender, що використовується для задач анімації обличчя, є shape keys. Вони дозволяють створювати кілька варіантів положення моделі (наприклад, усмішка, закриття очей та інше), які потім можна змішувати з основною формою обличчя. Кожен shape key може бути керований вручну або програмно через `value` в Python API [3].

При роботі з нейромережею FOMM значення shape keys можуть змінюватись у реальному часі відповідно до координат ключових точок, отриманих із відео. Наприклад, зменшення відстані між точками повік може вказувати на моргання, а зміщення кутиків рота на усмішку. Для кожного виразу створюється окремий shape key, а значення його активується відповідно до отриманих зсувів точок або через обчислений коефіцієнт.

Blender також підтримує інтеграцію з Python-сценаріями щоб автоматизувати процес оновлення значень shape keys на основі даних з нейромережі. У роботі використовувався підхід, при якому координати ключових точок з оживленого фото за допомогою FOMM зчитувались через скрипт і передавались у Blender через сокет [4, 5]. Python-скрипт у Blender зчитував ці дані та встановлював відповідні значення shape keys, змінюючи вираз на обличчі. Тому Blender виступає гнучким середовищем, що дозволяє реалізовувати реалістичну анімацію міміки обличчя на основі реальних відео, без необхідності використання дорогих систем захоплення руху.

У процесі реалізації задачі оживлення обличчя розглянуто кілька можливих програмних середовищ, наприклад Blender, Autodesk Maya, Unity та FaceWare Studio. Основними критеріями для вибору були: доступність, підтримка механізмів анімації через shape keys або blend shapes, наявність скриптів, а також можливість автоматизації процесу з урахуванням роботи нейромережі.

У таблиці 1.1 наведено порівняльну характеристику середовищ:

Таблиця 1.1 – Порівняльна характеристика середовищ

Параметр	**Blender**	**Autodesk Maya**	**Unity**	**FaceWare Studio**
Вартість	Безкоштовно	Платна (ліцензія)	Безкоштовно, Обмежена	Платна ($1000+)
Shape Keys	Повна підтримка	Підтримка	Часткова	Немає
Python API	Повна інтеграція	Обмежена	Немає (C#)	Немає
Підтримка ML / FOMM	Через скрипти	Складно реалізувати	Через плагіни	Закрита система
Гнучкість UI	Гнучке, кастомізоване	Потужне, але складне	Просте, обмежене	Для live mocap

Тому у результаті аналізу було обрано саме Blender, оскільки він поєднує у собі всі необхідні для реалізації проєкту характеристики, такі як повна підтримка shape keys, можливість автоматизованого керування через Python API, а також повна безкоштовність. Це дозволило створити анімовану модель без додаткових витрат і обмежень.

1.2 Засоби нейромережі FOMM з використанням Blender

First Order Motion Model (FOMM) – це нейромережева модель, що виконує анімацію зображення на основі руху з відео-джерела. Вона представляє рух об’єкта за допомогою невеликого набору ключових точок (keypoints) та локальних афінних перетворень [1, 6, 7]. По суті, FOMM навчається кодувати рух як зміщення цих ключових точок між кадрами. Для прикладу, при анімації обличчя FOMM може самостійно визначити точки на характерних ділянках (біля очей, рота, брів тощо) і відстежувати їх пересування.

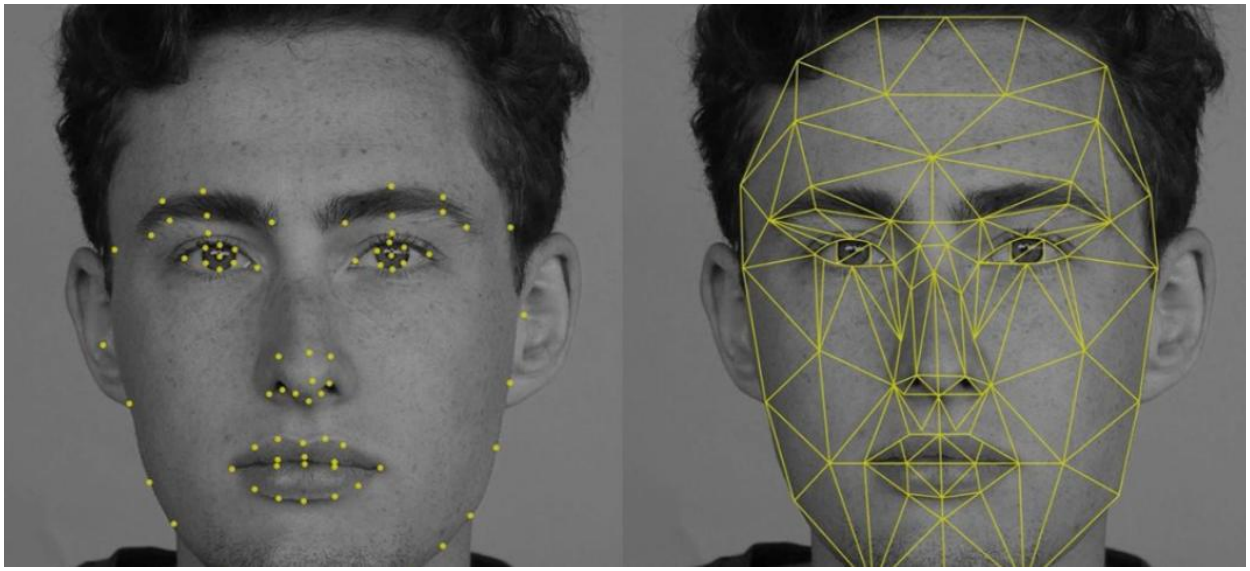


Рисунок 1.2 – Приклад ключових точок на обличчі

Таким чином отримується компактне представлення міміки у вигляді координат ключових точок та їх змін у часі. Це ще одна причина чому FOMM може застосовується у програмних рішеннях, які синхронізуються з редакторами (наприклад, Blender), де результати передаються через shape keys або rig-каркаси [8].

Процес оживлення зображення обличчя за допомогою нейромережі First Order Motion Model складається з кількох послідовних етапів. На першому кроці система отримує два вхідні дані: статичне зображення обличчя, яке буде оживлятись, та відео з джерелом руху, зазвичай це відео з реальним обличчям,

яке демонструє міміку чи емоції. Обличчя повинно бути не занадто далеко та не занадто близько.

Рисунок 1.3 – Формат обличчя для відео

Далі нейромережа FOMM автоматично знаходить ключові точки на обличчі – це координати певних анатомічних орієнтирів, таких як наприклад, очі, брови, рот, ніс. Ці точки визначають, як частини обличчя змінюються протягом відео. Потім для кожної точки обчислюється вектор зсуву, що описує її рух між кадрами відео таким чином [1]:

$$\Delta k_i = k_i^t - k_i^0, \tag{1.1}$$

де k_i^0– координати точки в нейтральному кадрі;

k_i^t – координати в поточному кадрі;

Δk_i– вектор зміщення.

Ця робота базується на цій ідеї, але в цій роботі адаптовано її під практичний сценарійд; записав відео з мімікою, зробив фото, оживив фото та передав результат у Blender на тривимірній моделі обличчя [11].

На завершальному етапі результати анімації прив’язуються до shape keys тривимірної моделі. Кожен рух міміки відповідає окремому ключу на моделі, значення якого змінюється відповідно до даних, згенерованих нейромережею [2, 3].

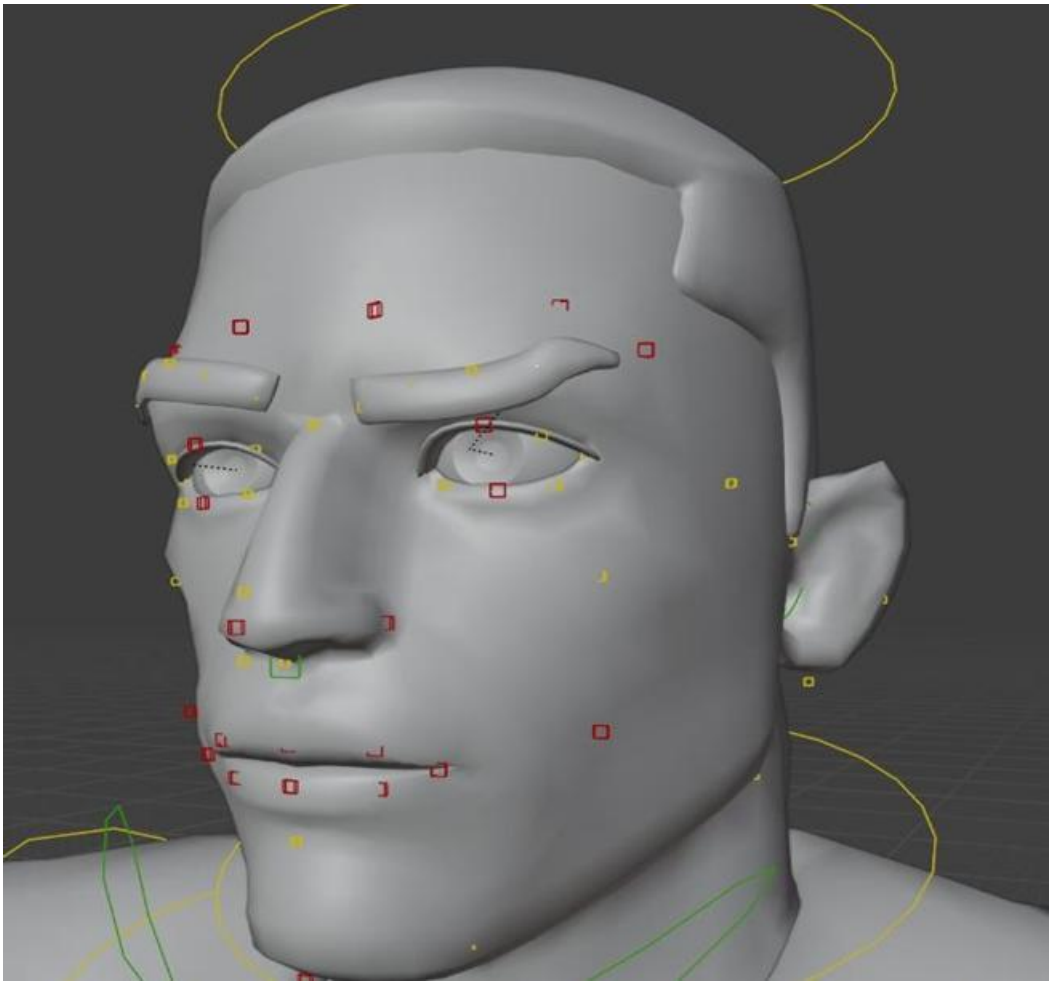


Рисунок 1.4 – Приклад shape keys тривимірної моделі

У Blender для анімації міміки широко застосовуються shape keys, це збережені форми сітки, які можна плавно змішувати з основною формою моделі. Shape keys дозволяють змінювати геометрію об’єкта для досягнення

потрібного виразу обличчя, оскільки як було сказано, shape keys можуть відповідати таким виразам, як усмішка, моргання та іншим.

Користувач заздалегідь створює ці ключі форми або вручну або за допомогою плагіну, задаючи положення вершин моделі для кожного виразу. Є базова форма обличчя і декілька відхилених форм. Під час анімації значення shape key регулюється від 0 (відсутність ефекту, базова форма) до 1 (повний ефект заданого ключа форми), що викликає відповідну зміну виразу обличчя на моделі [2] за формолую:

$$Value_{shape} = \frac{|\Delta k_i|}{|\Delta k_{max}|}, \tag{1.2}$$

де $|\Delta k_j|$ – довжина вектора зсуву в поточному кадрі;

$|\Delta k_{max}|$ – максимальне зміщення, яке відповідає 100% ефекту (значенню 1.0).

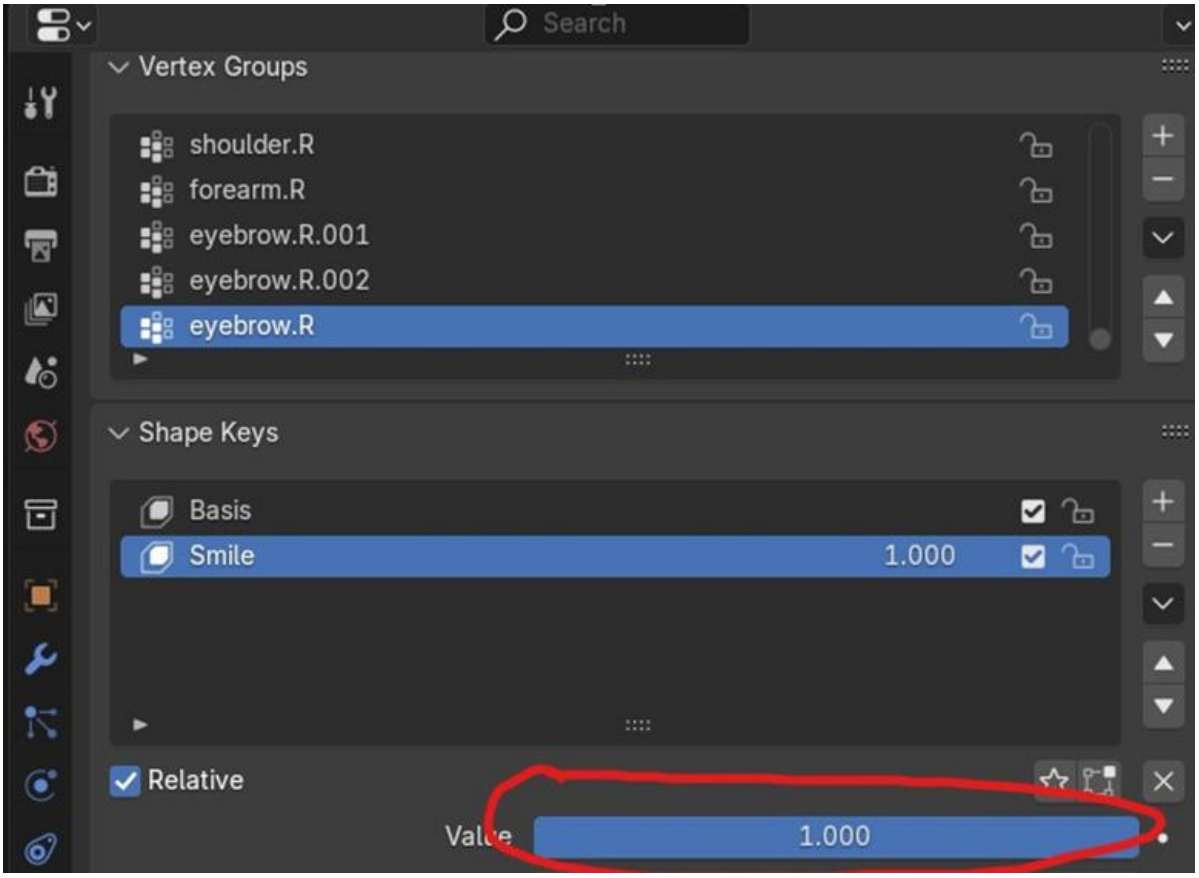


Рисунок 1.5 – Приклад налаштованого shape keys у моделі

Якщо порівняти цей метод з іншими методами анімації, такими як скелетна анімація або використання контролерів на кістках, shape keys у Blender надають значно гнучкіші й точніші можливості для керування мімікою обличчя. Скелетна анімація використовує набір кісток, що обертаються або зміщуються, змінюючи геометрію моделі через ваги. Такий підхід є ефективний для глобальних рухів, як поворот голови, але виявляється недостатньо точним для дрібних локальних змін, наприклад, при частковому закритті очей. На відміну від цього, shape keys дозволяють створювати деформовані варіанти цих конкретних ділянок сітки, які не залежать від обертання кісток. Це дозволяє точно контролювати положення кожної вершини моделі. Крім цього, ще однією важливою перевагою shape keys є не складна інтеграція з Python API Blender. Завдяки цьому, автоматичне оновлення значень shape keys відбувається в реальному часі через скрипт, який приймає дані з нейронної мережі або від інших систем трекінгу міміки Для скелетної анімації аналогічний сценарій потребував би значно складніших обчислень, включаючи інверсну кінематику та складну роботу з ваговими картами. Shape keys значно простіші в налаштуванні та не потребують додаткових об’єктів або складної структури rig. У поєднанні з UDP-з’єднанням вони дозволяють досягти плавної, стабільної та масштабованої передачі міміки навіть за нестабільного відео.

Для того, щоб перенести міміку з анімованого зображення, яке створюється за допомогою моделі FOMM, на тривимірній модель у Blender, необхідно реалізувати хороший зв’язок між зміною положення ключових точок обличчя та значеннями shape keys, які відповідають за деформацію геометрії обличчя у Blender. У цьому процесі важливо враховувати як анатомічну логіку обличчя, так і технічні обмеження моделі.

Передусім система аналізує послідовність кадрів з анімованого відео, отриманого через FOMM. За допомогою бібліотеки MediaPipe на кожному кадрі автоматично визначаються координати ключових точок обличчя. Ці точки несуть у собі інформацію про поточний стан міміки. Тому на першому етапі обов’язково фіксується так званий «нейтральний стан», тобто положення

обличчя без емоцій. Це дозволяє з кожного наступного кадру обчислювати лише зміщення, тобто наскільки точка змінилася порівняно з початковим станом. На наступному етапі ці зміщення інтерпретуються як інтенсивність певної міміки. Наприклад, якщо точка біля кута рота перемістилася вгору на 50% від максимально можливої зміни, яка була визначена під час калібрування, то відповідному shape key (наприклад, «Smile») присвоюється значення 0.5. Те саме стосується моргання. Зменшення вертикальної відстані між верхньою і нижньою повікою може прямо впливати на shape key «BlinkLeft» чи «BlinkRight».

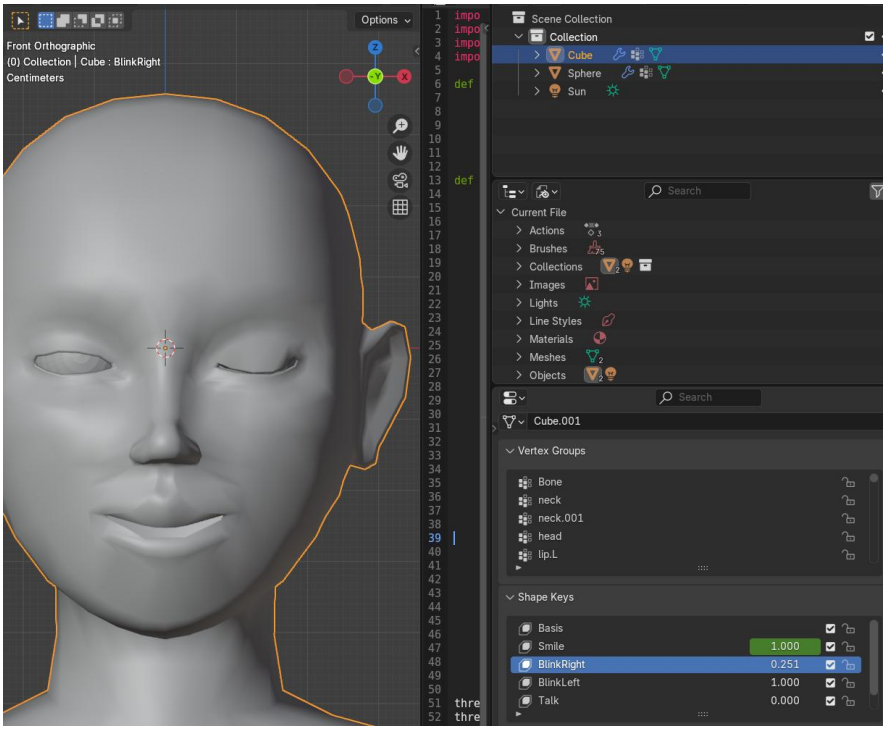


Рисунок 1.7 – Blender очікує на отримання даних

Таким чином, система кожного кадру формує набір чисел від 0 до 1, де кожне число означає ступінь активації певного виразу обличчя. Ці значення передаються у Blender через UDP-сокет, і там уже встановлюються як значення відповідних shape keys на об’єкті. Blender у реальному часі візуалізує міміку, яка повністю залежить від зміни позицій точок на анімованому зображенні.

Рисунок 1.7 – Blender очікує на отримання даних

Такий підхід, заснований на відносному зміщенні, забезпечує достатньо природну реакцію міміки та дозволяє уникнути раптових багів. Але, якщо потрібно досягти ще точнішого контролю, можна застосовувати додаткову нормалізацію або перетворення, наприклад, використовувати нелінійне масштабування щоб відображати емоцію не лінійно, а з урахуванням того, як людина насправді її сприймає. Також можна вводити вагові коефіцієнти для різних точок, які частіше мають вирішальне значення в передачі конкретного виразу.

1.3 Постановка задачі

Покращення точності та реалістичності методів анімації зображень обличчя є актуальним напрямком розвитку технологій віртуальних персонажів, віртуальної реальності, геймдева та інших інтерфейсів «людина – комп’ютер».

Об’єктом дослідження є процес оживлення статичного обличчя з використанням нейромережевого підходу на основі ключових точок та їх динаміки, що визначається з відео-джерела міміки.

Для досягнення мети необхідно проаналізувати існуючі підходи до анімації обличчя тривимірної моделі, такі як нейромережеві методи передачі міміки. Також потрібно дослідити принцип роботи моделі FOMM. Реалізувати передачу координат ключових точок за допомогою FOMM у відповідні shape keys з використанням Python API, налаштувати зв’язок між зміщенням ключових точок і значеннями shape keys для основних виразів, таких як усмішка, моргання та інших, а також провести експерименти з анімацією міміки на основі відео та оцінити якість отриманих результатів.

2 СИСТЕМА ОЖИВЛЕННЯ ОБЛИЧЧЯ НА ОСНОВІ ВІДЕОПОТОКУ

2.1 Етапи побудови системи передачі міміки у Blender

Передача міміки на тривимірну модель це процес, який включає кілька етапів. У межах цього дослідження запропоновано спосіб, який поєднує два основні компоненти, такі як попереднє оживлення зображення за допомогою нейронної мережі First Order Motion Model (FOMM) та передачу згенерованої міміки у середовище Blender через систему shape keys [2]. Такий підхід дозволяє досягти більшої реалістичності анімації при використанні лише одного вхідного зображення.

Для передачі міміки у систему подається попередньо записане відео, у якому користувач записав вирази обличчя, такі як усмішку, моргання, нейтральний стан та інші [8]. Такий підхід дозволяє уникнути залежності від вебкамери та її якості, а також забезпечує контрольовані умови зйомки, що позитивно впливає на подальшу якість обробки [1].

Відео, яке використовується для аналізу міміки, повинно відповідати певним критеріям, що забезпечують стабільну та точну роботу системи трекінгу. Фронтальний ракурс є важливим, тобто камера має бути розташована прямо перед обличчям користувача, оскільки при зйомці під кутом алгоритми можуть менш точно визначати ключові точки [9]. Камера повинна залишатися нерухомою, тому бажано уникати ручної зйомки, щоб забезпечити стабільність кадру. Освітлення має бути рівномірним, без тіней і засвічень, з бажаним фронтальним джерелом світла. Надмірно яскравий фон або контрастні джерела світла можуть викликати помилки в обробці зображення. Фон відео бажано залишити однорідним і спокійним, без зайвих рухомих елементів, які можуть відволікати модель. Відео повинно мати роздільну здатність не менше 640×480, бути збереженим у форматах mp4, avi або mov, та мати стабільну частоту кадрів у межах 25–30 кадрів на секунду [8].

Рисунок 2.1 – Приклад підходящого ракурсу для фото і відео

Також бажано, щоб у кадрі було лише одне обличчя, так як наявність сторонніх осіб або частин тіла (наприклад, руки, що торкаються обличчя) може призвести до помилок при трекінгу міміки. За потреби відео можна обрізати, стабілізувати або відкоригувати кольоровий баланс у будь-якому редакторі перед обробкою у FOMM.

Другим обов’язковим компонентом вхідних даних є статичне зображення обличчя, яке буде анімоване. Це зображення виконує роль так званого source image, основи, до якої застосовуються рухи обличчя з відео. Якість і формат цього зображення напряму впливають на результат генерації, тому його підготовка має відповідати певним критеріям.

Зображення обличчя повинно бути фронтальним, тобто обличчя має дивитися прямо в камеру. Будь-яке викривлення перспективи може призвести до деформацій при генерації анімації, оскільки модель сприймає таке зображення як нетипове для обробки. Важливо також, щоб зображення було

різким і добре освітленим. Необхідно уникати тіней, які можуть закривати риси обличчя, а також надмірно яскравих ділянок на шкірі, які можуть з’являтися через спалахи або інтенсивне локальне освітлення. Крім того, обличчя на зображенні має бути вільним від сторонніх елементів: волосся, руки, окуляри, маски, а також текст або графіка можуть заважати нейронній мережі правильно зчитати ключові точки, що впливає на точність анімації [5]. Рекомендовано, щоб зображення було обрізане таким чином, аби фокус був лише на голові, без надлишкового фону або зайвих об’єктів. Бажано також дотримуватися формату із співвідношенням сторін 1:1, що відповідає вимогам моделі FOMM для коректного функціонування [10].

Рисунок 2.2 –Приклад підходящого ракурсу для відео

Також важливо, щоб на зображенні була та сама особа, що й на driving video, хоча FOMM дозволяє використовувати різні обличчя, у межах цього проєкту використовується один і той самий користувач для обох файлів. Для вибору зображення у програмі реалізовано інтерфейс через бібліотеку Tkinter, що дозволяє обрати файл із файлової системи користувача. Програма додатково перевіряє тип файлу (*.jpg, *.png) та наявність за вказаним шляхом. У випадку помилки виводиться повідомлення та пропонується вибрати інший файл [10].

На наступному етапі здійснюється синтез анімованого обличчя на основі статичного зображення користувача та відео з мімікою.

Для цього використовується нейронна модель First Order Motion Model for Image Animation (FOMM) [1, 7, 14]. Основна її ідея це перенесення руху з одного обличчя на інше без використання тривимірної реконструкції, масок або маркерів. Модель працює у повністю двовимірному просторі, що робить її простою у використанні [11]. FOMM складається з модулю оцінки руху який знаходить ключові точки обличчя у кожному кадрі відео та обчислює локальні афінні перетворення та з генератора зображень, який деформує оригінальне зображення відповідно до отриманої мапи руху та генерує анімований кадр.

Апарат ключових точок успішно використовується у задачах комп'ютерного зору для розпізнавання об'єктів [23 – 39].

Кожна ключова точка має свою локальну область, у межах якої рух описується афінним перетворенням, що враховує обертання, масштаб і зсув.

$$x' = A_i \cdot x + b_i , \tag{2.1}$$

де x' – координати пікселя на вихідному зображенні;

A_i – матриця 2×2, що задає масштабування й обертання;

b_i – вектор зсуву для області i.

FOMM генерує відео, в якому обличчя з фотографії динамічно повторює рухи з відео. Освітлення, стиль та текстура зображення зберігаються. Генерація виконується покадрово, що дозволяє досягати плавної та реалістичної анімації.

Після успішного запуску нейронної моделі FOMM формується послідовність згенерованих кадрів, які разом утворюють анімоване відео. На цьому етапі відбувається остаточне поєднання кадрів у відеофайл, що забезпечує подальше використання результату для аналізу міміки та передачі в тривимірне середовище Blender. Під час генерації FOMM створює нове зображення для кожного кадру, в якому статичне зображення набуває позицій та міміки з нашим відео. Ці зображення мають однаковий розмір і зберігаються у внутрішній пам’яті під час виконання скрипта. Щоб зберегти анімацію у відеоформаті, використовується бібліотека ffmpeg [14]. Фінальний файл має назву result.mp4, і його структура повністю відповідає оригінальному відео за кількістю кадрів, але містить лише обличчя користувача з рухами міміки.

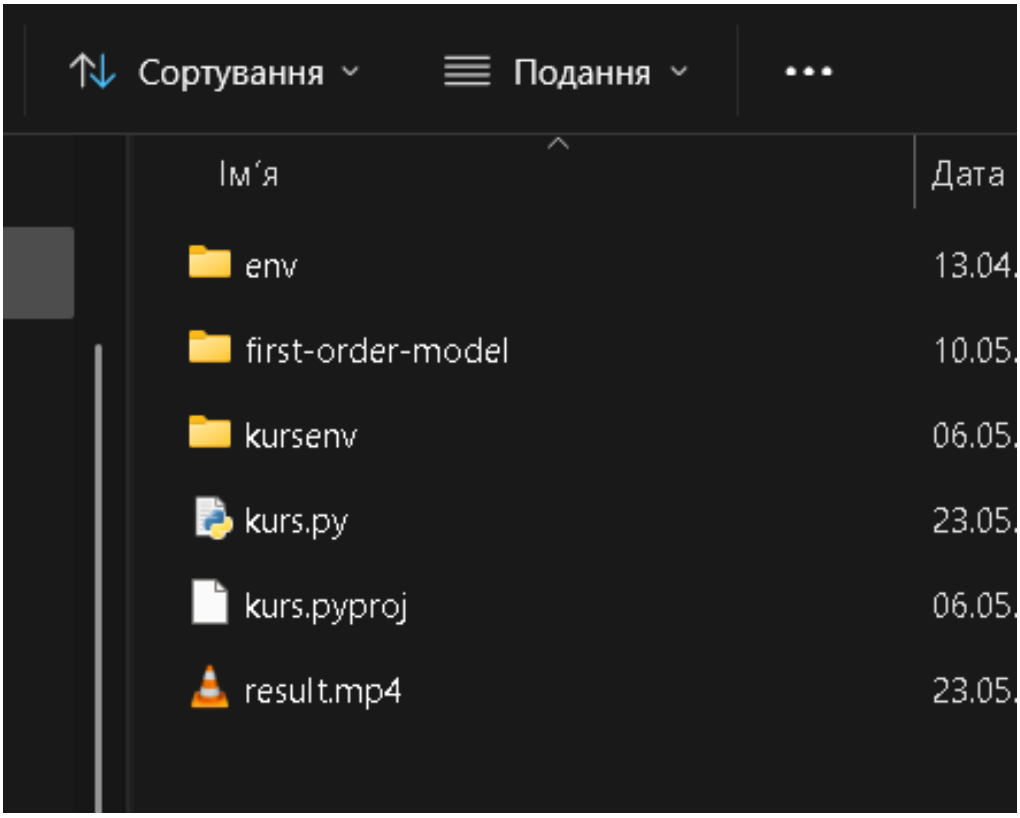


Рисунок 2.3 – Готовий файл result.mp4

Після генерації анімованого відео за допомогою FOMM наступним кроком є виділення ключових точок обличчя на кожному кадрі. Це дозволяє формалізувати міміку у вигляді числових координат, які далі передаються для управління shape keys у Blender. Для цього використовується бібліотека

MediaPipe, а саме компонент Face Mesh, який дозволяє визначити до 468 ключових точок на обличчі в реальному часі [1,13, 40, 43].

MediaPipe Face Mesh – це високошвидкісна модель комп'ютерного зору, яка працює на відеопотоці та використовує попередньо навчену нейронну мережу для знаходження ключових орієнтирів на обличчі. Вона не потребує маркерів або додаткового обладнання. На вході лтше відео, на виході 468 точок для кожного кадру [12].

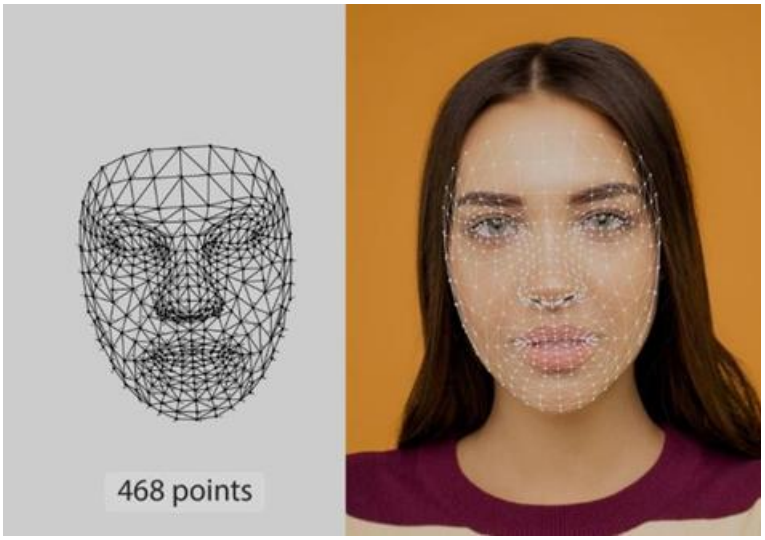


Рисунок 2.4 – Зображення з накладеними точками Face Mesh на обличчя

Для обробки відео застосовується бібліотека OpenCV [13, 25, 41], яка покадрово зчитує кадри, подає їх у Face Mesh, а у відповідь отримує координати точок. Результатом MediaPipe є матеріали для подальших розрахунків міміки. На основі руху ключових точок визначаються значення параметрів для shape keys (наприклад, Blink, Smile, Talk), які анімують тривимірну модель у Blender.

Після отримання координат ключових точок обличчя з кожного кадру відео за допомогою MediaPipe наступним кроком є перетворення цих координат на числові параметри міміки, які відповідають певним виразам обличчя. Ці параметри використовуються як керуючі значення для shape keys в середовищі Blender.

Параметри міміки це нормалізовані значення у діапазоні [0, 1], що описують ступінь прояву певного виразу обличчя, таких як Smile – інтенсивність усмішки, BlinkLeft / BlinkRight – рівень моргання лівим/правим оком, Talk – відкриття рота та інші. Розрахунок базується на евклідових відстанях між певними ключовими точками на обличчі, виділеними MediaPipe. Кожен параметр міміки має свою формулу. Але в цілому, вони схожі. Наприклад для усмішки:

$$Smile = \min(1.0, \max(0.0, (d_{mouth} - T_S) \cdot K_S)), \quad (2.2)$$

де d_{mouth} – відстань між правим і лівим кутами рота;

T_S – матриця 2×2, що задає масштабування й обертання;

K_S – вектор зсуву.

Для інших обчислення такі самі, змінюються лише значення та назви зміних. Далі усі параметри збираються в один пакет і надсилаються у Blender через сокет. У Blender ці значення приймаються Python-скриптом, який змінює відповідні shape keys на тривимірній моделі.

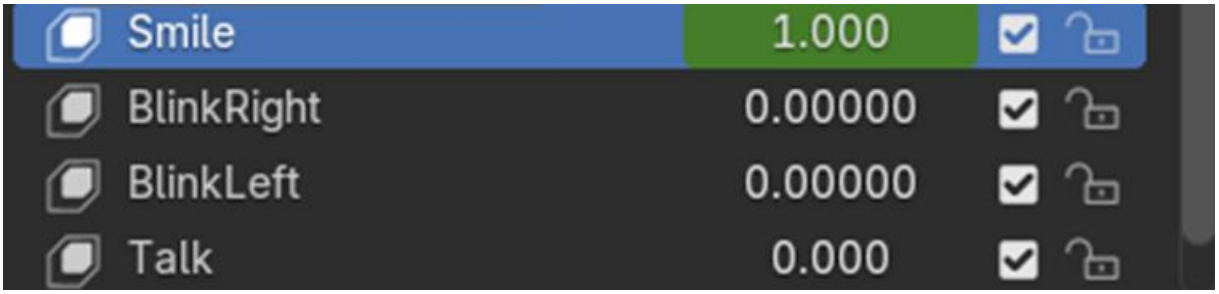


Рисунок 2.5 – Список Shape keys для моделі

2.2 Оцінка якості передачі міміки на тривимірну модель

Оцінка якості передачі міміки на тривимірну модель є достатньо важливим етапом при створенні системи, яка використовує комп’ютерний зір для анімації обличчя в реальному часі. У цьому підрозділі розглядаються два

ключові напрями, такі як візуальна точність передачі емоцій та затримки в системі, а також вплив зовнішніх умов, таких як освітлення і ракурс.

Одним із важливих параметрів є latency, це час, що минає від моменту появи емоції в кадрі до її відображення на тривимірну моделі. Затримка вимірюється в мілісекундах або секундах, і може бути викликана обробкою зображення (MediaPipe), передачею даних через сокети, оновленням shape keys у Blender. У більшості тестів затримка системи становила 20–80 мс, що сприймається користувачем як майже реальний час. Для вимірювання затримки було використано запис відео з годинником і визначив час між подією та реакцією моделі на записі.

Одним із важливих критеріїв якості системи передачі міміки є узгодженість між згенерованим відео, отриманим від FOMM, та анімацією тривимірної моделі у Blender, яка створюється на основі shape keys. Узгодженість визначає, наскільки точно поведінка обличчя на відео відтворюється тривимірною моделлю. Це означає, що темп і час змін міміки у Blender відповідає темпу у відео, тип міміки (усмішка, моргання, відкриття рота) правильно визначається та передається, інтенсивність виразу обличчя збігається або близька до тієї, що на відео, затримки та викривлення мінімальні або непомітні для людського ока [11].

Для оцінки узгодженості використовується візуальне порівняння відео, згенерованого FOMM, з виводом анімованої тривимірної моделі. Такий аналіз дозволяє побачити розбіжності між рухом рота та частотою та амплітудою моргань.

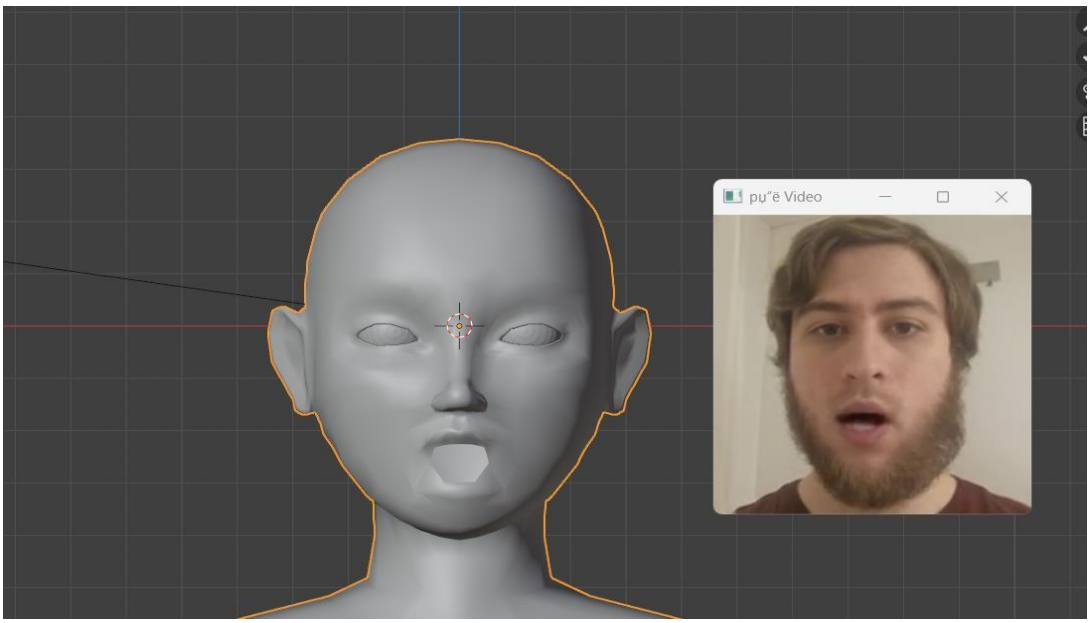


Рисунок 2.6 – Праворуч – кадр з FOMM, ліворуч – аналогічний момент у Blender.

Невідповідність може бути спричинена неправильним розрахунком параметрів міміки (наприклад, надто жорсткий поріг), втратами кадрів при обробці відео, відсутністю згладжування, що призводить до ривків, неправильним масштабуванням значень shape keys або неможливістю Blender точно повторити нюанси FOMM-анімації через обмежену кількість shape keys [3] .

Для підвищення якості можна зробити згладжування руху міміки (див. розділ 2.4), або покадрову синхронізацію, тобто визначення кадрів у FOMM–відео, де відбуваються ключові мімічні події (наприклад, початок усмішки), і відповідна прив’язка у Blender з подальшим калібруванням shape keys та встановлення правильних меж і масштабування значень або використання графічного накладання обох відео для порівняння [7, 8].

Стабільність системи передачі міміки має вирішальне значення для її практичного застосування. Одним із основних викликів є здатність моделі коректно працювати при різних зовнішніх умов, зокрема освітлення та ракурсу обличчя. Цей підпункт зосереджується на експериментальній перевірці стійкості роботи системи в різних умовах, що часто зустрічаються у реальному використанні. Освітлення є критичним фактором для роботи комп’ютерного

зору. У випадку FOMM воно впливає як на генерацію анімації, так і на стабільність трекінгу міміки після обробки відео через MediaPipe.

Тестувалися три сценарії, такі як м’яке фронтальне освітлення, рівномірне світло без тіней (еталон), бокове освітлення, яке утворює напівтінь з одного боку та заднє освітлення, підсвітка позаду користувача, що створює контрове світло.

Рисунок 2.7 – Порівняння скріншотів у трьох умовах освітлення.

Результати показали, що у сценарії 1 система відпрацьовує міміку найточніше (усмішка, моргання відслідковуються стабільно). У сценарії 2 точність знижується через часткове затемнення ключових зон обличчя (наприклад, очей). У сценарії 3 найвищий рівень помилок. MediaPipe не фіксує частину точок, а shape keys поводяться нестабільно.

Також, для оцінки чутливості до ракурсу було проведено тестування з чотирма варіантами положення голови:0, 30, 45 та 90 градусів.

У фронтальному ракурсі все працює стабільно. При нахилі обличчя понад 45°, частина точок зникає з поля огляду, особливо око, протилежне камері. Тобто, оптимальні умови це пряме м’яке світло та фронтальний ракурс (0°);

У сфері анімації обличчя за допомогою комп’ютерного зору існує низка готових рішень, які використовуються для передачі міміки на тривимірній моделі. Щоб оцінити ефективність запропонованого підходу, було проведено порівняння з найпоширенішими з них: Faceware, Live Link Face (Apple) та DeepMotion у таблиці 2.1.

Таблиця 2.1 – Порівняння з існуючими рішеннями

Система	**Вхідні дані**	**Платформа**	**Реальний час**	**Відкритий код**	**Blender-сумісність**	**Примітка**
Faceware Live	Відео, камера	Windows/ macOS	Так	Ні	Через плагін	Платна, професійне рішення
Live Link Face	iPhone TrueDepth	iOS	Так	Ні	Через Unreal Bridge	Висока точність,але закрите
DeepMotion	Відео	Онлайн-сервіс	Ні (асинхронно)	Ні	Експорт у FBX/JSON	Придатне для запису
Ця система	Відео	Python+ FOMM Blender	Так	Так	Так (пряме керування shape keys)	Безкоштовна кастомна

Основними перевагами реалізованої системи є її повна безкоштовність та використання інструментів з відкритим вихідним кодом, зокрема таких як First Order Motion Model (FOMM), MediaPipe, Blender і мова програмування Python. Це забезпечує широкі можливості для адаптації, масштабування та розповсюдження розробки без фінансових витрат. Однією з ключових функцій є пряма передача міміки у реальному часі з відео або анімованого зображення до 3D-моделі у Blender без необхідності у використанні додаткових плагінів або складних інструментів. Окрім цього, система відзначається гнучкістю, тобто кожен компонент, починаючи від алгоритму обробки координат до механізму візуалізації, можна змінювати або вдосконалювати залежно від потреб користувача чи вимог проєкту. Одночасно слід відзначити, що професійні рішення мають більш точну передачу мікровиразів, а також кращу стабільність у складних умовах, наприклад, при нахилах голови або слабкому освітленні.

2.3 Візуалізація міміки у Blender за допомогою shape keys

Фінальним етапом реалізації системи передачі міміки є візуалізація у середовищі Blender шляхом зміни значень відповідних shape keys. Саме цей процес забезпечує анімоване відображення міміки на тривимірній моделі обличчя в реальному часі. Shape keys є ключовим механізмом [2] , що дозволяє змінювати геометрію сітки моделі без використання кісток або додаткових модифікаторів.

Для реалізації системи анімації міміки у Blender першочерговим завданням є створення базового набору shape keys, які відповідають основним виразам обличчя. Ці shape keys виступають як канали для передачі параметрів міміки у тривимірний простір.

На основі аналізу міміки у відео та особливостей системи обробки даних (MediaPipe + FOMM) було виділено такі базові вирази, кожен з яких відповідає одному shape key у таблиці 2.2:

Таблиця 2.2 – Базові вирази, кожен з яких відповідає одному shape key

Дія	**Назва shape key**	**Призначення**
Усмішка	Smile	Розширення рота в боки
Моргання лівим оком	BlinkLeft	Опускання верхньої повіки зліва
Моргання правим оком	BlinkRight	Опускання верхньої повіки справа
Розмова (відкриття рота)	Talk	Відкриття роту

Процес створення shape keys починається з виділення тривимірного об’єкта голови персонажа. Далі в інтерфейсі Blender у вкладці Object Data Properties відкривається панель Shape Keys. Якщо це перше редагування об’єкта, спочатку створюється базова форма (Basis), яка слугує початковою геометрією. Після цього додаються нові shape keys для кожної потрібної міміки, наприклад, Smile, BlinkLeft, BlinkRight, Talk.

Для редагування конкретного shape key необхідно перейти в режим редагування (Edit Mode) й вручну змінити положення відповідних вершин моделі, які відповідають за окремі частини обличчя, такі як губи, повіки тощо. Після завершення редагування повертаються до Object Mode, де перевіряють плавність переходу між станами, змінюючи значення ключа у властивостях від 0 до 1. Після створення кожного ключа варто провести перевірку його роботи. Для цього змінюють його значення вручну в інтерфейсі Blender і спостерігають за реакцією моделі. При цьому важливо уникати надмірних деформацій, які виглядають неприродно (наприклад, наскрізного розкриття рота або зміщення очей поза межі орбіти при значенні Talk = 1.0). Таким чином, ручне налаштування shape keys дозволяє досягти високої якості та точності анімації на моделі.

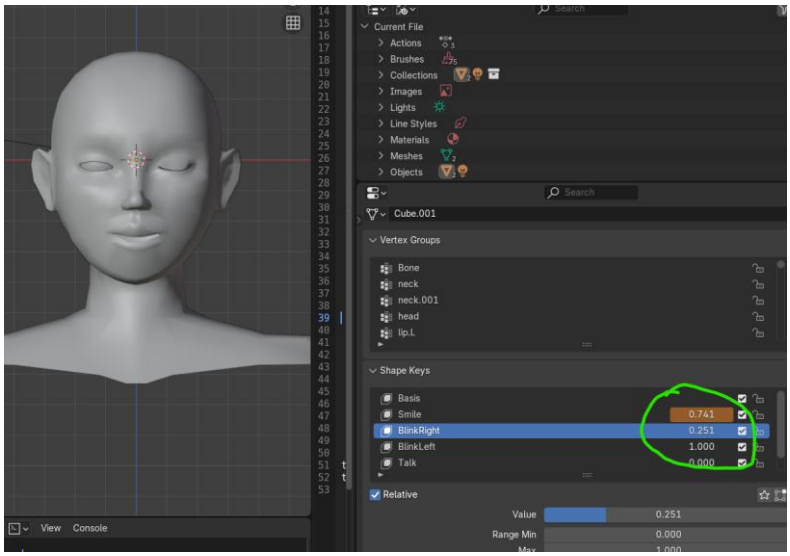


Рисунок 2.8 – Приклад змін значення параметрів shape keys

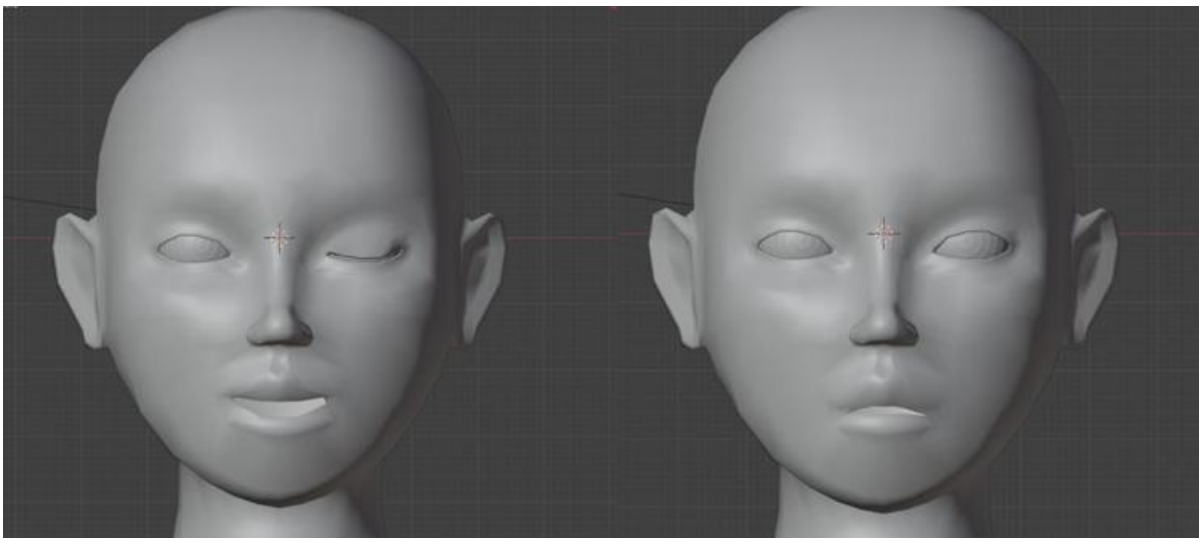


Рисунок 2.9 – Різниця між нейтральним положенням (друге) та зміненим (перше)

На зображенні 2.7 показано поточні значення параметрів shape keys у Blender, які відповідають за передачу міміки. Значення Smile становить 0.741, що свідчить про активацію усмішки. BlinkRight дорівнює 0.251, отже, праве око трохи відкрите. У той час як BlinkLeft має значення 1, що вказує на повне моргання лівим оком. Значення Talk становить 0, тобто рот повністю закритий.

Для динамічного керування мімікою тривимірної моделі в Blender було реалізовано передачу даних через UDP-сокет. Основна ідея полягає в тому, щоб розділити процес аналізу міміки (який здійснюється окремою програмою) і саму анімацію в Blender. Скрипт, що обробляє відео або оживлене зображення, за допомогою бібліотек socket та struct надслає вектор параметрів міміки у вигляді 4-х значень типу float: інтенсивність усмішки, відкриття рота, моргання лівого та правого ока. Ці значення оновлюються кожні 0.1 секунди для забезпечення плавності, і передаються через локальне з’єднання на порт 5005. У Blender одночасно запускається фоновий скрипт, який створює сервер [15]. Він приймає цей потік даних у реальному часі, розпаковує байти назад у числові значення і записує їх у відповідні shape keys моделі. Кожен параметр відповідає строго іменованому shape key: наприклад, значення усмішки записується у ключ Smile, моргання у BlinkLeft або BlinkRight, а рух рота у Talk. Це дозволяє досягти майже миттєвого відгуку тривимірної моделі на зміну міміки у відео.

Рисунок 2.10 – Графічне зображення архітектури

Подібна архітектура активно використовується в задачах синхронізації анімації у реальному часі та керування тривимірними персонажами. Вона має високу гнучкість, добре масштабується і дозволяє швидко тестувати нові моделі без модифікації коду Blender [2, 16].

Автоматичне оновлення shape keys у Blender реалізується за допомогою вбудованого Python-інтерпретатора та багатопотокового обробника вхідних даних. Для цього у середовищі Blender запускається фоновий скрипт, який слухає локальний UDP-порт і в режимі реального часу приймає 4 значеня типу float, що відповідають основним параметрам міміки, такими як усмішка, моргання лівим і правим оком, рух рота. Скрипт створює окремий потік, щоб не блокувати інтерфейс Blender. Після запуску він ініціює сокет на порту 5005 і виконує пошук тривимірного об’єкта в сцені, який містить набір shape keys з назвами, що відповідають очікуваним параметрам. Якщо всі ключі знайдені, то в процесі роботи скрипт починає приймати пакети з 4-х значень, які передаються у форматі struct (20 байт загалом). Кожне прийняте значення одразу ж призначається відповідному shape key, змінюючи геометрію обличчя моделі.

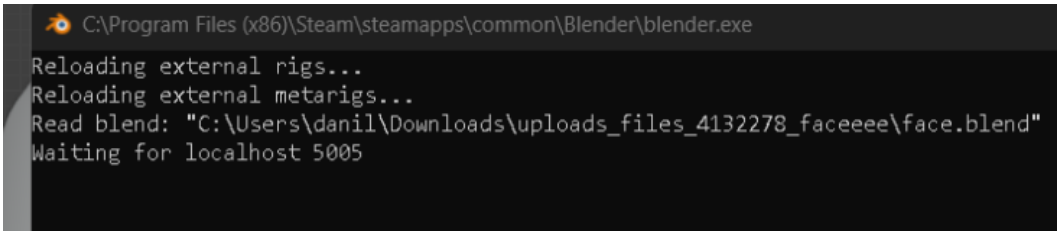


Рисунок 2.11 – Очікування на підключення

Оскільки передача відбувається з частотою приблизно 10 разів на секунду, це забезпечує плавну анімацію, що синхронізована з мімікою на відео або оживленому зображенні. Наприклад, якщо значення параметра усмішки дорівнює 0.75, відповідний shape key «Smile» активується на 75%, що призводить до підняття кутиків рота. Аналогічно обробляються моргання та інші вирази. У випадку припинення зв’язку або помилки, усі значення shape keys скидаються до нуля, щоб уникнути залипання міміки на моделі [8, 21].

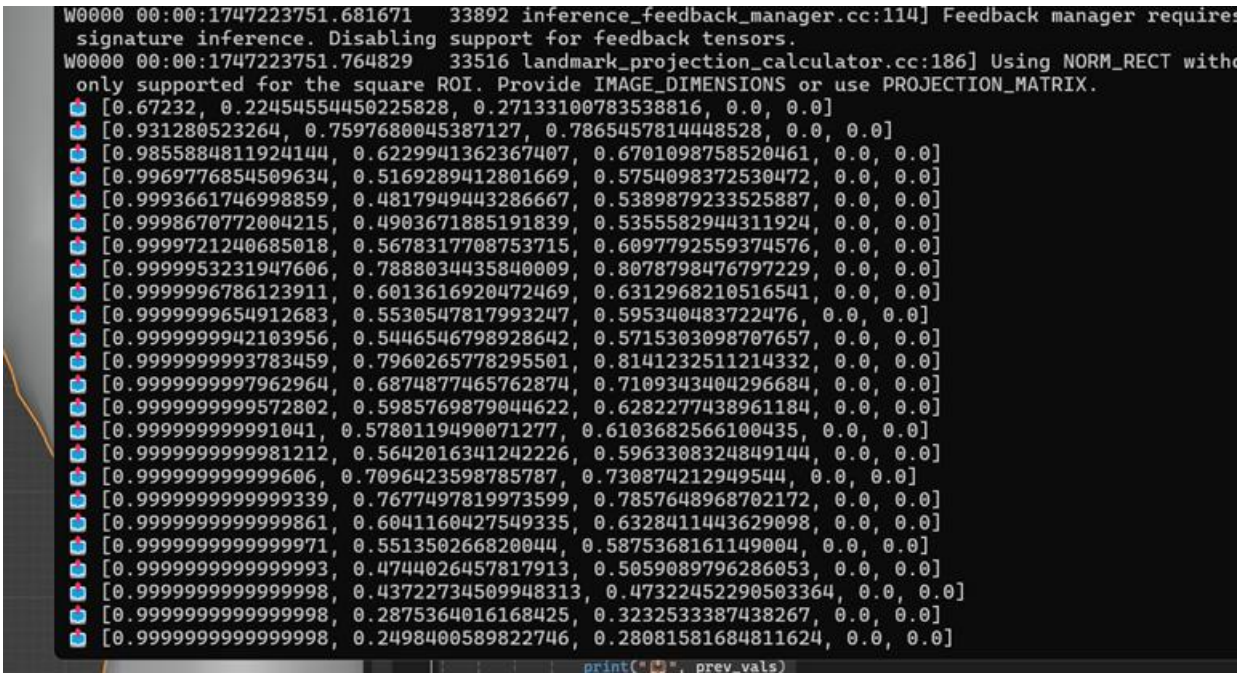


Рисунок 2.12 – Прийняті дані

У процесі автоматичної анімації обличчя за допомогою shape keys у Blender важливо не лише передавати значення параметрів міміки, але й забезпечити їхнє правильне масштабування, щоб деформації виглядали природно. Значення, які надходять від системи аналізу міміки (наприклад, через MediaPipe або FOMM), можуть суттєво варіюватися залежно від типу обличчя, положення камери, розміру відео або освітлення. Без попереднього нормування навіть незначне відкриття рота може призвести до повного спрацювання shape key, що виглядатиме неприродно або навіть деформує модель.

Щоб уникнути цього, для кожного параметра міміки визначаються базові порогові значення, на основі яких розраховується нормалізоване значення. Така лінійна нормалізація виконується за формулою:

$$x_{norm} = \min\left(1.0, \max\left(0.0, \frac{x-T}{S}\right)\right), \quad (2.3)$$

де x – виміряне значення;

T – початкова межа, після якої починається активація деформації;

S – коефіцієнт масштабування.

Ця формула дозволяє обрізати надмірно малі або великі значення і забезпечити контрольовану активацію міміки. Наприклад, для усмішки порогом може бути різниця координат між кутиками губ понад 0.10, а масштабуванням це множення на 40. Це означає, що лише при достатньо широкій усмішці shape key буде активовано більш ніж на половину. Крім цього, щоб уникнути ривків і залипання виразів, застосовується згладжування значень за допомогою інтерполяції між поточним і попереднім значенням. Це дозволяє зробити зміну міміки більш плавною, навіть якщо значення з відео змінюються різко.

2.4 Фільтрація та згладжування параметрів міміки

Зчитування міміки з відео або згенерованої анімації завжди супроводжується шумами, стрибками значень і мікродеформаціями, які можуть спотворити поведінку тривимірної моделі. Особливо це стосується швидких рухів обличчя, змін освітлення, втрати фокуса, часткового перекриття елементів обличчя, а також технічних обмежень самої камери чи алгоритму виявлення ключових точок [8]. Щоб уникнути ривків у міміці або неконтрольованих змін параметрів, система потребує фільтрації та згладжування сигналів перед тим, як передати їх у Blender.

Одним із найбільш ефективних методів фільтрації короткочасних шумів є ковзне середнє (moving average), яке дозволяє приглушити випадкові стрибки значень і водночас зберегти основну динаміку виразу обличчя. Суть методу полягає в тому, що кожне нове значення параметра розраховується як середнє арифметичне з кількох останніх спостережень, які накопичуються у фіксованому буфері. Наприклад, якщо враховуються останні п’ять значень параметра «Smile», то кожне нове оновлення цього параметра є результатом усереднення попередніх п’яти значень. Такий підхід дозволяє суттєво зменшити візуальну динаміку випадкових коливань, що не відповідають справжнім мімічним змінам, але водночас зберігає природність реакції моделі. Особливо ефективним це згладжування є у ситуаціях, коли система аналізує відео з нестабільним фокусом або змінним освітленням. У таких випадках точність розпізнавання координат ключових точок може коливатись, але ковзне середнє компенсує ці коливання, перетворюючи різкі переходи у більш плавні. Наприклад, параметр моргання не буде миттєво переходити від 0 до 1, якщо на одному з кадрів повіка випадково зникла або зрушилася, а замість цього відбудеться поступове наростання чи зменшення значення. Це дозволяє уникнути фальшивих моргань або непередбачуваного смикання очей у тривимірній моделі.

У цій реалізації системи передачі міміки передбачено механізм згладжування значень параметрів обличчя, щоб уникнути ривків і зробити зміну виразів максимально природною. Це здійснюється через функцію лінійної інтерполяції, яку реалізовано у вигляді окремої функції lerp() [17]. Вона дозволяє обчислювати нове значення параметра як плавний перехід між попереднім значенням і новим, отриманим з відео.

```
alpha = 0.2
prev_vals = [lerp(prev_vals[i], values[i], alpha) for i in range(5)]
```

Рисунок 2.13 – Згладжування значень параметрів міміки

I

В цій роботі було використано коефіцієнт згладжування alpha = 0.2, тобто кожне нове значення shape key оновлюється лише на 20% у напрямку до актуального значення, тоді як решта 80% зберігається від попереднього. Це дозволяє уникнути різких стрибків, які часто виникають при роботі з відео, особливо при мікрорухах або зміні освітлення. Такий підхід не лише покращує візуальне сприйняття, а й забезпечує стабільність анімації у Blender при використанні даних, згенерованих нейронною мережею або зчитаних за допомогою MediaPipe. Такий підхід виявився ефективним і дозволив уникнути ривків у міміці, зробивши анімацію більш живою та стійкою до помилок у вході. Візуально це виглядало так, ніби обличчя м'яко повертається до нейтрального стану, а не скаче назад. У своїй системі передачі міміки реалізовано механізм згладжування значень shape keys за допомогою коефіцієнта інерції, який позначається як alpha на рис 2.10. Цей коефіцієнт відповідає за те, наскільки швидко або повільно модель реагує на зміну вхідних параметрів, що надходять із відео або згенерованої анімації.

Завдяки цьому підходу анімація міміки у Blender виглядає не ривчасто, а плавно та природно.

$$x_t = (1 - \alpha) \cdot x_{t-1} + \alpha \cdot x_{new}, \quad (2.4)$$

де x_t – поточне значення параметра міміки на момент часу;

x_{t-1} – попереднє значення цього ж параметра на попередньому кроці;

x_{new} – нове значення параметра;

α – коефіцієнт масштабування.

3 ПРОГРАМНЕ МОДЕЛЮВАННЯ МЕТОДІВ АНІМАЦІЇ МІМІКИ ОБЛИЧЧЯ

У цьому розділі детально розглядається процес реалізації системи анімації міміки обличчя, що побудована на поєднанні нейронної моделі оживлення зображень (FOMM) та передачі параметрів міміки на тривимірну модель у середовищі Blender. Основна мета полягає у створенні ефективного програмного рішення, яке дозволяє аналізувати відео міміки автоматично визначати ключові параметри обличчя і передавати їх у вигляді числових значень для керування shape keys моделі.

3.1 Обґрунтування вибору програмного середовища

У процесі реалізації системи передачі міміки обличчя у Blender було використано середовище програмування Visual studio, Python, бібліотеки MediaPipe, OpenCV, socket-програмування, нейромережеву модель FOMM, а також середовище тривимірної графіки Blender. Усі ці інструменти є відкритими, добре задокументованими й активно використовуються у завданнях комп’ютерного зору та генерації зображень.

Було обрано середовище Visual Studio, оскільки воно є одним із найпотужніших і найзручніших інструментів для розробки програмного забезпечення. Visual Studio забезпечує стабільну роботу з великими проєктами, має зручний інтерфейс, вбудовану систему налагодження (debugging), автодоповнення коду (IntelliSense), а також підтримує роботу з різними мовами програмування, зокрема Python, C#, C++. Важливою перевагою є можливість легкої інтеграції з бібліотеками та зовнішніми компонентами, такими як OpenCV, MediaPipe або інструментами нейронних мереж. Крім того, Visual Studio підтримує створення графічних інтерфейсів користувача, роботу з потоками, мережею, а також має вбудовані засоби для тестування та аналізу програм.

Worldwide, May 2025 :

Rank	Change	IDE	Share	1-year trend
1		Visual Studio	28.78 %	+1.5 %
2		Visual Studio Code	15.24 %	+1.5 %
3	↑↑	Android Studio	11.22 %	+1.3 %
4		pyCharm	10.27 %	-0.6 %
5	↓↓	Eclipse	10.09 %	-1.8 %
6		IntelliJ	7.16 %	-0.4 %
7		NetBeans	3.56 %	-0.5 %
8		Xcode	2.97 %	-0.0 %
9		RStudio	2.85 %	-0.1 %
10		Sublime Text	2.34 %	-0.2 %

Рисунок 3.1 – Топ 10 найпопулярніших IDE

Використання Visual Studio дозволило організувати проєкт у вигляді зручного рішення, створити зручну структуру коду та швидко переходити між частинами реалізації. Це особливо важливо при побудові складних систем, де задіяні одразу кілька технологійБ, від відеообробки до передачі даних у Blender. Завдяки широкій підтримці та стабільності Visual Studio стала оптимальним вибором для виконання даної кваліфікаційної роботи.

Мова програмування Python була обрана як основна завдяки її універсальності, великій кількості бібліотек для роботи з відео і зображеннями, нейромережами та графікою. Python має високорівневий, інтерпретований синтаксис, що дозволяє швидко розробляти прототипи й автоматизувати завдання. Також він підтримує роботу з сокетами, що стало критично важливим для передачі даних між Python і Blender у режимі реального часу [17].

MediaPipe від Google потрібна для аналізу міміки на основі відео. Вона дозволяє виділяти ключові точки обличчя, зокрема ті, що відповідають за очі, губи та інше. Для кожного кадру система зчитує координати цих точок і формує параметри міміки, такі як усмішка, моргання, відкриття рота тощо, які потім

передаються в Blender через socket-з'єднання. Завдяки високій точності MediaPipe, навіть при варіаціях освітлення або ракурсу ключові точки залишаються стабільними [12].

FOMM (First Order Motion Model) – це нейромережева модель, яка оживляє статичне зображення, використовуючи рух з відео. Вона була інтегрована у систему як попередній крок до візуалізації у Blender. З її допомогою вдалося автоматично згенерувати відео з мімікою на базі обраного зображення користувача. Цей підхід дозволив уникнути використання камери або захоплення руху в реальному часі, що є значною перевагою при обмежених ресурсах 7.

Blender виступає як кінцева платформа для візуалізації. Його внутрішній Python API дає змогу отримувати й обробляти параметри міміки через shape keys. За допомогою UDP-сокетів анімовані значення надсилаються в Blender у реальному часі. Shape keys – це деформовані стани об'єкта, які відображають різні емоції персонажа. На основі даних, отриманих з MediaPipe, значення shape keys постійно оновлюються, і в результаті модель передає міміку з відео.

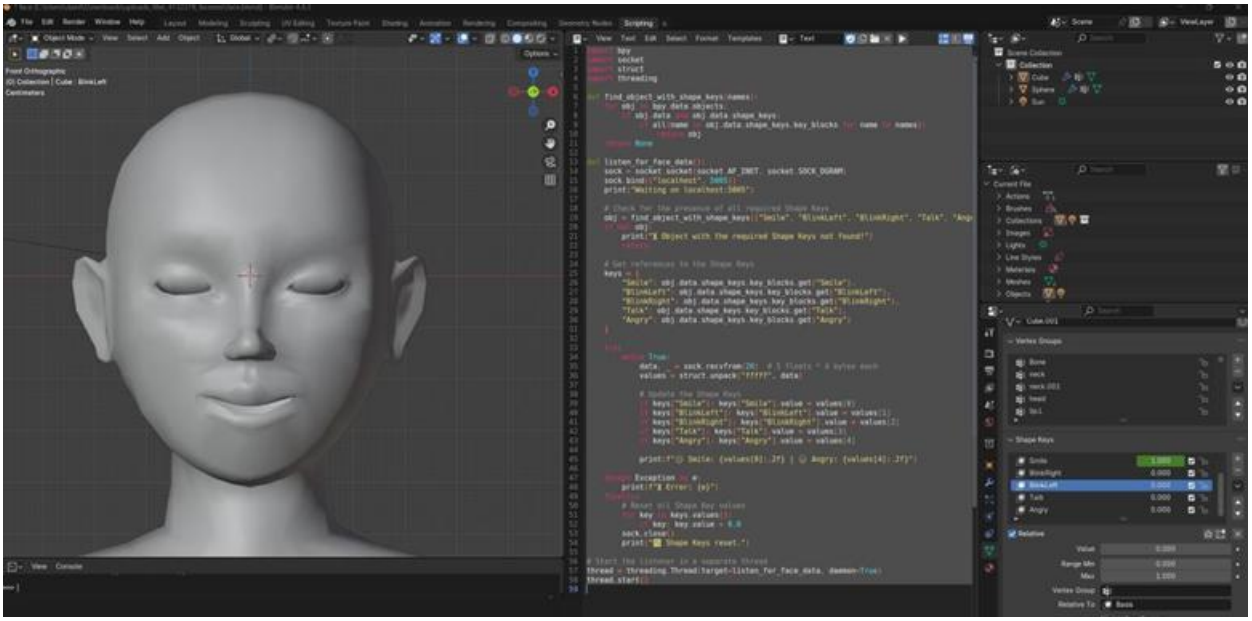


Рисунок 3.1 – Приклад інтерфейсу Blender

Також було використано бібліотеку OpenCV, для роботи з відео, таких як відкриття, зчитування кадрів та попередньої обробки зображень. OpenCV

підтримує основні формати відео та інтегрується з MediaPipe, що дозволило реалізувати трекінг без зайвих накладок.

Для візуалізації значень параметрів міміки під час розробки застосовувалась бібліотека Matplotlib, зокрема для побудови графіків зміни shape keys з часом. Це дало змогу візуально оцінити вплив фільтрації та точність передачі емоційних змін.

3.2 Особливості програмної реалізації

Для зручності взаємодії з програмним забезпеченням було реалізовано окремий графічний інтерфейс користувача, побудований із використанням стандартної бібліотеки мови Python-tkinter. Такий підхід дає змогу суттєво спростити процес керування всією системою, зробивши його більш інтуїтивно зрозумілим і доступним навіть для користувача без досвіду роботи з подібними програмами. Завдяки графічному інтерфейсу забезпечується чітка послідовність дій, а також зменшується ймовірність помилок при введенні параметрів або виборі файлів.

Реалізація власного GUI дозволяє розділити робочий процес на логічні етапи, кожен з яких представлений окремим інтерактивним елементом. Зокрема, інтерфейс містить кнопку для вибору зображення, яке буде оживлено, кнопку для завантаження відео, що містить міміку, яку необхідно перенести, а також кнопку для запуску нейронної мережі FOMM, саме вона відповідає за генерацію анімованого обличчя на основі вхідних даних. Крім того, в інтерфейсі присутні кнопки, що відповідають за запуск і зупинку системи трекінгу, яка аналізує міміку та передає її у реальному часі у середовище Blender.

Рисунок 3.2 – GUI програми

Після кожного вибору або запуску виводиться повідомлення про успішне виконання операції, це реалізовано за допомогою діалогових вікон messagebox.showinfo(), які підвищують зручність та прозорість взаємодії. У випадку помилки або відсутності необхідних файлів виводяться попередження через messagebox.showerror(), що дозволяє користувачеві швидко реагувати на проблеми. Приклади повідомлень приведені нижче:

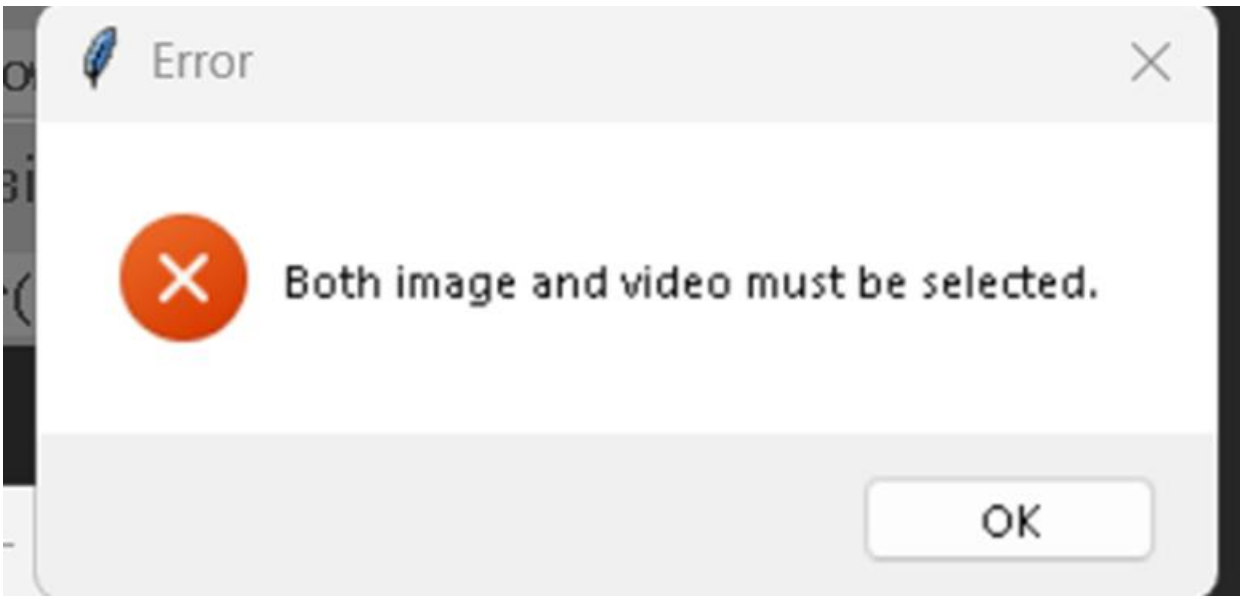


Рисунок 3.3 – Повідомлення про не успішне виконання операції вибору фото

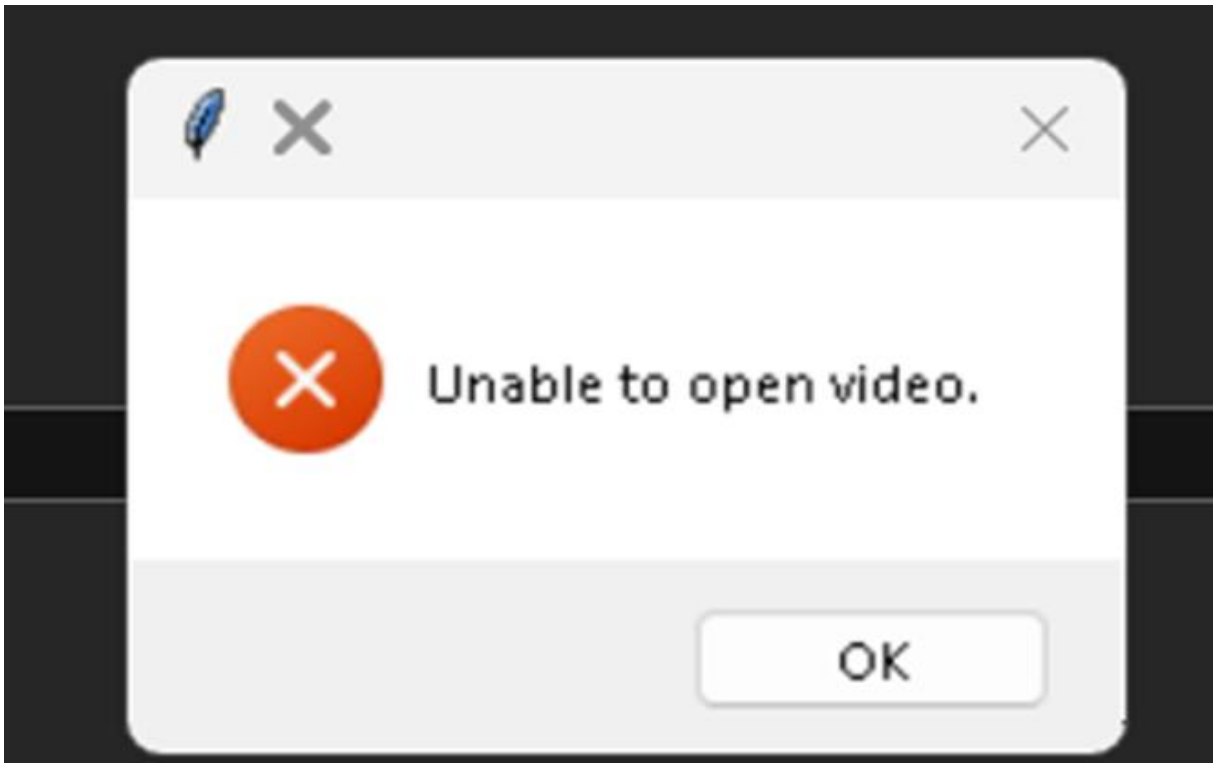


Рисунок 3.4 – Повідомлення про не успішне виконання операції вибору відео

Для завантаження вхідних даних у систему було реалізовано два окремі інтерактивні елементи: кнопку для вибору зображення та кнопку для вибору відео. Обидві функції засновані на виклику стандартного діалогового вікна filedialog.askopenfilename() з бібліотеки tkinter. При натисканні кнопки

відкривається вікно вибору файлу, де користувач може завантажити зображення у форматі .jpg, .png або .jpeg.

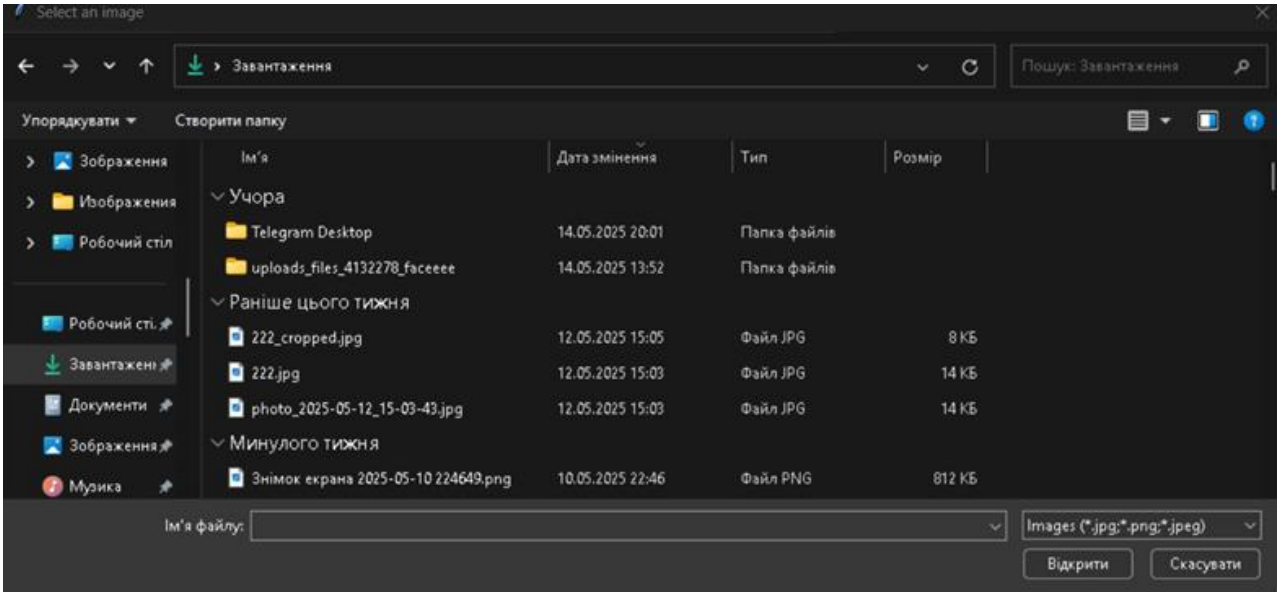


Рисунок 3.5 – Вікно вибору фото

Після вибору на екрані з’являється інформаційне повідомлення з підтвердженням шляху до зображення.

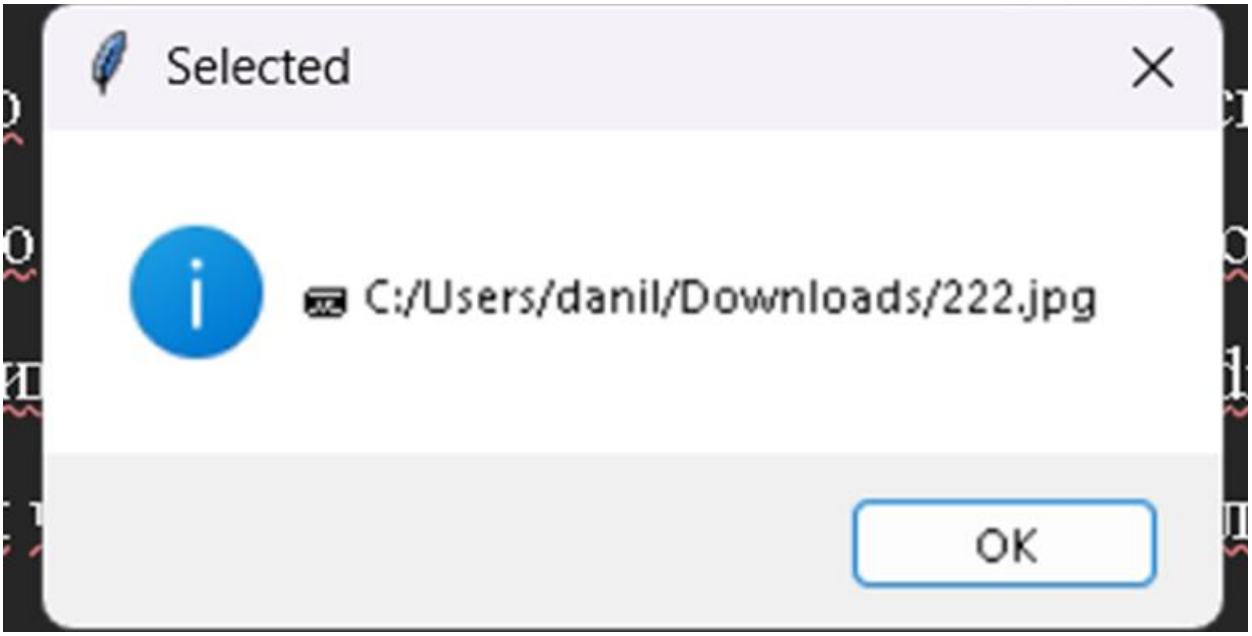


Рисунок 3.6 – Повідомлення про успішне виконання операції вибору фото

Обраний шлях зберігається у змінну self.image_path, яка використовується пізніше як аргумент під час запуску FOMM. Аналогічним чином працює кнопка для відео. Вона відкриває діалог вибору відеофайлу у форматах .mp4, .avi, .mov, .mkv.

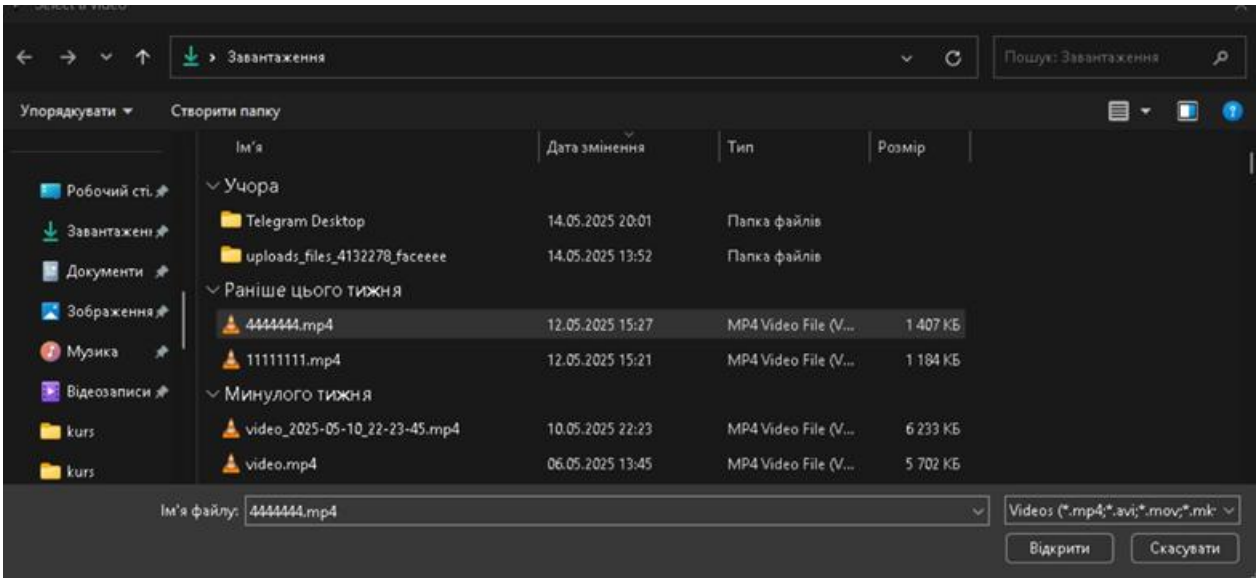


Рисунок 3.7 – Вікно вибору відео

Вибране відео зберігається у змінну self.video_path і використовується для генерації міміки за допомогою нейромережі FOMM [20].

Рисунок 3.8 – Повідомлення про успішне виконання операції вибору відео

Наступним етапом у створенні системи передачі міміки стало застосування моделі FOMM, яка дозволяє оживити статичне зображення обличчя шляхом переносу міміки з відео. У нашому випадку FOMM викликається з графічного інтерфейсу за допомогою кнопки запуску FOMM.

При натисканні цієї кнопки формується системна команда, яка через subprocess.run() [21] викликає Python-скрипт demo.py, передаючи параметри конфігурації, модель і шляхи до зображення та відео. Щоб не блокувати інтерфейс під час тривалих обчислень, запуск FOMM реалізовано у фоновому потоці з використанням threading.Thread, що дозволяє зберегти чутливість GUI і уникнути ефекту «замерзання» застосунку. У випадку успішного завершення генерації користувач отримує сповіщення, а згенероване відео зберігається у тій самій директорії, що й оригінальні дані. Його подальша обробка відбувається у підсистемі MediaPipe.

На рисунку 3.17 наведено приклад процесу генерації анімації в терміналі, що свідчить про поетапне проходження кадрів через модель. Варто зазначити, що швидкість виконання залежить від конфігурації комп'ютера, а також того, чи використовується обчислення на CPU чи GPU.

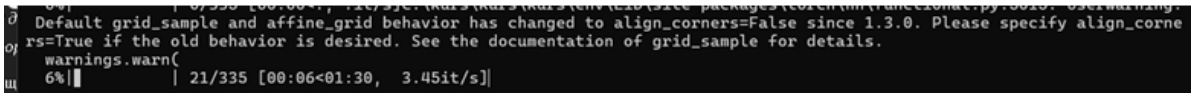


Рисунок 3.9 – Процес роботи FOMM

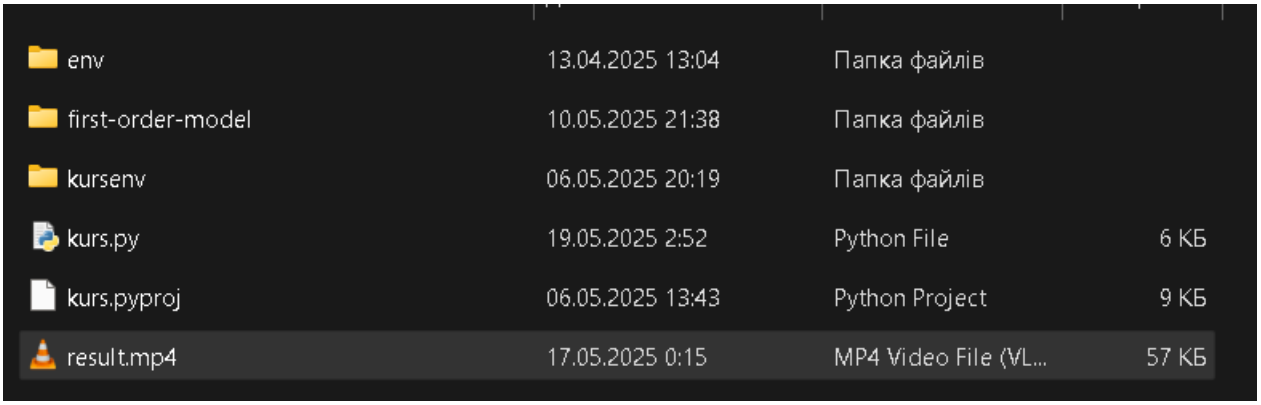


Рисунок 3.10 – Кінець роботи FOMM та кінцевий файл result.mp4

Після генерації анімованого відео з мімікою за допомогою нейронної мережі FOMM наступним етапом є його обробка з метою отримання параметрів міміки. Для цього застосовується бібліотека MediaPipe, яка дозволяє з відеозапису отримувати координати ключових точок обличчя. Зокрема,

використовується модуль face_mesh, який автоматично визначає положення 468 анатомічних точок на обличчі користувача, у тому числі області очей, рота, брів і контурів обличчя. Аналіз відео починається з покадрової обробки. Кожен кадр відео конвертується у формат RGB, оскільки алгоритми MediaPipe працюють саме з цим колірним простором. Після цього запускається процедура виявлення обличчя, і якщо розпізнавання пройшло успішно, система зчитує координати ключових точок. Наприклад, для виявлення посмішки вимірюється горизонтальна відстань між точками 61 (лівий край рота) і 291 (правий край рота). Для оцінки моргання вимірюється вертикальна відстань між точками верхнього та нижнього повік, наприклад, 386 і 374 для лівого ока та 159 і 145 для правого. Далі визначені відстані (таблиця 3.1) нормалізуються та масштабуються до діапазону від 0 до 1 (2.4).

Таблиця 3.1 – Відповідність міміки та точок обличчя

Міміка	Точки обличчя	Shape Key у Blender
Усмішка	61, 291	Smile
Моргання Л	386, 374	BlinkLeft
Моргання П	159, 145	BlinkRight
Розмова	13, 14	Talk

Після обробки відео з мімікою та розрахунку числових значень параметрів, таких як посмішка, моргання, рух рота, постає завдання передати ці значення у тривимірне середовище Blender для анімації обличчя. Для цього використовується механізм сокет-з’єднання на основі UDP (User Datagram Protocol), що дозволяє у реальному часі надсилати дані з програми до Blender.

Передача відбувається за допомогою парисунку 4-х чисел з плаваючою комою (тип float32), упакованих у двійковий формат через функцію struct.pack(„fffff“, ...), що є компактною формою передачі, зручною для обробки у Blender. Ці значення відповідають чотирьом ключовим мімічним станам: Smile, BlinkLeft, BlinkRight, Talk. На боці Blender запускається скрипт

[15], який відкриває сокет на порту 5005 і постійно очікує вхідні пакети. Як тільки нові дані отримані, скрипт розпаковує їх (struct.unpack(„fffff“, data)) та оновлює відповідні shape key на об’єкті тривимірної моделі. Таким чином, модель «оживає» відповідно до зчитаної з відео міміки [20].

```
def run_tracking(self):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    blender_address = ("localhost", 5005)
```

Рисунок 3.11 – Підключення сокету на стороні програми

```
def listen_for_face_data():
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind(("localhost", 5005))
    print("Waiting on localhost:5005")
```

Рисунок 3.12 – Підключення сокету на стороні Blender

Перевагою UDP є мінімальна затримка порівняно з TCP, що важливо для синхронної анімації обличчя. Водночас цей протокол не гарантує доставку пакетів, тому у реалізації передбачено механізм згладжування останніх отриманих значень, що дозволяє зменшити візуальні стрибки у разі втрати кадру. За потреби кожне з отриманих значень проходить додаткову нормалізацію або обмежується в межах [0, 1], що відповідає припустимому діапазону деформації shape keys у Blender. Далі модель оживає у Blender завдяки тому, що її міміка оновлюється за допомогою shape keys, значення яких змінюються згідно з даними, отриманими з відео.

```
if keys["Smile"]: keys["Smile"].value = values[0]
if keys["BlinkLeft"]: keys["BlinkLeft"].value = values[1]
if keys["BlinkRight"]: keys["BlinkRight"].value = values[2]
```

Рисунок 3.13 – Отримання даних від програми у блендері

При написанні коду було використовувано деякі патерни проєктування. Наприклад, шаблон «Model–View–Controller» (розділення моделі та представлення).

Шаблон MVC працює в загальній архітектурі взаємодії між даними (міміка з відео), логікою їх обробки (аналіз і згладжування) та візуалізацією (анімація у Blender) [18]. У системі можна умовно виділити такі компоненти, як Model – це дані, отримані з аналізу координат обличчя (MediaPipe) [8], включно з розрахованими значеннями shape keys.

```
if results.multi_face_landmarks:
    lm = results.multi_face_landmarks[0].landmark
    values[0] = max(0.0, min((lm[291].x - lm[61].x - 0.25) * 40, 1.0))
    values[1] = max(0.0, min((0.0225 - abs(lm[386].y - lm[374].y)) * 130, 1.0))
    values[2] = max(0.0, min((0.0225 - abs(lm[159].y - lm[145].y)) * 130, 1.0))
    values[3] = max(0.0, min((abs(lm[13].y - lm[14].y) - 0.01) * 50, 1.0))
```

Рисунок 3.14 – Model – отримані дані про міміку з відео

View – це візуальна анімація міміки у Blender за допомогою shape keys. Файл FaceTrackerApp виконує роль логіки (Controller), який зчитує відео, аналізує міміку, обробляє її, а потім надсилає готові значення через сокет. Скрипт у Blender (listen_for_face_data()) – це представлення (View), що лише відображає, не обчислюючи нічого самостійно

```
    values = struct.unpack("ffff", data)

    if keys["Smile"]: keys["Smile"].value = values[0]
    if keys["BlinkLeft"]: keys["BlinkLeft"].value = values[1]
    if keys["BlinkRight"]: keys["BlinkRight"].value = values[2]
    if keys["Talk"]: keys["Talk"].value = values[3]
```

Рисунок 3.15 – View – отриманя даних у Blender

Використання шаблону MVC дозволяє відокремити розрахункову частину системи від візуальної реалізації. Завдяки цьому код легко підтримується, доповнюється, а також тестується окремо. Наприклад, можна

змінити спосіб візуалізації (наприклад, перейти з Blender на Unity), не змінюючи логіку обробки координат [15, 21].

Далі на черзі шаблон «Observer» (Спостерігач). Шаблон «Observer» застосовується для реалізації механізму сповіщення про зміну стану даних у реальному часі. У системі передачі міміки цей підхід реалізовано через UDP-зв'язок. Скрипт FaceTrackerApp в Python у кожному циклі надсилає нові значення параметрів міміки (Smile, BlinkLeft, BlinkRight, Talk) через сокет, а скрипт у Blender (listen_for_face_data()) приймає ці повідомлення та оновлює відповідні shape keys. Таким чином, Blender є «спостерігачем», який чекає на зміни в стані міміки та реагує на них.

```
sock.sendto(data, blender_address)
```

Рисунок 3.16 – Скрипт FaceTrackerApp надсилає нові значення

```
data, _ = sock.recvfrom(16)
values = struct.unpack("ffff", data)
```

Рисунок 3.17 – Blender приймає повідомлення

Шаблон «Observer» дозволяє відокремити джерело даних (модель аналізу міміки) від підписника (візуалізація в Blender). Це дозволяє обробляти дані незалежно від способу візуалізації. Наприклад, у майбутньому можна реалізувати ще одного «спостерігача» (наприклад, запис у файл або надсилання в іншу програму), не змінюючи код модуля, що генерує міміку [18, 20].

Далі було використано Шаблон «Singleton» (Одинак). Шаблон Singleton використовується в контексті створення UDP-сокета, який передає значення міміки в Blender. У системі створюється лише один екземпляр сокета, що використовується протягом усього часу обробки відео. Повторне створення об'єкта сокета не допускається, замість цього повторно використовується вже

відкритий ресурс. Це реалізація принципу єдиного доступу до мережевого каналу

```
run_tracking(self):
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

Рисунок 3.18 – Створюється сокет

Сокет створюється один раз на початку методу run_tracking() та передається в обробку циклу. Закриття відбувається лише після завершення процесу (self.stop()), що відповідає життєвому циклу Singleton-об'єкта. Такий підхід дозволяє уникнути помилок багатократного відкриття порту, та зберегти стабільність з'єднання. Крім того, він дозволяє централізовано керувати ресурсом (закрити його, якщо щось пішло не так), що є основною перевагою Singleton.

Також можна сказати про шаблон «Thread» (Потік виконання) реалізований у кількох частинах системи для асинхронного виконання завдань, які не повинні блокувати основний інтерфейс користувача або основний потік Blender

```
def run_fomm_threaded(self):
    threading.Thread(target=self.run_fomm, daemon=True).start()
```

Рисунок 3.19 – Асинхронне виконання завдань у FaceTrackerApp

```
thread = threading.Thread(target=listen_for_face_data, daemon=True)
thread.start()
```

Рисунок 3.20 – Асинхронне виконання завдань у Blender

Це дозволяє читати відео, аналізувати міміку і передавати значення в Blender паралельно, не блокуючи інші частини програми.

3.3 Результати оживлення обличчя

У ході тестування системи анімації міміки обличчя було сформовано набір вхідних даних, який складався із зображень користувачів та відеофрагментів з мімікою. Основною вимогою до зображень було чітке фронтальне розташування обличчя, відсутність сильних тіней, перекриттів та розмиття, а також нейтральне освітлення. Формат зображень це .jpg або .png, роздільна здатність в цьому випадку неважливе, головне щоб зображення було у форматі квадрата, що відповідає рекомендованим параметрам нейронної мережі FOMM (First Order Motion Model). Для забезпечення відтворення різних типів міміки було також зібрано кілька відеофайлів з демонстрацією емоцій: усмішки, моргання, здивування та мовлення. Відео у форматі .mp4 з частотою 60 кадрів на секунду та тривалістю 5–15 секунд. На рисунку 3.14 подано приклади зображень користувачів, які використовувались у тестуванні, а також відео, що були використані для генерації міміки.

Рисунок 3.21 – Приклад вхідних зображень та відео користувачів, використані для генерації міміки

У ході експерименту було враховано фактор різноманіття зовнішності користувачів, такі як стать, наявність окулярів або бороди. Такий підхід дозволив оцінити універсальність моделі оживлення. Зокрема, виявлено, що точність відтворення міміки дещо знижується у випадках, коли обличчя має

сильно нестандартні риси або в кадрі присутні додаткові елементи перед обличчям.

У процесі оживлення зображень за допомогою нейронної мережі First Order Motion Model статичне фото обличчя перетворюється на відео з реалістичними рухами, які імітують міміку та рухи людини з відео-джерела. На першому етапі детектор визначає координати ключових точок як на початковому зображенні, так і на кожному кадрі відео. Потім формується поле зсуву, яке описує, як ці точки переміщуються. Далі, генератор створює зображення, у якому обличчя з вихідного фото слідує за рухами, які спостерігались на відео-джерелі.

Після запуску нейронної мережі FOMM у середовищі розробки модель поетапно обробляє вхідне зображення та відео. У терміналі виводиться технічна інформація про перебіг процесу, включаючи кількість оброблених кадрів, час виконання, та попередження бібліотек, які не впливають на точність результату. Це дає змогу розробнику контролювати правильність запуску моделі та бачити, як кадри послідовно проходять через нейронну мережу.

Завдяки цій архітектурі FOMM здатна передавати вирази обличчя та емоції навіть у тих випадках, коли джерельне зображення є статичним.

Найбільш вражаючою особливістю є те, що така анімація не вимагає складного моделювання або рігінгу, усе виконується на основі двовимірної інформації. У роботах [1] було показано, що навіть при мінімальній кількості вихідної інформації FOMM здатна створити переконливу імітацію рухів, зберігаючи текстуру обличчя та його риси. Це робить модель особливо корисною у завданнях реалістичної анімації для усіх цілей.

```
def run_fomm(self):
    if not self.image_path or not self.video_path:
        messagebox.showerror("Error", "Both image and video must be selected.")
        return

    python_path = r"C:\kurs\kurs\kurs\env\Scripts\python.exe"

    command = [
        python_path,
        "first-order-model/demo.py",
        "--config", "first-order-model/config/vox-256.yaml",
        "--checkpoint", "first-order-model/checkpoints/vox-cpk.pth.tar",
        "--source_image", self.image_path,
        "--driving_video", self.video_path,
        "--relative",
        "--adapt_scale",
        "--cpu"
    ]

    try:
        subprocess.run(command, check=True)
        messagebox.showinfo("☑ Done", "Animation complete. You can now send it to Blender.")
    except subprocess.CalledProcessError as e:
        messagebox.showerror("❌ FOMM", f"Error running FOMM:\n{e}")
```

Рисунок 3.22 – Команда для початку роботи FOMM

Метод класу FaceTrackerApp. Він запускається після натискання кнопки Run FOMM у графічному інтерфейсі. Перш ніж запускати модель, метод перевіряє, чи обрав користувач зображення (self.image_path) та відео (self.video_path). Якщо хоча б один шлях не задано, виводиться повідомлення про помилку у вигляді спливаючого вікна (через messagebox.showerror), і метод завершує виконання через return.

Далі в коді формується список аргументів, який передається до функції subprocess.run() для запуску скрипту demo.py із проєкту First Order Motion Model (FOMM). Цей список визначає конфігурацію запуску та параметри, необхідні для генерації анімованого відео.

Першим елементом списку є python_path, що вказує на шлях до інтерпретатора Python, за допомогою якого виконується скрипт. Далі йде demo.py – основний файл FOMM, який відповідає за обробку вхідних даних і побудову відео. Після нього передається параметр --config, що визначає шлях до YAML-файлу з архітектурою нейронної мережі. Наступний аргумент --checkpoint вказує на файл з попередньо навченими вагами моделі, які використовуються під час генерації.

Параметр --source_image містить шлях до статичного зображення обличчя, яке потрібно оживити, а --driving_video – шлях до відео з мімікою, яке слугує джерелом руху. Додатковий аргумент --relative активує режим, у якому модель оперує не абсолютними координатами ключових точок, а їх зміщенням відносно початкового стану. Це дозволяє досягти більшої гнучкості та реалістичності при передачі рухів.

Опція --adapt_scale вмикає адаптивне масштабування рухів, що забезпечує точніше відтворення навіть у випадку різниці розмірів між джерелом і анімованим зображенням. Далі, аргумент --cpu повідомляє моделі, що слід використовувати центральний процесор, а не графічний (GPU), що актуально для систем без підтримки CUDA. Усі ці параметри разом формують команду запуску FOMM з обраними налаштуваннями.

Далі модуль `subprocess.run` запускає зовнішній процес з усіма аргументами. Якщо команда завершується успішно, користувач бачить спливаюче повідомлення, що анімація завершена.

Після завершення виконання команди користувач отримує спливаюче повідомлення про успішну генерацію анімованого відео. Повідомлення інформує, що анімація завершена, і результат можна використовувати для подальшої обробки в Blender. Така візуальна індикація покращує взаємодію з користувачем, дозволяючи йому впевнено перейти до наступного етапу роботи.

Після завершення процесу оживлення статичного зображення користувача за допомогою моделі FOMM, згенероване відео обробляється системою трекінгу, яка базується на бібліотеці MediaPipe. На цьому етапі з відео у реальному часі зчитуються ключові точки обличчя, що дозволяє визначити параметри міміки: усмішку, моргання лівого і правого ока, а також ступінь відкриття рота під час розмови. Всі ці параметри нормуються у діапазоні від 0 до 1, після чого об’єднуються в пакет даних та передаються у Blender через локальне мережеве з’єднання (через сокет).

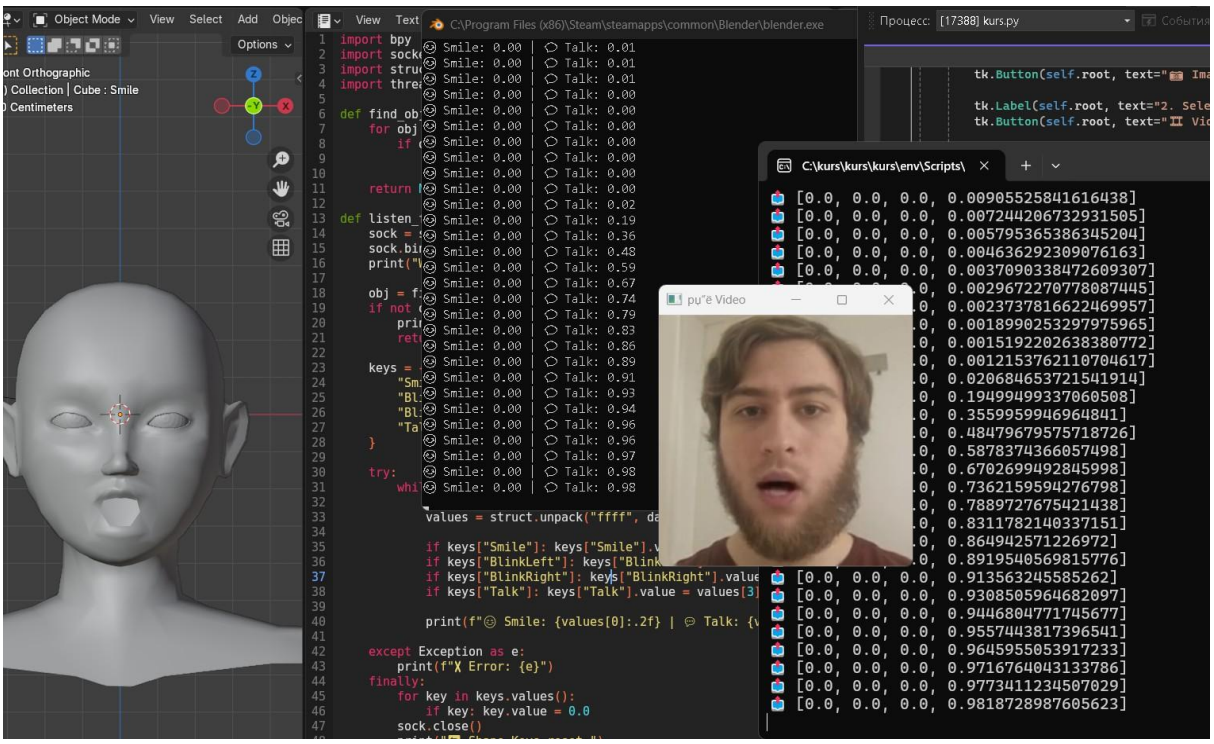


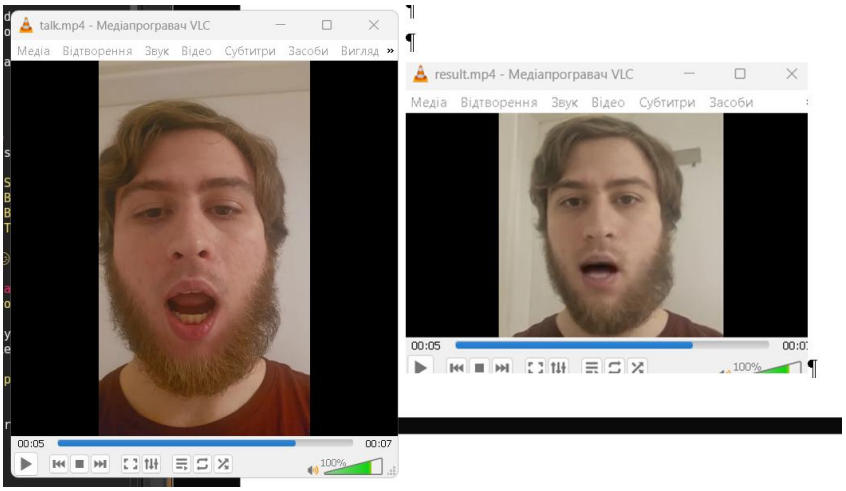
Рисунок 3.23 – Результат оживлення моделі яка відкриває рот

Програма, що обробляє міміку, пакує значення у вигляді чотирьох чисел з плаваючою точкою (типу float) та передає їх на порт 5005 комп’ютера, де очікує скрипт, запущений у середовищі Blender. Цей скрипт розпаковує отримані дані, а потім відповідно оновлює значення shape keys – Smile, BlinkLeft, BlinkRight та Talk – на моделі обличчя. Зміни в цих параметрах викликають анімацію відповідних м’язових деформацій на сітці моделі, що створює ефект «оживлення» та відтворення міміки, зчитаної з відео.

Крім цього, для оцінки ефективності підходу з використанням нейромережі First Order Motion Model (FOMM) було здійснено порівняння двох підходів до зчитування міміки, такі як безпосередньо з реального відео користувача або зі згенерованого анімованого відео на основі статичного зображення (через FOMM). З одного боку, розглядалося використання реального відео з обличчям користувача, на якому зафіксовані природні рухи, такі як усмішка, моргання чи вимова. З іншого боку, обробка відео, згенерованого нейронною мережею First Order Motion Model (FOMM), у якому міміка переноситься на статичне зображення користувача.

Порівняння цих підходів дозволяє оцінити, наскільки реально відзняте відео обличчя забезпечує кращу точність та деталізацію, а також чи може FOMM слугувати стабільною альтернативою в умовах, коли якість відеозйомки обмежена або запис обличчя неможливий. Крім точності, важливою складовою аналізу була стабільність трекінгу, тобто наскільки чітко і без втрат система визначає ключові точки на обличчі при різних джерелах вхідних даних. У результаті, експеримент мав на меті визначити, який метод є більш придатним для інтеграції з Blender, особливо в контексті подальшої передачі параметрів міміки на shape keys.

У ході експерименту були використані два джерела вхідних даних: перше – це відео, записане із природною мімікою користувача, друге – згенероване відео, створене за допомогою моделі FOMM, де ті самі рухи були перенесені на статичне зображення обличчя.



(а) (б)

Рисунок 3.24 – Порівняння якості двох відео:

(а) оригінал; (б) створене FOMM

На рис. 3.24 два відео: ліворуч оригінальне відео з реальним обличчям користувача (talk.mp4), праворуч анімоване відео, створене моделлю FOMM (result.mp4). Обидва кадри синхронізовані за часом – 00:05, проте візуально помітні кілька важливих відмінностей.

По-перше, на оригінальному відео відкриття рота є більш виразним, з чітко помітним положенням губ і природним натягом шкіри. У той же час, у згенерованому FOMM-відео рот виглядає менш відкритим, імітація губ не повністю відповідає реальній міміці, зокрема помітна деяка асиметрія або змазування деталей навколо рота. Це може бути пов’язано з тим, що модель FOMM працює в двовимірному просторі, і її реконструкція базується на апроксимації форми за допомогою локальних афінних перетворень, а не на тривимірному аналізі.

По-друге, у FOMM-відео також спостерігається втрата частини мімічних мікродеталей, таких як натяг шкіри щік чи чітке розмежування зубів. У деяких випадках у генерації може з’являтися легке зміщення рис обличчя або «розмиття» виразу, чого немає в оригіналі. Це особливо критично для задач, де важлива точність передачі артикуляції (наприклад, синхронізація з промовою).

Для обох випадків застосовувалася однакова формула визначення параметра «Talk», яка базується на вертикальній відстані між ключовими точками губ, виявленими за допомогою MediaPipe. У згенерованому відео (result.mp4), яке створюється FOMM на основі статичного зображення, рот відкривається лише майже так само, але є маленька погрішність, навіть у ті моменти, коли у вихідному відео користувач демонструє максимально виражене розкриття рота. Це пов’язано з обмеженнями генеративної моделі, вона прагне зберегти стабільність структури обличчя та уникати сильних деформацій. Як наслідок, амплітуда руху зменшується, що знижує реалістичність при моделюванні

Натомість при прямому використанні відео (talk.mp4), без FOMM, відстань між губами вимірюється точно у кожному кадрі, і тому параметр «Talk» має більш правильне значення. У Blender це призводить до чіткішої анімації відкриття рота: рух стає глибшим, динаміка змін природнішою, а синхронізація з вимовою точнішою. Різниця є, але не така велика

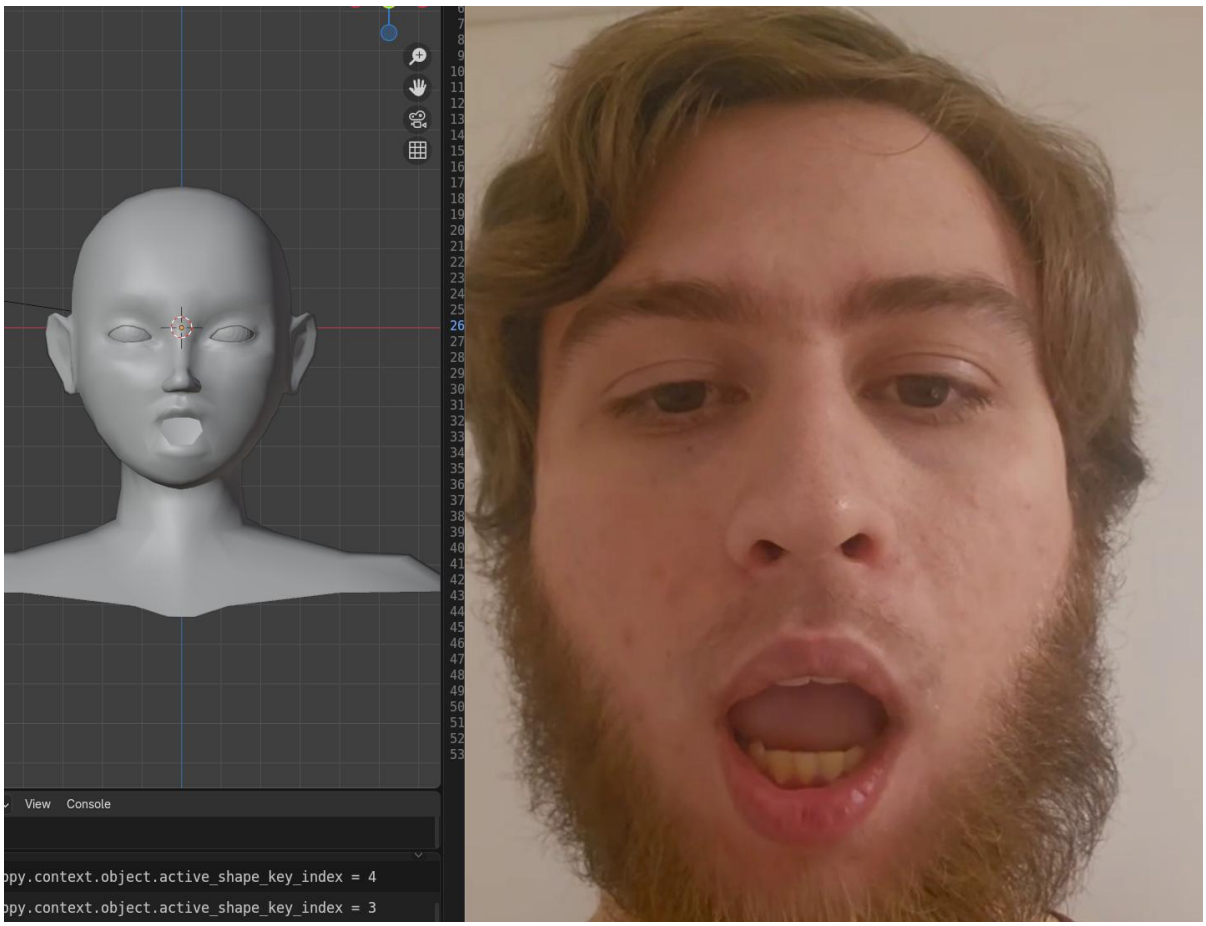


Рисунок 3.25 – Деформація моделі без участі FOMM

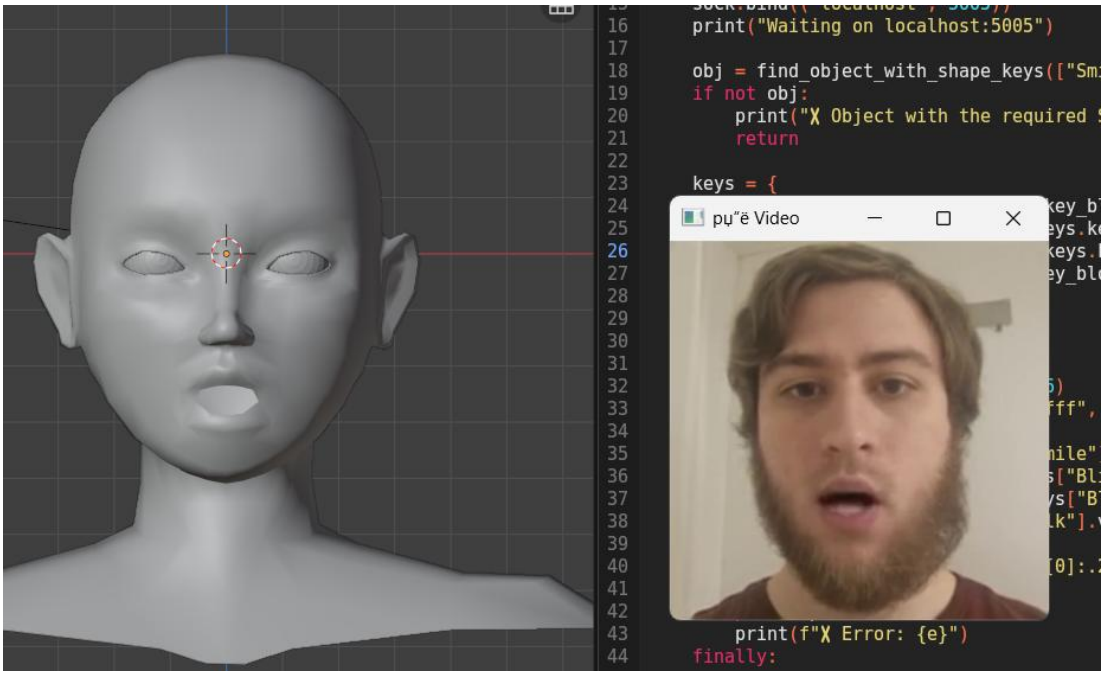


Рисунок 3.26 – Деформація моделі з FOMM

Другим важливим аспектом експерименту стало порівняння моргання. Цей рух є вкрай показовим при створенні живого образу, оскільки він відбувається часто. Для відстеження моргання в обох відео використовувався один і той самий алгоритм, який вимірює вертикальну відстань між верхнім і нижнім краєм повіки для обох очей. Ця відстань, при її значному зменшенні, інтерпретується як моргання, і відповідні shape keys у Blender активуються.

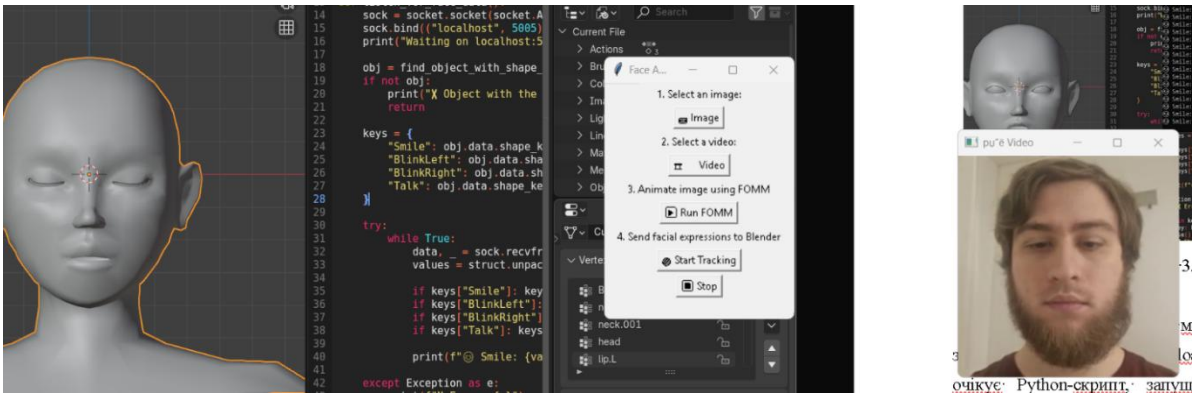


Рисунок 3.27 – Деформація очей з FOMM

У випадку з відео, згенерованим через FOMM, спостерігалося часткове або неповне закриття очей. Навіть при вираженому морганні у вихідному відео, у згенерованому результаті очі лишалися напіввідкритими або рух повік був занадто слабким. Це пояснюється тим, що генеративна модель прагне зберігати загальну консистентність зображення, тому обмежує сильні деформації області очей, які зазвичай вважаються слабкими для генерації. Модель FOMM намагається уникати різких змін у регіоні очей, адже будь-яка похибка там дуже помітна для людського спостерігача. З цієї причини навіть при наявності точних координат keypoints, відповідні афінні перетворення у генераторі зображень не забезпечують достатньої амплітуди деформацій у цій зоні. В результаті, моргання виглядає приглушеним або невиразним, що знижує ефект «оживлення» і може створити враження штучності або «завислого» погляду.

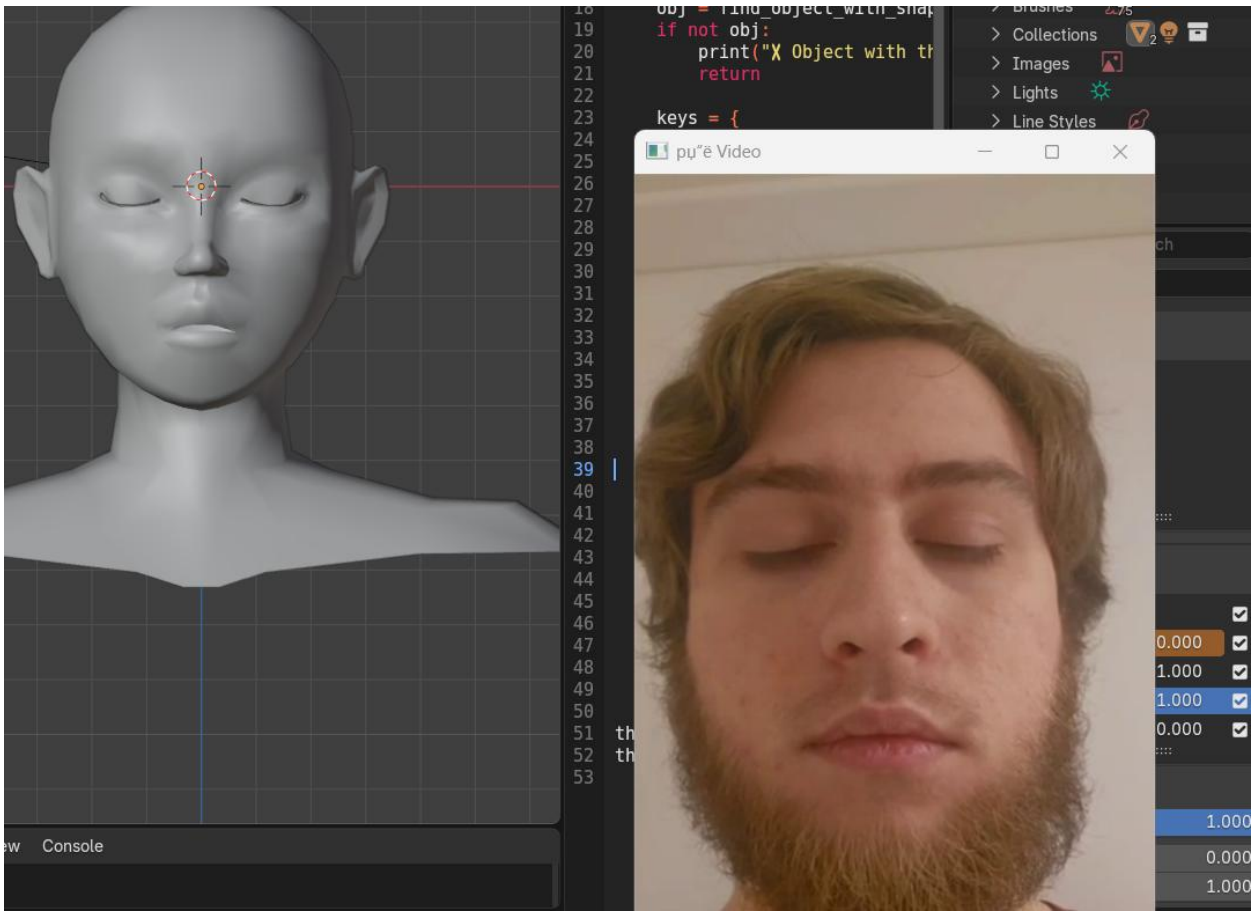


Рисунок 3.28 – Деформація очей без FOMM

Натомість при використанні оригінального відео, без попереднього оживлення через FOMM, трекінг очей спрацьовував чітко. У моменти моргання повіки повністю змикалися, і параметри BlinkLeft та BlinkRight у Blender отримували значення, близькі до 1.0. Візуально це забезпечувало більш правдоподібну анімацію: моргання виглядало природно, з чіткою фазою закриття та відкриття, без затримок чи зависань.

На завершальному етапі тестування системи було проведено аналіз впливу різного освітлення на якість виявлення міміки та результат її передачі у Blender. Було записано декілька варіантів відео в однаковій позі, але з різними умовами освітлення: рівномірне світло спереду, затемнене середовище та змішане освітлення зі сторонніми джерелами світла на фоні.

Рисунок 3.29 – Приклад різного освітлення

У межах експерименту було перевірено, як різні умови освітлення впливають на результати трекінгу міміки не лише при обробці реального відео, а й у випадку відео, згенерованого за допомогою FOMM. Це дозволило оцінити стійкість обох підходів до зміни візуальних характеристик сцени, зокрема рівня яскравості, напрямку світла та наявності тіней або контрастного фону.

У відео без використання FOMM (реальний запис) система MediaPipe продемонструвала достатню адаптивність: навіть при слабкому або нерівномірному освітленні вона змогла точно розпізнати більшість ключових точок обличчя, зокрема ті, що відповідають за моргання, рух рота і брів. Це забезпечило плавну і реалістичну передачу міміки до Blender.

Натомість у випадку відео, згенерованого за допомогою FOMM, ситуація виявилася майже без змін. Незважаючи на те, що зовнішній вигляд особи залишався не таким чітким, модель FOMM під впливом неідеального освітлення не почала демонструвати артефактів, таких як змазування, нестабільність обрисів очей, і часткові спотворення. Але мережа починала втрачати точність при визначенні координат делікатних зон. У результаті shape keys в Blender активувалися непослідовно: моргання ставало уривчастим або взагалі не відображалося, а усмішка втрачала симетричність.

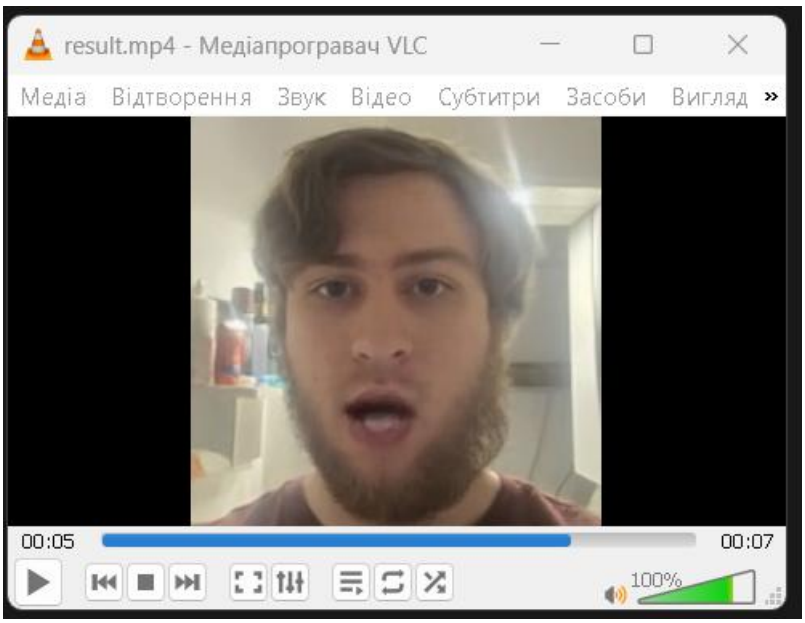


Рисунок 3.30– Приклад створеного відео з освітленням позаду

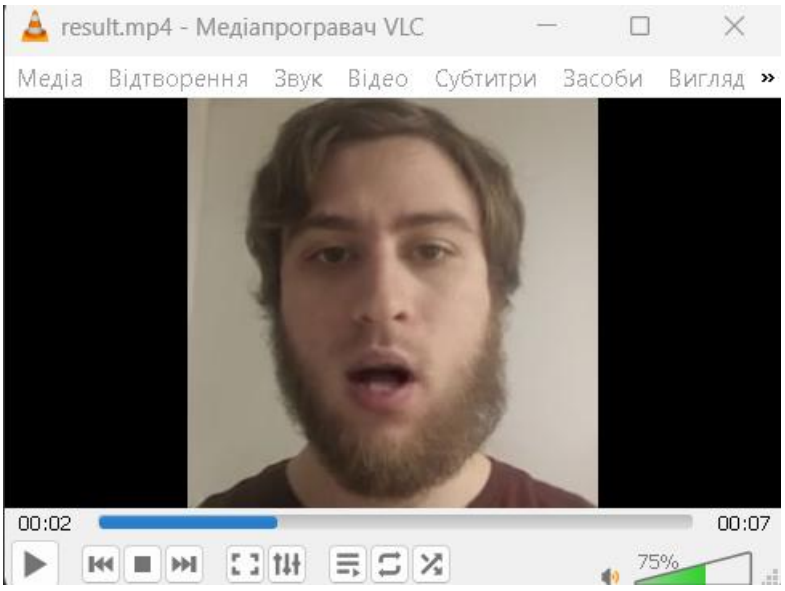


Рисунок 3.31– Приклад створеного відео з освітленням з боку

Отже, хоча FOMM ефективно передає загальні риси міміки, він менш придатний для точного зчитування швидких або делікатних мікрорухів.

Таким чином, на основі даних MediaPipe та резульату роботи FOMM, що надходять з інтерфейсу кориувача, Blender динамічно оновлює параметри shape keys. Це дозволяє досягнути плавної та правдоподібної імітації людської міміки на тривимірній моделі обличчя.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та реалізовано систему передачі міміки на модель обличчя в Blender на основі попереднього оживлення статичного зображення за допомогою нейромережі FOMM.

Проведено поетапний аналіз і реалізацію основних складових системи: від підготовки вхідних відеоданих і генерації анімації за допомогою нейронної моделі до обробки ключових точок міміки через MediaPipe і передавання параметрів міміки в Blender через shape keys.

Система забезпечує візуально правдоподібну анімацію виразів обличчя на моделі, а застосування згладжування (з коефіцієнтом інерції α) дозволяє уникати ривків і шуму при передачі міміки.

Запропонований підхід з використанням FOMM дозволив досягти стабільного результату навіть у випадку, коли оригінальне відео навіть має неідеальні умови зйомки.

Отримані результати можуть бути використані для розробки інтерактивних персонажів, анімації у відео-матеріалах або створення персоналізованих аватарів. Далі, можливе удосконалення точності відповідності shape keys і додавання додаткових емоцій, таких як страх або здивування.

Результати роботи апробовано у вигляді тез доповідей під час Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [23]».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сайт Siarohin A., et al. First Order Motion Model for Image Animation. arXiv:1906.08172, 2019. URL: https://arxiv.org/abs/1906.08172 (дата звернення 24.04.2025).

2. Сайт Blender Documentation: Shape Keys. URL: https://docs.blender.org/manual/en/latest/animation/shape_keys/index.html (дата звернення 24.04.2025).

3. Сайт Blender Python API: ShapeKey. URL: https://docs.blender.org/api/current/bpy.types.ShapeKey.html (дата звернення 25.04.2025).

4. Сайт UDP сокети в Python. URL: https://wiki.python.org/moin/UdpCommunication (дата звернення 25.04.2025)

5. Сайт Python socket programming documentation. URL: https://docs.python.org/3/library/socket.html (дата звернення 25.04.2025).

6. Сайт Siarohin A., et al. Supplementary Materials: Handling occlusions in motion transfer. URL: https://arxiv.org/html/2412.06174v1 (дата звернення 25.04.2025).

7. Сайт Siarohin et al., 2019. First Order Motion Model for Image Animation. URL: https://arxiv.org/abs/2003.00196 (дата звернення 28.04.2025).

8. Сайт Blender Python API: ShapeKey. URL: https://docs.blender.org/api/current/bpy.types.ShapeKey.html (дата звернення 02.05.2025)

9. Сайт Python Docs: Tkinter File Dialog. URL: https://docs.python.org/3/library/dialog.html (дата звернення 06.05.2025).

10. Сайт Python Tkinter Docs – File Dialogs. URL: https://docs.python.org/3/library/dialog.html (дата звернення 06.05.2025)

11. Сайт Siarohin A., et al. First Order Motion Model for Image Animation. arXiv:1906.08172. URL: https://arxiv.org/abs/1906.08172 (дата

звернення 06.05.2025).

12. Сайт MediaPipe Documentation. URL: https://github.com/google-ai-edge/mediapipe (дата звернення 06.05.2025).

13. Сайт OpenCV Documentation. VideoCapture Class Reference. URL:https://docs.opencv.org/4.x/d8/dfe/classcv_1_1VideoCapture.html (дата звернення 07.05.2025)

14. Сайт Ffmpeg. URL: https://ffmpeg.org/ (дата звернення 07.05.2025).

15. Сайт Blender Python API. URL: https://docs.blender.org/api/current/https://ffmpeg.org/ (дата звернення 07.05.2025).

16. Сайт Pavllo D., Grangier D., Auli M. QuaterNet: A Quaternion-based Recurrent Model for Human Motion. arXiv:1805.06485. URL: https://arxiv.org/abs/1805.06485 (дата звернення 07.05.2025).

17. Сайт Python Software Foundation. Python Language Reference. URL: https://www.python.org (дата звернення 07.05.2025).

18. Сайт Learning OpenCV. URL: https://www.bogotobogo.com/cplusplus/files/OReilly%20Learning%20OpenCV.pdf (дата звернення 07.05.2025).

19. Сайт Tkinter GUI Programming – офіційне керівництво Python. URL:https://docs.python.org/3/library/tkinter.html (дата звернення 11.05.2025).

20. Сайт Офіційний репозиторій FOMM. URL:https://github.com/AliaksandrSiarohin/first-order-model (дата звернення 11.05.2025).

21. Сайт Документація Python – модуль subprocess. URL:https://docs.python.org/3/library/subprocess.html (дата звернення 12.05.2025).

22. Сайт Документація Python – модуль threading . URL:https://docs.python.org/3/library/threading.html (дата звернення 12.05.2025).

23. Сайт Комп'ютерний зір, системний аналіз та математичне моделювання. URL: https://drive.google.com/file/d/1y0KHfmR0qEQ8KR2JE1EBLD9-UO_muxJl/view (дата звернення 12.05.2025).

24. Gorokhovatskyi, V., Vlasenko, N. (2021). Редукція опису зображення у складі множини дескрипторів на основі метричного критерію інформативності. *Advanced Information Systems,* 5(4), 10–16.

25. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., Vlasenko, N. (2023) Explanation of CNN Image Classifiers with Hiding Parts. In: J. Benois-Pineau, R. Bourqui, D. Petkovic (eds), *Explainable Deep Learning Artificial Intelligence*, pp. 125-146, Academic Press, 346 p.

26. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, 33 (1), 113-125.

27. Gorokhovatskyi, V., Gadetska, S., & Stiahlyk, N. (2023). Accelerating Image Classification based on a Model for Estimating Descriptor-to-Class Distance. *International Journal of Computing*, 22(4), 485-492.

28. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, 11, 126938-126949.

29. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, 73376-73385.

30. Gorokhovatskyi V., Gadetska S., Stiahlyk N. (2020) Image structural classification technologies based on statistical analysis of descriptions in the form of bit descriptor set. In CEUR Workshop Proceedings: *Computer Modeling and Intelligent Systems* (CMIS-2020), 2608, 1027-1039.

31. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M.

(2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, Computers, *Materials & Continua*, vol. 80, no. 2, 3085-3106.

32. Gorokhovatskyi, V., Stiahlyk, N., Mazur, Y., Vechirska, A. (2024) Способи метричної грануляції для опису зображень у задачі класифікації. Системи управління, навігації та зв'язку. *Збірник наукових праць*, 3(77), 106-112.

33. Гороховатський В.О., Гадецька С.В. (2020) Статистичне оброблення та аналіз даних у структурних методах класифікації зображень (монографія), Харків, ФОП Панов А.Н., 128 с., ISBN 978-617-7859-69-6, DOI: 10.30837/978-617-7859-69-6

34. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. *Advanced Information Systems*, 9(1), 5–12. https://doi.org/10.20998/2522-9052.2025.1.01

35. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, *IEEE Access*, vol. 13, pp. 43631-43641, doi: 10.1109/ACCESS.2025.3548910.

36. Gorokhovatskyi V., Tvoroshenko I. (2024) An effective method for transforming an image description into a compact vector for classification. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / V. Snytyuk (Editor). – *Kyiv: Publishing House «Caravela»,* 25-28. https://openarchive.nure.ua/handle/document/29476

37. Gorokhovatskyi, V., Gadetska, S., Stiahlyk, N. (2024) Classification of images based on distance assessment. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / V. Snytyuk (Editor). – Kyiv: *Publishing House «Caravela»*, 22-24. https://openarchive.nure.ua/handle/document/29478

38. Pupchenko, D., Gorokhovatskyi, V. (2024) Accelerated filtration of

ultrasound images. Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / V. Snytyuk (Editor). – Kyiv: Publishing *House «Caravela»*, 69-72. https://openarchive.nure.ua/handle/document/29492

39. Гороховатський В.О., Гадецька С.В., Стяглик Н.І. (2019) Вивчення статистичних властивостей моделі блочного подання для множини дескрипторів ключових точок зображень. *Радіоелектроніка, інформатика, управління*, №2, 100–107.

40. V. Gorokhovatsky, Y. Putyatin and V. Stolyarov (2017) Research of Effectivenss of Structural Image Classification Methods using Cluster Data Model, *Radio Electronics Computer Science Control*, vol. 3, no. 42, 78-85.

41. Gorokhovatsky, V.A., Putyatin, Y.P. (2008) Structural recognition of images on the basis of voting models of attributes of typical points, Data recording, *storage and processing*, 10(4), 75-85.

42. Gorokhovatskyi V.A., Zamula A.A. (2016) Employment of Intelligent Technologies in Multiparametric Control Systems. *Telecommunications and Radio Engineering*. Vol. 75, No 19, 1775–1785.

43. Gadetska, S.V., Gorokhovatskyi, V. O., Stiahlyk, N. I., Vlasenko, N.V. Statistical data analysis tools in image classification methods based on the description as a set of binary descriptors of key points. Radio Electronics, *Computer Science*, Control, 2021, №4, 58-68.