МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «РЕАЛІЗАЦІЯ РЕКОНСТРУКЦІЇ ТРИВИМІРНОЇ МОДЕЛІ ОБЛИЧЧЯ НА ОСНОВІ ВІДЕОЗАПИСУ»

Виконав: студент 3 курсу, групи 6.1223-с2
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
С.О. Петрик
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, к.т.н Добровольський Г.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, к.ф.-м.н. Мильцев О.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, к.пед.н., доцент

_____ Пшенична О.С.
(підпис)

" ____ " _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Петрику Сергію Олександровичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Реалізація реконструкції тривимірної моделі обличчя на основі відеозапису

керівник роботи Добровольський Геннадій Анатолійович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 29.05.2026

3. Вихідні дані до роботи Тема роботи та постановка задачі; наукові джерела з комп'ютерного зору, глибокого навчання та 3D-реконструкції; документація OpenCV, PyTorch, InsightFace, MICA, FLAME; відеодані MIT OpenCourseWare для проведення експериментального дослідження.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Постановка задачі реконструкції тривимірної моделі обличчя на основі відеозапису та аналіз сучасних методів 3D-реконструкції обличчя.
   2. Дослідження моделей FLAME, MICA та методів детекції, та фільтрації кадрів.
   3. Розробка програмного пайплайну реконструкції тривимірної моделі обличчя.
   4. Оцінювання якості реконструкції та аналіз продуктивності системи.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 08.12.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.12.2025	
2.	Обробка методичних та теоретичних		
	джерел.	29.12.205	
3.	Розробка першого та другого розділу.	12.01.2026	
4.	Розробка третього розділу.	09.03.2026	
5.	Розробка четвертого розділу.	04.05.2026	
6.	Оформлення та нормоконтроль	20.05.2026	
7.	Захист кваліфікаційної роботи.	20.06.2026	

Студент _____ С.О. Петрик
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.А. Добровольський
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Реалізація реконструкції тривимірної моделі обличчя на основі відеозапису»: 101 с., 26 рис., 4 табл., 22 джерел, 3 додатки.

БІОМЕТРИЧНА АВТЕНТИФІКАЦІЯ, ГЛИБОКЕ НАВЧАННЯ, ДЕТЕКЦІЯ ОБЛИЧЧЯ, КОМП'ЮТЕРНИЙ ЗІР, МОРФОВАНА МОДЕЛЬ ОБЛИЧЧЯ, ПАРАМЕТРИЧНА МОДЕЛЬ FLAME, РЕКОНСТРУКЦІЯ ТРИВИМІРНОЇ МОДЕЛІ, ТРИВИМІРНА ГЕОМЕТРІЯ ОБЛИЧЧЯ, ФІЛЬТРАЦІЯ КАДРІВ, ЦИФРОВИЙ АВАТАР.

Об'єкт дослідження – процес реконструкції тривимірної моделі обличчя людини на основі відеоданих в умовах неконтрольованого середовища.

Предмет дослідження – методи та програмні засоби детекції обличчя, фільтрації кадрів і реконструкції тривимірної геометрії обличчя з використанням нейромережевих моделей.

Мета роботи: розробка програмного пайплайну для автоматичної реконструкції тривимірної моделі обличчя на основі відеозапису та оцінка придатності отриманих моделей для задач біометричної автентифікації.

Методи дослідження – методи комп'ютерного зору, згорткові нейронні мережі, параметричні морфовані моделі, методи статистичного аналізу, методи оцінювання подібності тривимірних моделей, ROC-аналіз, методи обробки та фільтрації зображень.

У роботі розроблено систему реконструкції тривимірної моделі обличчя на основі відеозапису з використанням нейронної мережі MICA та параметричної моделі FLAME. Реалізований пайплайн включає комбіновану фільтрацію кадрів, детекцію обличчя, відсів кадрів з нахилом голови понад допустимий поріг, пакетну нейромережеву реконструкцію та медіанне злиття параметрів форми. Проведено кількісне оцінювання із застосуванням ROC-аналізу

SUMMARY

Bachelor's Qualifying Theses «Implementation of 3D Face Reconstruction from Video Records»: 101 pages, 26 figures, 4 tables, 22 references, 3 supplements.

BIOMETRIC AUTHENTICATION, DEEP LEARNING, FACE DETECTION, COMPUTER VISION, FACE MORPHABLE MODEL, FLAME PARAMETRIC MODEL, THREE-DIMENSIONAL MODEL RECONSTRUCTION, THREE-DIMENSIONAL FACIAL GEOMETRY, FRAME FILTERING, DIGITAL AVATAR.

Object of the study – the process of reconstructing a three-dimensional human face model from video data in uncontrolled environmental conditions.

Subject of the study – methods and software tools for face detection, frame filtering, and three-dimensional facial geometry reconstruction using neural network models.

Aim of the study: to develop a software pipeline for automatic three-dimensional face model reconstruction from video recordings and to evaluate the suitability of the resulting models for biometric authentication tasks.

Methods of research – computer vision methods, convolutional neural networks, parametric morphable models, statistical analysis methods, three-dimensional model similarity assessment methods, ROC analysis, image processing and filtering methods.

The thesis presents a system for three-dimensional face model reconstruction from video recordings using the MICA neural network and the FLAME parametric model. The implemented pipeline incorporates combined frame filtering, face detection, rejection of frames with head rotation exceeding a defined threshold, batch neural network reconstruction, and median merging of shape parameters. Quantitative evaluation was performed using ROC analysis.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 8
1 Аналіз предметної області та постановка задачі 11
1.1 Актуальність реконструкції тривимірних моделей обличчя 11
1.2 Огляд існуючих методів 3D-реконструкції обличчя 13
1.2.1 Методи на основі стереоскопії та структурованого світла 14
1.2.2 Методи на основі морфованих моделей (3DMM) 14
1.2.3 Методи на основі глибокого навчання 15
1.3 Порівняльний аналіз існуючих рішень та обґрунтування вибору підходу 16
1.4 Постановка задачі та вимоги до системи 18
2 Теоретичні основи 20
2.1 Морфована модель обличчя FLAME 20
2.1.1 Математичне описання моделі 21
2.1.2 Параметри форми, виразів і пози 23
2.2 Нейронна мережа MICA та детектор RetinaFace 25
2.2.1 Архітектура та ключові точки RetinaFace 26
2.2.2 Архітектура і принцип роботи MICA 28
2.3 Метрики якості кадрів: дисперсія Лапласіана та SSIM 29
3 Проєктування та реалізація системи 32
3.1 Загальна архітектура пайплайну та вибір програмних засобів 32
3.2 Отримання та фільтрація кадрів 34
3.2.1 Страйд-семплінг 34
3.2.2 Фільтрація за різкістю та SSIM 35
3.3 Детекція, оцінка пози голови та вирівнювання обличчя 36

3.3.1 Детекція та відсів кадрів за кутами yaw і pitch 37
3.3.2 Афінне вирівнювання за ключовими точками 38
3.4 Реконструкція 3D-форми за допомогою MICA 39
3.4.1 Пакетна обробка з контролем відеопам’яті 39
3.4.2 Медіанне злиття параметрів та декодування меша 41
3.5 Постобробка та експорт 3D-моделі 43
3.6 Логування та обробка помилок 44
3.7 Запуск пайплайну та реалізація CLI 46
4 Тестування та аналіз результатів 49
4.1 Опис тестових даних і умов експерименту 49
4.2 Аналіз якості фільтрації та підготовки кадрів 51
4.3 Кількісна оцінка подібності 3D-моделей облич 54
4.3.1 Порівняння метрик подібності та відстані 55
4.3.2 ROC-аналіз та оцінка класифікаційної здатності 58
4.4 Візуальна оцінка реконструкції 62
4.5 Оцінка продуктивності системи 66
Висновки 75
Перелік посилань 78
Додаток А Графічне представлення пайплайну 81
Додаток Б Код пайплайну мовою Python 82
Додаток В Приклади роботи пайплайну 101

ВСТУП

Останніми роками розвиток технологій комп’ютерного зору та машинного навчання набув інтенсивного характеру, що зумовило суттєве розширення сфер їх практичного застосування. Одним із перспективних і активно досліджуваних напрямів є реконструкція тривимірних моделей обличчя людини на основі двовимірних зображень і відеопослідовностей. Ця технологія знаходить застосування в системах біометричної автентифікації, доповненої та віртуальної реальності, комп’ютерних іграх, кінематографі, телекомунікаціях, а також у медицині та соціальній робототехніці.

Особливої актуальності реконструкція тривимірної моделі обличчя набуває в контексті біометричної автентифікації користувачів електронних пристроїв та інформаційних систем. Традиційні методи ідентифікації, засновані на паролях або двовимірному розпізнаванні облич, мають низку недоліків, зокрема низьку стійкість до компрометації та атак типу «спуфінг» (використання фотографій або відео замість реального обличчя). Перехід до аналізу тривимірної геометрії дає змогу суттєво підвищити рівень безпеки, підтверджуючи реальний об’єм об’єкта. Однак більшість наявних комерційних систем біометрії обличчя потребують спеціалізованих сенсорів, таких як інфрачервоні камери або структуроване світло, що збільшує вартість і обмежує їх масове впровадження. У зв’язку з цим перспективним є підхід, що полягає в реконструкції тривимірної моделі обличчя на основі звичайного відеозапису та дає змогу підвищити надійність автентифікації без використання дорогого обладнання.

Додатковим важливим напрямом застосування тривимірних моделей обличчя є створення персоналізованих 3D-аватарів. Із зростанням популярності віртуальних просторів, метавсесвітів, онлайн-ігор і систем дистанційного спілкування зростає потреба в реалістичному й водночас автоматично згенерованому цифровому представленні користувача.

Використання відеозапису в поєднанні з нейромережевими методами дає змогу отримувати не лише статичну форму, а й динаміку міміки в режимі реального часу, що робить такі технології затребуваними в сучасних інтерактивних застосунках.

Історично задачі реконструкції тривимірних об’єктів розв’язувалися з використанням методів фотограмметрії та спеціалізованих 3D-сканерів. Попри високу точність, такі підходи вимагали контрольованих умов зйомки та складного апаратного забезпечення. Із розвитком обчислювальних потужностей і глибоких нейронних мереж з’явилися методи, що дозволяють відновлювати форму та геометрію обличчя за одним зображенням або відеопослідовністю. Вагомий внесок у розвиток цього напряму зробили параметричні моделі обличчя, такі як 3D Morphable Model (3DMM), а також сучасні нейромережеві архітектури, навчені на великих наборах даних [1–3]. В останні роки особлива увага приділяється методам, що використовують відеозаписи, оскільки вони дають змогу враховувати динаміку міміки та отримувати більш стійкі й точні результати реконструкції [4, 5].

Попри значний прогрес у цій галузі, задача реконструкції тривимірної моделі обличчя на основі відеозапису залишається актуальною. До основних нерозв’язаних проблем належать чутливість алгоритмів до умов освітлення, поворотів голови, часткових оклюзій, а також необхідність балансу між точністю реконструкції та обчислювальною складністю. Крім того, багато сучасних рішень є закритими й погано адаптованими для інтеграції в користувацькі застосунки, що зумовлює потребу в розробленні та реалізації власних програмних рішень, орієнтованих на практичне використання.

Метою цієї дипломної роботи є реалізація системи реконструкції тривимірної моделі обличчя на основі відеозапису з використанням нейромережевих методів для задач автентифікації та візуалізації.

Для досягнення поставленої мети в роботі передбачається розв’язання таких завдань:

- виконати порівняльний аналіз наявних методів і підходів до реконструкції 3D-моделей обличчя за відеозаписами;
- дослідити сучасні нейромережеві архітектури та параметричні моделі, що застосовуються для відновлення геометрії обличчя;
- розробити програмну реалізацію реконструкції тривимірної моделі обличчя;
- провести експериментальну оцінку якості отриманих моделей;
- дослідити можливості інтеграції розробленого рішення в системи біометричної автентифікації та генерації цифрових аватарів.

Об’єктом дослідження є методи та технології комп’ютерного зору, що застосовуються для аналізу зображень і відеопослідовностей обличчя людини.

Предметом дослідження є нейромережеві алгоритми та параметричні моделі реконструкції тривимірної геометрії обличчя на основі відеозапису.

Під час виконання роботи використовуються методи аналізу та синтезу, глибокого навчання, нейромережевого моделювання, а також експериментальні методи оцінювання якості реконструкції.

Новизна роботи полягає в практичній реалізації та адаптації сучасних нейромережевих методів реконструкції тривимірної моделі обличчя для роботи з відеозаписами, отриманими в неконтрольованих умовах, з орієнтацією на прикладне використання в користувацьких системах.

Практична значимість роботи полягає в можливості використання розробленого програмного рішення в системах біометричної автентифікації користувачів, а також під час створення персоналізованих 3D-аватарів для віртуальних і доповнених середовищ.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність реконструкції тривимірних моделей обличчя

У сучасному світі стрімкий розвиток інформаційних технологій, комп’ютерного зору та систем штучного інтелекту призводить до появи нових методів обробки та аналізу зображень людини. Одним із важливих напрямів досліджень у цій галузі є реконструкція тривимірних моделей обличчя. Дана задача полягає у відновленні просторової геометрії обличчя людини на основі двовимірних зображень або відеопослідовностей. На відміну від традиційних двовимірних методів аналізу обличчя, тривимірні моделі дозволяють враховувати геометричну структуру поверхні обличчя, що забезпечує більш точне та стабільне представлення об’єкта дослідження [1].

Актуальність дослідження методів тривимірної реконструкції обличчя обумовлена широким спектром практичних застосувань цієї технології. Однією з основних сфер використання є біометрична ідентифікація та автентифікація користувачів. Сучасні системи інформаційної безпеки дедалі частіше використовують біометричні параметри людини як засіб підтвердження особи. Обличчя є одним із найбільш зручних біометричних параметрів, оскільки його можна аналізувати безконтактним способом за допомогою звичайних камер. Однак традиційні системи розпізнавання облич, що базуються на двовимірних зображеннях, мають певні обмеження. Їхня точність може суттєво знижуватися через зміну освітлення, положення голови або міміки людини. Використання тривимірних моделей дозволяє частково усунути ці недоліки, оскільки геометричні характеристики обличчя залишаються відносно стабільними незалежно від умов зйомки [6].

Значну роль технології тривимірної реконструкції обличчя відіграють і в системах автентифікації користувачів електронних пристроїв. У сучасних смартфонах, планшетах та комп’ютерах широко застосовуються системи

біометричного доступу. Технології розпізнавання облич дозволяють розблоковувати пристрої, підтверджувати фінансові операції та забезпечувати доступ до конфіденційної інформації. Використання тривимірних моделей обличчя підвищує рівень безпеки таких систем, оскільки ускладнює можливість їх обходу за допомогою фотографій або відеозаписів [7].

Ще одним важливим напрямом використання технологій реконструкції обличчя є індустрія комп'ютерної графіки та цифрових медіа. У сучасних відеоіграх, кіноіндустрії та системах віртуальної реальності широко застосовуються тривимірні моделі персонажів. Створення таких моделей традиційно потребує значних трудових та часових ресурсів, оскільки виконується художниками вручну або за допомогою спеціалізованих 3D-сканерів. Використання автоматичних методів реконструкції обличчя на основі зображень або відео дозволяє значно спростити цей процес та створювати реалістичні цифрові аватари користувачів.

Крім того, технології тривимірної реконструкції обличчя активно застосовуються у медичній сфері. Зокрема, вони використовуються для планування пластичних та реконструктивних операцій, ортодонтичного лікування та дослідження анатомічних особливостей обличчя пацієнтів. Тривимірні моделі дозволяють лікарям більш точно аналізувати форму обличчя та прогнозувати результати хірургічних втручань.

Важливим напрямом застосування є також криміналістика та судово-медична експертиза. У цих галузях тривимірні моделі можуть використовуватися для реконструкції зовнішності людини на основі фотографій або відеозаписів, отриманих із камер спостереження. Такі технології допомагають підвищити ефективність розслідування злочинів та ідентифікації невідомих осіб.

Останніми роками розвиток глибокого навчання та нейронних мереж значно розширив можливості реконструкції обличчя. Сучасні алгоритми дозволяють відновлювати тривимірну форму обличчя навіть за одним зображенням або коротким відеозаписом. Використання таких методів

забезпечує високу швидкість обробки даних і відкриває можливість створення систем реконструкції, що працюють у режимі реального часу [8].

Таким чином, реконструкція тривимірних моделей обличчя є важливим і актуальним напрямом досліджень у галузі комп'ютерного зору. Широкий спектр її застосувань − від біометричних систем безпеки до комп'ютерної графіки та медицини − обумовлює необхідність подальшого розвитку методів і алгоритмів тривимірної реконструкції. Це робить дослідження у цій галузі важливими як з наукової, так і з практичної точки зору.

1.2 Огляд існуючих методів 3D-реконструкції обличчя

Задача реконструкції тривимірної моделі обличчя полягає у відновленні геометричної структури поверхні обличчя на основі вхідних візуальних даних, таких як фотографії або відеозаписи. Для розв'язання цієї задачі було розроблено різні підходи, які відрізняються принципами роботи, вимогами до вхідних даних та точністю результатів. Загалом методи реконструкції тривимірних моделей обличчя можна поділити на три основні групи: геометричні методи, методи на основі статистичних морфованих моделей та методи глибокого навчання [6].

Розвиток цих підходів відбувався разом із розвитком комп'ютерного зору та обчислювальних технологій. Спочатку використовувалися геометричні методи з кількома камерами, пізніше з'явилися статистичні моделі, що дозволяють реконструювати обличчя навіть за одним зображенням. У сучасних системах найбільшого поширення набули методи глибокого навчання, які забезпечують високу точність та швидкість обробки даних [1].

У наступних підрозділах буде розглянуто основні підходи до реконструкції тривимірних моделей обличчя: геометричні методи, методи на основі морфованих моделей та сучасні алгоритми глибокого навчання.

1.2.1 Методи на основі стереоскопії та структурованого світла

Методи на основі стереоскопії та структурованого світла належать до класичних підходів реконструкції тривимірної геометрії об’єктів. Основна ідея цих методів полягає у використанні декількох зображень одного й того самого об’єкта, отриманих із різних ракурсів. На основі відповідностей між точками на цих зображеннях обчислюються їхні просторові координати у тривимірному просторі [9].

Одним із поширених підходів є метод Structure-from-Motion, який дозволяє відновлювати структуру сцени та параметри камер на основі послідовності зображень або відеозапису. У цьому випадку спочатку визначаються характерні точки на різних кадрах, після чого встановлюються відповідності між ними. На основі цих даних обчислюється тривимірне положення точок та формується модель об’єкта.

Іншим варіантом є використання структурованого світла. У цьому методі на об’єкт проектується спеціальний світловий шаблон, деформація якого на поверхні об’єкта дозволяє визначити його геометрію. Такі системи використовуються у професійних 3D-сканерах і забезпечують високу точність реконструкції.

Основним недоліком геометричних методів є необхідність використання спеціального обладнання або великої кількості зображень, що обмежує можливість їх використання у простих системах, наприклад на звичайних персональних комп’ютерах або мобільних пристроях.

1.2.2 Методи на основі морфованих моделей (3DMM)

Методи на основі морфованих моделей обличчя (3D Morphable Model, 3DMM) є одним із найбільш поширених підходів до реконструкції обличчя. У цьому підході використовується статистична модель форми та текстури обличчя, побудована на основі великої кількості тривимірних сканів [1].

Основна ідея полягає у представленні форми обличчя як лінійної комбінації базових компонентів. Кожне нове обличчя може бути описане набором параметрів моделі, які визначають його індивідуальні особливості. Під час реконструкції ці параметри підбираються таким чином, щоб зображення моделі максимально відповідало вхідному зображенню обличчя.

Перевагою цього підходу є можливість реконструкції тривимірної моделі навіть за одним зображенням. Крім того, використання статистичної моделі дозволяє отримувати стабільні результати навіть за наявності часткових перекриттів або змін освітлення.

Разом з тим точність реконструкції залежить від якості навчального набору даних, на основі якого була побудована модель.

1.2.3 Методи на основі глибокого навчання

Останніми роками значного розвитку набули методи реконструкції обличчя на основі глибокого навчання. Ці методи використовують нейронні мережі для автоматичного визначення параметрів тривимірної моделі обличчя на основі двовимірних зображень або відеоданих [10].

Більшість сучасних алгоритмів використовують згорткові нейронні мережі (CNN), які здатні ефективно аналізувати зображення та виділяти характерні ознаки. Після навчання на великих наборах даних такі моделі можуть відновлювати форму обличчя навіть за одним зображенням.

До сучасних реалізацій методів реконструкції обличчя на основі глибокого навчання належать алгоритми DECA, MICA та Deep3DFaceRecon. Ці підходи використовують нейронні мережі для визначення параметрів статистичних моделей обличчя на основі зображень або відеоданих. Поєднання глибоких нейронних мереж із параметричними моделями обличчя дозволяє враховувати міміку, освітлення та положення голови і отримувати деталізовані тривимірні моделі.

Методи глибокого навчання є найбільш перспективним напрямом розвитку технологій реконструкції обличчя, оскільки вони забезпечують високу точність, швидкість роботи та можливість використання звичайних відеокамер без спеціалізованого обладнання.

1.3 Порівняльний аналіз існуючих рішень та обґрунтування вибору підходу

Існуючі методи реконструкції тривимірних моделей обличчя відрізняються за принципами роботи, вимогами до вхідних даних, точністю результатів та обчислювальною складністю. Тому при розробці системи реконструкції важливо обрати підхід, який забезпечує оптимальне співвідношення між якістю отриманої моделі, швидкістю роботи та вимогами до апаратних ресурсів. У цьому підрозділі проведено порівняльний аналіз основних підходів до реконструкції обличчя, а саме геометричних методів, методів на основі морфованих моделей та методів глибокого навчання.

Геометричні методи реконструкції, такі як стереоскопічна зйомка, фотограмметрія або метод Structure-from-Motion, базуються на використанні декількох зображень об’єкта, отриманих з різних ракурсів. На основі відповідностей між характерними точками на різних зображеннях обчислюються їх координати у тривимірному просторі. Перевагою такого підходу є висока точність реконструкції геометрії поверхні об’єкта. Проте для отримання якісного результату необхідна значна кількість зображень або використання декількох синхронізованих камер. Крім того, такі методи часто потребують контрольованих умов зйомки та значних обчислювальних ресурсів, що обмежує можливість їх використання у звичайних програмних системах [9].

Методи на основі морфованих моделей обличчя (3D Morphable Model, 3DMM) дозволяють подолати частину обмежень геометричних підходів. У

цьому випадку використовується статистична модель обличчя, побудована на основі великого набору тривимірних сканів. Реконструкція обличчя зводиться до підбору параметрів цієї моделі таким чином, щоб її проекція максимально відповідала вхідному зображенню. Перевагою цього підходу є можливість реконструкції тривимірної моделі навіть за одним зображенням. Також такі методи є відносно стабільними до змін освітлення та часткових перекриттів обличчя. Водночас точність реконструкції значною мірою залежить від якості статистичної моделі та навчального набору даних [1].

Сучасні методи реконструкції обличчя все частіше використовують алгоритми глибокого навчання. У таких системах застосовуються згорткові нейронні мережі, які навчаються визначати параметри тривимірної моделі обличчя на основі двовимірних зображень або відеоданих. Після навчання такі моделі можуть швидко відновлювати геометрію обличчя навіть за одним кадром. Крім того, використання відеопослідовностей дозволяє підвищити точність реконструкції за рахунок об’єднання інформації з декількох кадрів [10].

Сучасні алгоритми реконструкції обличчя, зокрема DECA, MICA та Deep3DFaceRecon, демонструють високу точність відновлення геометрії обличчя та здатні працювати з даними, отриманими зі звичайних камер. Поєднання статистичних моделей обличчя з можливостями глибоких нейронних мереж дозволяє підвищити стійкість реконструкції до змін освітлення, міміки та положення голови.

З урахуванням проведеного аналізу можна зробити висновок, що найбільш перспективним підходом для задачі реконструкції обличчя на основі відеозапису є використання методів глибокого навчання. Такі методи не потребують спеціалізованого обладнання, можуть працювати з даними, отриманими за допомогою звичайних камер, та забезпечують достатню точність реконструкції. Крім того, використання відеопослідовностей дозволяє підвищити стабільність результатів шляхом об’єднання інформації з декількох кадрів.

Отже, у даній роботі для реалізації системи реконструкції тривимірної моделі обличчя доцільно використовувати підхід, що базується на алгоритмах глибокого навчання у поєднанні зі статистичними моделями обличчя. Такий підхід забезпечує оптимальний баланс між точністю реконструкції, швидкістю обробки даних та вимогами до апаратних ресурсів.

1.4 Постановка задачі та вимоги до системи

На основі аналізу предметної області та існуючих методів реконструкції тривимірних моделей обличчя основною задачею роботи є розробка програмної системи, що забезпечує реконструкцію тривимірної моделі обличчя людини на основі відеозапису з використанням методів комп’ютерного зору та алгоритмів глибокого навчання.

Використання відеозапису як джерела даних дозволяє отримувати зображення обличчя з різних ракурсів, що підвищує точність реконструкції моделі. Отримані тривимірні моделі можуть застосовуватися у біометричних системах автентифікації, комп’ютерній графіці, технологіях віртуальної та доповненої реальності, а також при створенні персоналізованих тривимірних аватарів.

Для досягнення поставленої мети система повинна виконувати обробку відеопослідовності з виділенням кадрів, детекцію обличчя та визначення опорних точок для вирівнювання зображення. Після цього здійснюється відбір кадрів відповідно до критеріїв якості та положення голови. На основі відібраних кадрів за допомогою попередньо навченої нейронної мережі визначаються параметри тривимірної моделі обличчя. Результати обробки декількох кадрів можуть об’єднуватися для підвищення стабільності та точності реконструкції.

Заключним етапом роботи системи є формування тривимірної сітки обличчя (mesh) та її експорт у стандартний формат тривимірної графіки для подальшого використання.

З урахуванням поставлених задач до розроблюваної системи висуваються такі основні вимоги.

Функціональні вимоги:

- отримання кадрів із відеозапису з фільтрацією за якістю – відбраковування розмитих та дубльованих кадрів;
- детекція обличчя на кожному кадрі, оцінка кута повороту голови та відсів кадрів із неприпустимим поворотом;
- вирівнювання області обличчя за ключовими точками;
- передбачення параметрів форми FLAME для кожного кадру за допомогою MICA;
- об’єднання (злиття) параметрів форми з кількох кадрів в єдиний стійкий вектор;
- декодування підсумкового вектора форми у нейтральну тривимірну сітку;
- постобробка та експорт сітки у формат OBJ.

Вимоги до системи:

- вхідні дані: відеозапис у поширених форматах (MP4, AVI);
- вихідні дані: тривимірна полігональна сітка у форматі OBJ;
- апаратне середовище: персональний комп’ютер з GPU не менше 8 ГБ VRAM (цільова конфігурація: 12 ГБ VRAM, 16 ГБ ОЗП);
- управління пам’яттю: батчева обробка для запобігання переповненню відеопам’яті;
- налаштовуваність: ключові параметри (крок кадрів, пороги фільтрації, параметри згладжування) повинні задаватися через аргументи командного рядка;
- логування: виведення інформації про хід обробки через стандартний механізм logging.

2 ТЕОРЕТИЧНІ ОСНОВИ

У цьому розділі розглядаються теоретичні основи реконструкції тривимірних моделей обличчя. Зокрема, описано параметричну морфовану модель обличчя FLAME, що використовується для представлення геометрії обличчя, а також сучасні підходи на основі глибокого навчання, включаючи нейронну мережу MICA та алгоритм детекції обличчя RetinaFace. Окрему увагу приділено метрикам оцінки якості вхідних зображень, зокрема дисперсії Лапласіана та індексу структурної подібності SSIM, які застосовуються для відбору інформативних кадрів.

2.1 Морфована модель обличчя FLAME

Морфовані моделі обличчя (3D Morphable Models) є одним із найбільш поширених інструментів для опису тривимірної геометрії людського обличчя. Такі моделі дозволяють представити форму обличчя у вигляді параметризованої математичної моделі, яка може змінюватися відповідно до індивідуальних особливостей людини, міміки та положення голови [11].

Однією з сучасних і широко використовуваних моделей є FLAME (Faces Learned with an Articulated Model and Expressions). Вона була запропонована дослідниками Інституту Макса Планка і побудована на основі великого набору тривимірних сканів людських облич [12]. Модель дозволяє відокремлено керувати параметрами форми, міміки та пози голови, що робить її зручною для використання у задачах реконструкції обличчя, комп’ютерної графіки та анімації.

Основною особливістю моделі FLAME є використання артикульованої структури, що дозволяє враховувати рухи щелепи та інші деформації обличчя.

Це робить модель більш реалістичною порівняно з класичними морфованими моделями, які описують лише статичну форму обличчя.

Модель FLAME представляє поверхню обличчя у вигляді тривимірної полігональної сітки (mesh), що складається з фіксованої кількості вершин і трикутників. Зміна параметрів моделі призводить до зміни координат цих вершин, що дозволяє отримувати різні варіанти геометрії обличчя. Завдяки цьому модель може використовуватися як універсальний інструмент для генерації та реконструкції тривимірних моделей обличчя [12].

У сучасних системах реконструкції обличчя, зокрема в нейронних мережах MICA або DECA, модель FLAME використовується як базове параметричне представлення обличчя, параметри якого визначаються за допомогою алгоритмів машинного навчання на основі двовимірних зображень або відеоданих [13].

2.1.1 Математичне описання моделі

Модель FLAME представляє тривимірну геометрію обличчя як функцію декількох наборів параметрів, що описують форму, міміку та положення голови. Загалом модель можна представити у вигляді функції

$$M(\beta,\psi,\theta), \tag{2.1}$$

де β − параметри форми обличчя;

ψ − параметри міміки (виразів);

θ − параметри пози та артикуляції.

Геометрія обличчя задається координатами вершин тривимірної сітки:

$$V = T + Bs(\beta) + Be(\psi), \tag{2.2}$$

де T − базова форма обличчя (template mesh);

$Bs(\beta)$ − деформації, що описують індивідуальну форму обличчя;

$Be(\psi)$ − деформації, що відповідають міміці.

Базова форма T є середньою геометрією обличчя, отриманою на основі статистичного аналізу великої кількості 3D-сканів. Вона складається з приблизно 5000 вершин, що утворюють трикутну полігональну сітку [12].

Параметри форми β визначають індивідуальні геометричні особливості обличчя людини, такі як форма щелепи, розмір носа, ширина обличчя та інші морфологічні характеристики. Ці параметри отримуються за допомогою методу аналізу головних компонент (PCA), що дозволяє представити форму обличчя у вигляді лінійної комбінації базових компонент [11, 14].

Параметри міміки ψ описують деформації обличчя, що виникають під час різних виразів, наприклад посмішки, здивування або нахмурення. Для їх моделювання використовується набір базових виразів, отриманих на основі аналізу динамічних 3D-сканів обличчя.

Параметри пози θ визначають положення голови та рухи щелепи. У моделі FLAME використовується скелетна структура, яка дозволяє описувати артикуляцію окремих частин обличчя. Це дає можливість моделювати природні рухи обличчя та забезпечує більш реалістичну реконструкцію [12].

Після визначення всіх параметрів обчислюється остаточне положення вершин сітки, яке формується шляхом застосування лінійної комбінації деформацій та подальшої трансформації відповідно до параметрів пози. Отримана геометрія може бути використана для побудови тривимірної моделі обличчя або подальшого рендерингу.

Таким чином, модель FLAME забезпечує компактне та математично зручне представлення геометрії людського обличчя, що робить її ефективним інструментом для задач реконструкції, аналізу та синтезу облич у системах комп’ютерного зору та комп’ютерної графіки.

2.1.2 Параметри форми, виразів і пози

У морфованій моделі обличчя FLAME геометрія обличчя описується за допомогою набору параметрів, які відповідають за індивідуальну форму обличчя, міміку та положення голови. Таке параметричне представлення дозволяє компактно описати складну тривимірну структуру обличчя та ефективно керувати її змінами. Це, у свою чергу, значно спрощує задачу реконструкції, оскільки складна геометрія зводиться до оцінки відносно невеликої кількості параметрів. Основними групами параметрів у моделі FLAME є параметри форми, параметри виразів обличчя та параметри пози (артикуляції) [12].

Параметри форми (β) описують індивідуальні геометричні особливості обличчя людини. До таких особливостей належать загальна форма черепа, ширина та довжина обличчя, форма носа, положення вилиць, форма підборіддя та інші морфологічні характеристики. У моделі FLAME ці параметри отримуються шляхом статистичного аналізу великої кількості тривимірних сканів обличчя різних людей. Для цього використовується метод аналізу головних компонент (Principal Component Analysis, PCA), який дозволяє виділити основні напрямки варіації форми обличчя в наборі даних [11].

У результаті такого аналізу формується набір базових векторів, які описують можливі зміни геометрії обличчя. Кожне конкретне обличчя може бути представлене як лінійна комбінація цих базових компонент [14]. Кількість параметрів форми зазвичай становить близько 100 компонент, що дозволяє достатньо точно описувати індивідуальні особливості обличчя при відносно невеликій кількості параметрів.

Параметри виразів обличчя (ψ) відповідають за моделювання мімічних деформацій. Людське обличчя є дуже динамічною структурою, здатною до складних рухів під час емоційних виразів, мовлення або жування. Для коректного моделювання цих процесів у моделі FLAME використовується

набір базових виразів, отриманих на основі аналізу динамічних 3D-сканів обличчя [12].

Параметри виразів дозволяють описувати такі мімічні зміни, як посмішка, підняття брів, відкривання рота або зморщування лоба. Зазвичай модель використовує близько 50 параметрів виразів, які визначають інтенсивність відповідних деформацій. Комбінація цих параметрів дозволяє відтворювати широкий спектр мімічних рухів обличчя. Це особливо важливо для задач комп'ютерної анімації, синтезу міміки та реконструкції обличчя з відеоданих [11].

Параметри пози (θ) описують положення та орієнтацію голови у просторі, а також рухи окремих частин обличчя, зокрема нижньої щелепи. На відміну від параметрів форми та виразів, які відповідають за деформації поверхні, параметри пози визначають геометричні трансформації всієї моделі або її окремих частин. У моделі FLAME для цього використовується скелетна (articulated) структура, яка дозволяє враховувати рухи щелепи та шиї [12].

Параметри пози включають кути обертання голови навколо трьох осей, що відповідають нахилу, повороту та обертанню голови. Також враховується положення нижньої щелепи, яке має значний вплив на форму обличчя під час мовлення або відкривання рота. Такий підхід дозволяє більш точно моделювати природні рухи обличчя та забезпечує реалістичність реконструйованих моделей.

У практичних задачах реконструкції обличчя параметри форми, виразів і пози визначаються за допомогою алгоритмів комп'ютерного зору та нейронних мереж. Наприклад, сучасні системи реконструкції обличчя використовують згорткові нейронні мережі, які аналізують зображення або відео та прогнозують відповідні параметри моделі FLAME. Після цього на основі отриманих параметрів обчислюється геометрія тривимірної сітки обличчя [13]. Фактично, задача реконструкції зводиться до оцінювання цих параметрів, що зменшує обчислювальну складність задачі порівняно з прямим відновленням геометрії.

Таким чином, використання параметричної моделі FLAME дозволяє компактно та ефективно описувати форму та динаміку людського обличчя. Поділ параметрів на групи форми, виразів та пози забезпечує гнучкість моделі та дозволяє застосовувати її у широкому спектрі задач, включаючи реконструкцію обличчя, біометричну ідентифікацію, комп'ютерну графіку та створення цифрових аватарів.

2.2 Нейронна мережа MICA та детектор RetinaFace

Сучасні системи реконструкції тривимірних моделей обличчя все частіше базуються на використанні методів глибокого навчання. Завдяки здатності нейронних мереж автоматично виявляти складні залежності між двовимірними зображеннями та тривимірною геометрією обличчя, такі підходи дозволяють значно підвищити точність та ефективність реконструкції [13, 15].

У задачах реконструкції обличчя зазвичай використовується поєднання двох основних компонентів:

- алгоритм детекції обличчя та визначення ключових точок;
- нейронна мережа для відновлення тривимірної геометрії.

На першому етапі застосовуються алгоритми детекції обличчя, які забезпечують локалізацію обличчя на зображенні та визначення його ключових точок. Ці опорні точки використовуються для приведення обличчя до канонічного вигляду шляхом вирівнювання, що забезпечує узгодженість вхідних даних для подальшої обробки. Як один із сучасних і ефективних підходів у цій задачі може використовуватися алгоритм RetinaFace, який демонструє високу точність детекції навіть у складних умовах [16].

Після етапів детекції та вирівнювання обличчя виконується реконструкція його тривимірної геометрії за допомогою нейронних мереж на основі вхідних зображень. Одним із сучасних і ефективних підходів є модель

MICA (Model-based Identity Consistent Architecture), яка здійснює прогнозування параметрів морфованої моделі обличчя (FLAME), що дозволяє відновити тривимірну форму обличчя з урахуванням індивідуальних характеристик людини та забезпечити узгодженість результатів для різних зображень однієї особи [13].

2.2.1 Архітектура та ключові точки RetinaFace

RetinaFace є сучасним алгоритмом детекції обличчя, який належить до класу однокрокових (single-shot) детекторів і забезпечує високу точність локалізації облич навіть у складних умовах зйомки. Він був розроблений як розширення підходів на основі згорткових нейронних мереж і враховує як глобальні, так і локальні ознаки зображення для точного визначення положення обличчя та його ключових точок [16]. Завдяки цьому RetinaFace здатний ефективно працювати із зображеннями, отриманими у реальних умовах, де присутні шуми, складне освітлення та варіації пози обличчя.

Архітектура RetinaFace базується на використанні глибокої згорткової нейронної мережі як екстрактора ознак. У якості базової мережі зазвичай використовується ResNet або MobileNet, які дозволяють отримувати багаторівневі ознаки зображення. Для обробки об’єктів різного масштабу застосовується піраміда ознак (Feature Pyramid Network, FPN), яка поєднує інформацію з різних рівнів мережі та дозволяє ефективно детектувати обличчя як великого, так і малого розміру [16].

На відміну від класичних детекторів, RetinaFace використовує багатозадачний підхід, який одночасно виконує кілька завдань:

- класифікацію (визначення наявності обличчя);
- регресію координат обмежувального прямокутника (bounding box);
- визначення ключових точок обличчя;
- додаткове прогнозування щільних ознак (у деяких реалізаціях).

Такий підхід дозволяє підвищити точність детекції та забезпечити більш повну інформацію про обличчя вже на етапі його локалізації.

Однією з ключових особливостей RetinaFace є використання anchor-based механізму детекції, при якому на різних рівнях зображення розміщуються так звані «якорі» (anchors), що відповідають можливим положенням об’єктів. Для кожного такого якоря мережа прогнозує ймовірність наявності обличчя та уточнює його координати. Це дозволяє ефективно знаходити обличчя навіть при значних варіаціях масштабу та положення.

Важливим компонентом RetinaFace є визначення п’яти ключових точок обличчя (facial landmarks). До них належать:

- центр лівого ока;
- центр правого ока;
- кінчик носа;
- ліва точка рота;
- права точка рота.

Ці точки відіграють важливу роль у подальшій обробці обличчя. Зокрема, вони використовуються для вирівнювання (alignment) обличчя, що дозволяє привести різні зображення до єдиного канонічного вигляду та спрощує подальші етапи аналізу, включаючи реконструкцію тривимірної моделі. Крім того, стабільність визначення цих точок безпосередньо впливає на точність наступних етапів обробки.

Ще однією перевагою RetinaFace є його здатність працювати в умовах «in-the-wild», тобто при наявності складного освітлення, часткових перекриттів, різних ракурсів та виразів обличчя. Це досягається завдяки використанню великого навчального набору даних та спеціальних функцій втрат, які враховують різні аспекти детекції [16].

Крім того, RetinaFace демонструє високу продуктивність і може працювати в реальному часі при використанні сучасних графічних процесорів.

Це робить його придатним для інтерактивних систем, мобільних додатків та інших задач, де важлива швидкість обробки даних.

Таким чином, алгоритм RetinaFace є ефективним інструментом для детекції обличчя та визначення ключових точок, що забезпечує високу точність і стабільність результатів. Його використання є важливим етапом у системах реконструкції тривимірних моделей обличчя, оскільки якість подальших етапів значною мірою залежить від точності локалізації обличчя та визначення його ключових характеристик.

2.2.2 Архітектура і принцип роботи MICA

Модель MICA (Model-based Identity Consistent Architecture) є сучасною нейронною мережею, призначеною для реконструкції тривимірної геометрії обличчя на основі двовимірних зображень. Вона належить до класу методів, що поєднують глибинне навчання зі статистичними морфованими моделями обличчя, зокрема моделлю FLAME [13].

Основною ідеєю MICA є прогнозування параметрів тривимірної моделі обличчя, які описують індивідуальні особливості людини. На відміну від попередніх підходів, MICA орієнтована на збереження ідентичності особи, що забезпечує отримання більш точних та стабільних результатів реконструкції навіть за різних умов зйомки [13].

Архітектурно модель MICA складається з декількох основних компонентів. Першим етапом є екстракція ознак зображення за допомогою згорткової нейронної мережі (CNN) [15]. Для цього зазвичай використовується попередньо навчена мережа, наприклад ResNet, яка дозволяє отримати компактне представлення вхідного зображення у вигляді вектора ознак.

На наступному етапі отримані ознаки подаються на регресійний блок, який прогнозує параметри морфованої моделі обличчя. До таких параметрів належать параметри форми, що визначають індивідуальну геометрію обличчя.

У деяких варіаціях моделі також можуть прогнозуватися параметри текстури або освітлення.

Особливістю MICA є використання механізму узгодженості ідентичності (identity consistency). Це означає, що модель навчається таким чином, щоб отримані параметри форми залишалися стабільними для різних зображень однієї і тієї ж людини. Такий підхід дозволяє зменшити вплив змін освітлення, ракурсу та міміки на результат реконструкції.

Після визначення параметрів моделі виконується їх передача у декодер, який використовує модель FLAME для обчислення координат вершин тривимірної сітки обличчя. Таким чином, нейронна мережа фактично виконує задачу переходу від двовимірного зображення до параметричного опису тривимірної геометрії [13].

Важливою перевагою MICA є можливість роботи з одиночними зображеннями, а також висока швидкість обробки даних. Це робить модель придатною для використання у реальних системах, включаючи інтерактивні додатки та системи реального часу.

Таким чином, нейронна мережа MICA є ефективним інструментом для реконструкції тривимірних моделей обличчя, оскільки поєднує переваги глибинного навчання та параметричних моделей, забезпечуючи високу точність та стабільність результатів.

2.3 Метрики якості кадрів: дисперсія Лапласіана та SSIM

Якість вхідних зображень є одним із ключових факторів, що впливають на точність реконструкції тривимірних моделей обличчя. Низька різкість, наявність шуму, розмиття або надмірна схожість сусідніх кадрів у відеопослідовності можуть суттєво погіршити результат відновлення геометрії. Тому перед обробкою даних важливо виконати оцінку якості кадрів та відбір найбільш інформативних із них. Для цього широко застосовуються

різні метрики, серед яких одними з найбільш поширених є дисперсія Лапласіана та індекс структурної подібності (SSIM) [17, 18].

Дисперсія Лапласіана використовується як міра різкості зображення. Вона базується на застосуванні оператора Лапласа, який є другим похідним від інтенсивності зображення і дозволяє виділяти області з високою частотою змін, тобто границі об’єктів. Для обчислення метрики зображення спочатку переводиться у відтінки сірого, після чого до нього застосовується оператор Лапласа. Отримане зображення містить значення, що характеризують інтенсивність змін яскравості [18].

Кількісна оцінка різкості визначається як дисперсія отриманого зображення:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(L_i - \mu)^2 \qquad (2.3)$$

де L_i − значення пікселів після застосування оператора Лапласа;

μ − середнє значення;

N − кількість пікселів.

Чим більша дисперсія, тим більше в зображенні різких переходів, а отже, тим воно чіткіше. Розмиті зображення мають менші значення дисперсії, оскільки високочастотні компоненти згладжуються [19]. Таким чином, дана метрика дозволяє ефективно відфільтровувати кадри з розмиттям, що є важливим при обробці відео, де можуть виникати ефекти руху або фокусування.

Разом із тим, дисперсія Лапласіана не враховує структурну схожість між кадрами. Вона оцінює лише локальні зміни інтенсивності, але не дозволяє визначити, наскільки два зображення відрізняються між собою. Саме тому для задач відбору кадрів доцільно використовувати додаткову метрику − індекс структурної подібності.

SSIM (Structural Similarity Index) є метрикою, що оцінює ступінь подібності між двома зображеннями з урахуванням їх структури, контрасту та

яскравості [17]. На відміну від простих метрик, таких як середньоквадратична помилка (MSE), SSIM орієнтований на особливості сприйняття зображення людиною.

Загальна формула SSIM має вигляд:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \quad (2.4)$$

де μ_x, μ_y − середні значення інтенсивності;

σ_x^2, σ_y^2 – дисперсії;

σху − коваріація між зображеннями;

C_1, C_2 − стабілізуючі константи.

Значення SSIM знаходиться у діапазоні від 0 до 1, де 1 відповідає повній ідентичності зображень. Низькі значення SSIM свідчать про суттєві відмінності між зображеннями.

У задачах обробки відео SSIM використовується для виявлення надто схожих кадрів. Якщо значення SSIM між поточним кадром та попереднім перевищує певний поріг, такі кадри вважаються надлишковими та можуть бути виключені з подальшої обробки. Це дозволяє зменшити обчислювальні витрати та уникнути дублювання інформації.

Комбіноване використання дисперсії Лапласіана та SSIM дозволяє ефективно вирішувати задачу відбору кадрів. Перша метрика забезпечує відбір чітких, нерозмитих зображень, тоді як друга − усуває надлишкові кадри, що містять практично однакову інформацію. Такий підхід дозволяє сформувати набір якісних і різноманітних зображень, що є важливим для підвищення точності реконструкції тривимірної моделі обличчя.

Таким чином, використання метрик дисперсії Лапласіана та SSIM є ефективним інструментом попередньої обробки відеоданих. Вони дозволяють покращити якість вхідної інформації, зменшити обсяг даних та підвищити стабільність і точність подальших етапів реконструкції обличчя.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

У даному розділі розглядається процес проєктування та програмної реалізації системи реконструкції тривимірної моделі обличчя на основі відеозапису. Основна увага приділяється побудові послідовного обчислювального конвеєру (пайплайну), який забезпечує перетворення вхідних відеоданих у тривимірну геометричну модель обличчя користувача.

Реалізована система складається з кількох взаємопов’язаних етапів, кожен з яких виконує окрему функцію обробки даних. Зокрема, система включає етапи вилучення та фільтрації кадрів, детекції обличчя, оцінки положення голови, вирівнювання зображень, реконструкції тривимірної форми, злиття параметрів та постобробки отриманого меша. Такий підхід дозволяє розділити складну задачу на низку більш простих підзадач, що спрощує реалізацію та підвищує надійність роботи системи.

Проєктування системи здійснювалося з урахуванням вимог до точності реконструкції, обчислювальної ефективності та можливості роботи з даними, отриманими в умовах реального середовища. Особлива увага приділялася мінімізації впливу шумів, розмиття та варіацій положення обличчя у відеопотоці.

3.1 Загальна архітектура пайплайну та вибір програмних засобів

Архітектура розробленої системи базується на послідовному пайплайні, в якому кожен етап обробки виконується незалежним модулем і передає результати наступному етапу. Така модульна структура забезпечує гнучкість системи, можливість її масштабування та спрощує подальшу модифікацію окремих компонентів.

Загальна схема роботи пайплайну включає такі основні етапи:

- вилучення кадрів з відеозапису з попередньою фільтрацією якості;
- детекція обличчя та визначення ключових точок;
- оцінка положення голови та відсів неінформативних кадрів;
- вирівнювання обличчя до канонічного вигляду;
- реконструкція тривимірної форми обличчя для кожного кадру;
- злиття параметрів форми з різних кадрів;
- декодування фінальної тривимірної моделі та її експорт.

Кожен із зазначених етапів реалізований з використанням сучасних бібліотек комп'ютерного зору та машинного навчання. Для обробки зображень і відео використовується бібліотека OpenCV, яка забезпечує ефективні засоби роботи з кадрами, перетворення кольорових просторів та геометричних операцій. Для оцінки якості кадрів застосовується бібліотека scikit-image, зокрема метрика структурної подібності (SSIM).

Детекція обличчя та визначення ключових точок реалізовані за допомогою фреймворку InsightFace, який інтегрує моделі глибокого навчання для аналізу обличчя. Використання цього фреймворку дозволяє автоматизувати виділення облич на зображеннях, отримання ключових точок, необхідних для вирівнювання та приведення обличчя до канонічного вигляду. Такий підхід забезпечує узгодженість вхідних даних для подальшої обробки та створює основу для точного аналізу та реконструкції 3D-моделей обличчя.

Для реконструкції тривимірної форми обличчя використовується нейронна мережа MICA, яка прогнозує параметри морфованої моделі FLAME. Обчислення виконуються з використанням бібліотеки PyTorch, що дозволяє ефективно задіяти графічний процесор для прискорення обробки.

Постобробка та експорт тривимірної моделі реалізовані за допомогою бібліотеки Trimesh, яка забезпечує інструменти для очищення геометрії та збереження результатів у стандартному форматі OBJ.

Таким чином, запропонована архітектура поєднує сучасні алгоритми комп'ютерного зору та глибокого навчання, що дозволяє отримати точну та стабільну реконструкцію тривимірної моделі обличчя на основі відеоданих.

3.2 Отримання та фільтрація кадрів

Одним із ключових етапів побудови системи реконструкції тривимірної моделі обличчя є отримання інформативних кадрів із відеозапису та їх подальша фільтрація. Відеопослідовність зазвичай містить велику кількість кадрів, частина з яких є надлишковими або має низьку якість через розмиття, рух камери чи зміни освітлення. Використання таких кадрів може негативно вплинути на точність наступних етапів обробки та призводить до збільшення обчислювальних витрат.

З метою підвищення якості вхідних даних застосовується комбінований підхід, що включає страйд-семплінг, фільтрацію за різкістю та усунення надлишкових кадрів на основі метрики SSIM.

Наведений на рисунку 3.1 фрагмент коду реалізує послідовний процес відбору кадрів із відеопотоку, який поєднує декілька критеріїв якості. Такий підхід дозволяє сформувати компактний та інформативний набір зображень, що використовується на наступних етапах реконструкції.

3.2.1 Страйд-семплінг

Як видно з рисунку 3.1, первинне зменшення кількості кадрів здійснюється за допомогою страйд-семплінгу. Його реалізація базується на перевірці індексу поточного кадру відносно заданого кроку (frame_stride). Якщо умова кратності не виконується, кадр пропускається.

Такий підхід дозволяє уникнути обробки надлишкових кадрів, які практично не відрізняються між собою, особливо у відео з високою частотою кадрів. Вибір значення параметра frame_stride визначає баланс між швидкістю обробки та повнотою представленої інформації. Зменшення кількості кадрів знижує обчислювальне навантаження на систему та пришвидшує виконання наступних етапів пайплайну.

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
    # Обробляємо кожен N-й кадр
    if raw_idx % frame_stride != 0:
        raw_idx += 1
        continue
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Фільтр різкості
    lap_var = _sharpness(gray)
    if lap_var < sharpness_threshold:
        raw_idx += 1
        continue
    # SSIM-фільтрація
    if last_gray is not None:
        similarity = ssim(last_gray, gray, data_range=255)
        if similarity >= ssim_threshold:
            raw_idx += 1
            continue
    frames.append(frame.copy())
    last_gray = gray
    raw_idx += 1
    if len(frames) >= max_frames:
        break
```

Рисунок 3.1 – Алгоритм отримання та фільтрації кадрів

3.2.2 Фільтрація за різкістю та SSIM

Подальша обробка кадрів, як показано рисунку 3.1, включає фільтрацію за якісними характеристиками зображення. На першому етапі оцінюється різкість кадру за допомогою дисперсії Лапласіана. Для цього зображення переводиться у відтінки сірого, після чого до нього застосовується оператор Лапласа, що дозволяє виділити області з різкими змінами інтенсивності. Чим вищим є значення дисперсії отриманого зображення, тим більш чітким вважається кадр. Кадри з низьким значенням різкості відкидаються як розмиті та неінформативні, оскільки вони можуть призвести до погіршення результатів на етапах детекції обличчя та реконструкції геометрії [19].

Важливим аспектом є вибір порогового значення параметра sharpness_threshold. Занадто низьке значення може призвести до збереження розмитих кадрів, тоді як занадто високе – до відкидання корисних зображень. Тому даний параметр підбирається експериментально з урахуванням характеристик вхідного відео.

Після відбору кадрів за різкістю застосовується фільтрація за допомогою метрики SSIM, яка дозволяє визначити ступінь структурної подібності між поточним та попереднім кадром. На відміну від простих метрик різниці пікселів, SSIM враховує яскравість, контраст та структуру зображення, що робить її більш придатною для оцінки візуальної схожості. Якщо значення SSIM перевищує заданий поріг, кадр вважається надто схожим і виключається з подальшої обробки.

Застосування SSIM є особливо важливим у випадках, коли відео містить повільні зміни положення обличчя. У таких умовах навіть після страйд-семплінгу можуть залишатися кадри з мінімальними відмінностями, які не додають нової інформації для реконструкції. Видалення таких кадрів дозволяє зменшити обчислювальне навантаження та підвищити різноманітність вибірки [10].

Комбіноване використання фільтрації за різкістю та SSIM дозволяє одночасно усунути як неякісні, так і надлишкові кадри. У результаті формується компактний та інформативний набір зображень, який забезпечує стабільність роботи системи та підвищує точність реконструкції тривимірної моделі обличчя.

3.3 Детекція, оцінка пози голови та вирівнювання обличчя

Після формування набору якісних кадрів виконується процес детекції обличчя, оцінки положення голови та вирівнювання зображень. Ці операції є критично важливими, оскільки точність реконструкції тривимірної моделі

значною мірою залежить від правильності локалізації обличчя та узгодженості його орієнтації у всіх кадрах.

На даному етапі використовується фреймворк InsightFace, який забезпечує детекцію обличчя та визначення п’яти ключових точок. Отримані координати дозволяють не лише локалізувати обличчя, але й оцінити його орієнтацію у просторі та виконати подальше вирівнювання.

3.3.1 Детекція та відсів кадрів за кутами yaw і pitch

Після детекції обличчя для кожного кадру визначаються ключові точки: положення очей, носа та куточків рота. На основі цих точок виконується оцінка кута повороту голови відносно камери. Основна мета цього етапу – відсіяти кадри, на яких обличчя знаходиться під надто великим кутом, що може негативно вплинути на якість реконструкції. Фрагмент коду для реалізації обчислення кутів нахилу обличча наведено на рисунку 3.2.

```
# Отримання додаткових параметрів на основі ключових точок
eye_center    = (left_eye + right_eye) / 2.0
mouth_center  = (mouth_left + mouth_right) / 2.0
eye_width     = np.linalg.norm(right_eye - left_eye)
nose_offset_x = nose[0] - eye_center[0]
# Обчислення кута нахилу обличча yaw
yaw_deg = float(np.degrees(np.arctan2(nose_offset_x, eye_width * 0.8)))
face_height = mouth_center[1] - eye_center[1]
# Обчислення кута нахилу обличча pitch
if face_height < 1e-3:
    pitch_deg = 0.0
else:
    nose_rel  = (nose[1] - eye_center[1]) / face_height - 0.45
    pitch_deg = float(np.degrees(np.arctan2(nose_rel * face_height,
                                            face_height * 0.5)))
```

Рисунок 3.2 – Реалізація обчислення кутів повороту обличчя

Оцінка кута повороту виконується за допомогою геометричних співвідношень між ключовими точками. Зокрема, кут yaw (поворот

вліво/вправо) визначається як зміщення носа відносно центру між очима, а кут pitch (нахил вгору/вниз) – як відносне положення носа між лініями очей і рота.

Якщо значення кутів перевищують задані пороги (наприклад, yaw > 40° або pitch > 30°), відповідний кадр відкидається. Це дозволяє виключити з подальшої обробки зображення з екстремальними ракурсами, які можуть призвести до спотворення тривимірної моделі.

Таким чином, на цьому етапі формується підмножина кадрів, що відповідають умовам фронтального або близького до фронтального положення обличчя.

3.3.2 Афінне вирівнювання за ключовими точками

Після відбору кадрів виконується вирівнювання обличчя до канонічного вигляду. Це необхідно для того, щоб усі зображення мали однакову орієнтацію та масштаб, що суттєво спрощує подальшу реконструкцію.

Вирівнювання здійснюється на основі п’яти ключових точок, отриманих під час детекції. Для цього використовується афінне перетворення, яке дозволяє привести координати точок обличчя до заздалегідь визначеного еталонного положення [20].

У коді, зображеному на рисунку 3.3, функція estimateAffinePartial2D обчислює матрицю афінного перетворення, яка мінімізує відхилення між поточними координатами ключових точок та еталонними значеннями. Далі за допомогою функції warpAffine виконується трансформація зображення.

```
transform, _ = cv2.estimateAffinePartial2D(
    landmarks, ref_lm, method=cv2.LMEDS
)
if transform is None:
    continue
aligned = cv2.warpAffine(
    frame, transform, (output_size, output_size),
)
face_crops.append(aligned)
```

Рисунок 3.3 – Реалізації афінного вирівнювання

Афінне вирівнювання дозволяє нормалізувати положення обличчя, усуваючи вплив поворотів, зміщень та масштабування. Це забезпечує узгодженість вхідних даних для нейронної мережі та підвищує стабільність і точність реконструкції тривимірної моделі.

Таким чином, етап детекції, оцінки пози та вирівнювання обличчя формує стандартизований набір зображень, який є оптимальним для подальшої обробки в системі реконструкції.

3.4 Реконструкція 3D-форми за допомогою MICA

На основі результатів детекції, відбору та вирівнювання кадрів здійснюється реконструкція тривимірної форми обличчя. Для цього використовується нейронна мережа MICA (Model-based Identity Consistent Architecture), яка призначена для відновлення геометрії обличчя на основі одного або декількох зображень.

MICA є моделлю, що базується на параметричному представленні обличчя (FLAME) та дозволяє оцінювати параметри форми, які описують індивідуальні геометричні особливості людини. На відміну від класичних методів, ця модель забезпечує стабільність реконструкції навіть при зміні ракурсу або освітлення, що є особливо важливим при роботі з відеоданими.

Реконструкція тривимірної форми проводиться для кожного вирівняного кадру, після чого отримані результати об’єднуються, що дозволяє сформувати фінальну узгоджену модель обличчя.

3.4.1 Пакетна обробка з контролем відеопам’яті

Обробка великої кількості зображень за допомогою нейронної мережі потребує значних обчислювальних ресурсів, особливо при використанні графічного процесора (GPU). Обробка великої кількості зображень одночасно

може призвести до переповнення відеопам’яті, тому в системі використовується пакетна (batch) обробка даних.

Суть підходу полягає у розбитті вхідного набору зображень на невеликі групи фіксованого розміру (chunk_size), які обробляються послідовно. Це дозволяє контролювати використання пам’яті та забезпечити стабільну роботу системи навіть на обмежених ресурсах.

Для реалізації цього підходу у кожній ітерації циклу (рисунок 3.4) формується підмножина зображень (chunk), яка проходить попередню обробку: перетворення кольорового простору, нормалізацію та перетворення у формат тензорів PyTorch. Отримані дані формуються в два набори різного розміру (224×224 та 112×112), що обумовлено особливостями архітектури MICA [21].

```
for start in tqdm(
    range(0, len(images_bgr), self.chunk_size),
    desc="Reconstructing 3D faces",
    unit="chunk",
):
    chunk = images_bgr[start : start + self.chunk_size]
    imgs_rgb = [cv2.cvtColor(im, cv2.COLOR_BGR2RGB) for im in chunk]
    # [отримання imgs_224 з imgs_rgb шляхом масштабування, деталі наведено в додатку Б]
    batch = torch.stack(
        [torch.from_numpy(im.astype(np.float32) / 255.0).permute(2, 0, 1)
         for im in imgs_224],
        dim=0,
    ).to(self.device)
    # [отримання imgs_112 з imgs_rgb, деталі наведено в додатку Б]
    arcface_batch = torch.stack(
        [torch.from_numpy(im.astype(np.float32) / 255.0).permute(2, 0, 1)
         for im in imgs_112],
        dim=0,
    ).to(self.device)
    arcface_batch = (arcface_batch - 0.5) / 0.5
    # Прямий прохід MICA
    codedict = self.model.encode(batch, arcface_batch)
    opdict   = self.model.decode(codedict)
    # Отримання результатів реконструкції
    verts  = opdict["pred_canonical_shape_vertices"].cpu().numpy()
    shapes = opdict["pred_shape_code"].cpu().numpy()
    for i in range(verts.shape[0]):
        all_verts.append(verts[i])
        all_shape.append(shapes[i])
    # Звільнення відеопам'яті GPU перед обробкою наступного чанка
    del batch, arcface_batch, codedict, opdict
    torch.cuda.empty_cache()
```

Рисунок 3.4 – Алгоритм пакетної обробки зображень
в моделі MICA з контролем використання відеопам’яті

На наступному кроіці виконується прямий прохід нейронної мережі за допомогою функцій encode та decode, у результаті чого отримуються вершини тривимірної моделі та параметри форми обличчя. Отримані результати для кожного пакета послідовно накопичуються у відповідних структурах даних для їх подальшої обробки та аналізу [10, 22].

Важливим елементом є звільнення відеопам’яті після обробки кожного пакета. Для цього видаляються проміжні змінні та викликається функція torch.cuda.empty_cache(), що дозволяє уникнути переповнення пам’яті при обробці великої кількості кадрів.

Таким чином, використання пакетної обробки забезпечує ефективне використання обчислювальних ресурсів і дозволяє масштабувати систему без втрати стабільності роботи.

3.4.2 Медіанне злиття параметрів та декодування меша

Оскільки реконструкція тривимірної форми обличчя виконується для кожного кадру окремо, отримані параметри можуть містити незначні відхилення, спричинені шумами, змінами освітлення або похибками детекції. Щоб забезпечити стабільність і узгодженість моделі, параметри форми об’єднують між усіма обробленими кадрами.

Для об’єднання параметрів форми застосовано метод медіанного злиття (median fusion), який є стійким до окремих аномальних значень. На відміну від середнього значення, медіана менш чутлива до одиничних помилкових оцінок, що можуть виникати у складних умовах зйомки. Це дозволяє отримати більш надійне узагальнене представлення геометрії обличчя.

Для реалізації методу медіанного злиття всі вектори параметрів форми shape_params_list об’єднуються у матрицю, після чого для кожної компоненти обчислюється медіанне значення (рисунок 3.5). Отриманий вектор fused_shape використовується як узагальнений опис форми обличчя, який може застосовуватися для подальшої реконструкції тривимірної моделі або аналізу.

```
stacked     = np.stack(shape_params_list, axis=0)
fused_shape = np.median(stacked, axis=0)
```

Рисунок 3.5 – Реалізації злиття параметрів

Наступним етапом є декодування отриманих параметрів у тривимірну геометрію. Для цього використовується параметрична модель FLAME, яка перетворює вектор параметрів у набір вершин і граней, що описують поверхню обличчя. При цьому параметри виразів та пози встановлюються у нульові значення, що дозволяє отримати нейтральну форму обличчя без впливу міміки чи рухів (рисунок 3.6).

```
model  = recon_model.model
device = recon_model.device
flame = model.flame
with torch.no_grad():
    shape_t = torch.from_numpy(fused_shape).float().unsqueeze(0).to(device)
    exp  = torch.zeros((1, 100), dtype=torch.float32, device=device)
    pose = torch.zeros((1, 6),   dtype=torch.float32, device=device)
    vertices, _, _ = flame(
        shape_params=shape_t,
        expression_params=exp,
        pose_params=pose,
    )
    verts = vertices.squeeze(0).cpu().numpy()
    faces = flame.faces_tensor.cpu().numpy().astype(np.int32)
```

Рисунок 3.6 – Реалізація декодування 3D-геометрії обличчя

У результаті отримується тривимірна сітка, яка складається з вершин (vertices) та індексів граней (faces). Така структура дозволяє виконувати подальшу обробку, візуалізацію та експорт моделі у стандартні формати.

Таким чином, медіанне злиття параметрів у поєднанні з декодуванням через FLAME забезпечує формування стабільної, узгодженої та геометрично коректної тривимірної моделі обличчя, що є фінальним результатом роботи системи.

3.5 Постобробка та експорт 3D-моделі

Після отримання тривимірної геометрії обличчя у вигляді набору вершин та граней виконується етап постобробки, метою якого є покращення якості меша та підготовка його до подальшого використання. Незважаючи на високу точність реконструкції, отримана модель може містити незначні артефакти, пов’язані з похибками оцінки параметрів або обмеженнями моделі.

У розробленому проекті постобробка виконується з використанням бібліотеки trimesh, яка забезпечує зручні інструменти для роботи з тривимірними сітками.

На першому етапі створюється об’єкт меша на основі отриманих вершин та індексів граней (рисунок 3.7). Параметр process=True активує вбудовані механізми очищення геометрії. Зокрема, виконується видалення дубльованих вершин, некоректних граней та інших дефектів сітки. Це дозволяє підвищити топологічну коректність моделі та підготувати її до подальшої обробки або експорту.

```
mesh = trimesh.Trimesh(vertices=verts, faces=faces, process=True)
# [вивід рядка логування, деталі наведено в додатку Б]
if apply_smoothing:
    trimesh.smoothing.filter_laplacian(
        mesh,
        lamb=smooth_lambda,
        iterations=smooth_iterations,
        implicit_time_integration=False,
        volume_constraint=True,
    )
# [вивід рядка логування, деталі наведено в додатку Б]
out_dir = os.path.dirname(output_path)
    if out_dir:
        os.makedirs(out_dir, exist_ok=True)
mesh.export(output_path)
```

Рисунок 3.7 – Реалізації обробки та експорту 3D-мешу

За потреби може застосовуватися додаткове згладжування поверхні за допомогою лапласівського фільтра. Це дозволяє зменшити високочастотний

шум та зробити поверхню більш гладкою. Згладжування є опціональним етапом і використовується залежно від вимог до якості фінальної моделі. Надмірне згладжування може призвести до втрати дрібних деталей, тому параметри обираються з урахуванням компромісу між гладкістю поверхні та збереженням геометричних особливостей.

Після завершення постобробки виконується експорт меша у файл. У системі використовується стандартний формат OBJ, який підтримується більшістю програм для роботи з тривимірною графікою. Перед збереженням перевіряється наявність цільової директорії, яка за потреби створюється автоматично. Це забезпечує коректну роботу системи незалежно від початкового стану файлової структури.

Таким чином, етап постобробки та експорту забезпечує отримання фінальної тривимірної моделі обличчя у зручному для подальшого використання вигляді. Використання бібліотеки trimesh дозволяє автоматизувати процес очищення та покращення геометрії, що підвищує загальну якість результатів роботи системи.

3.6 Логування та обробка помилок

Для забезпечення стабільної роботи системи реконструкції тривимірної моделі обличчя та зручності аналізу процесу її виконання використовується механізм логування та базова обробка помилок. Оскільки система складається з декількох послідовних етапів (обробка відео, детекція обличчя, реконструкція, постобробка), важливо мати можливість відслідковувати виконання кожного з них та оперативно реагувати на можливі збої.

У реалізованій системі використовується стандартна бібліотека logging, яка дозволяє організувати структурований вивід повідомлень різного рівня важливості. На початковому етапі виконується конфігурація логера із зазначенням рівня повідомлень та формату їх відображення (рисунок 3.8).

```
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(message)s",
    datefmt="%H:%M:%S",
)
logger = logging.getLogger(__name__)
```

Рисунок 3.8 – Налаштування системи логування

Такий формат дозволяє отримувати інформативні повідомлення, що містять часові мітки, рівень важливості та текст повідомлення, що значно спрощує аналіз виконання програми.

У процесі роботи системи використовуються різні рівні логування (рисунок 3.9). Рівень INFO застосовується для фіксації ключових етапів виконання пайплайну, таких як початок обробки відео, кількість отриманих кадрів, запуск реконструкції або завершення роботи системи. Це дозволяє відслідковувати загальний хід виконання програми без необхідності детального аналізу коду. Для повідомлень про помилки використовується рівень ERROR, який сигналізує про виникнення критичних ситуацій, що перешкоджають подальшому виконанню алгоритму. Наприклад, це може бути неможливість відкрити відеофайл або відсутність кадрів після фільтрації.

```
logger.info("Loading MICA on device: %s", self.device)
logger.info("Saved mesh to: %s", output_path)
logger.error("Cannot open video: %s", input_path)
logger.error("Cannot open video: %s", input_path)
```

Рисунок 3.9 – Приклади різних рівнів логування

Крім логування, у системі реалізовано обробку помилок за допомогою механізму виключень (рисунок 3.10). Це дозволяє контролювати коректність виконання кожного етапу та запобігати використанню некоректних даних. Зокрема, перед початком обробки перевіряється доступність відеофайлу, а також наявність кадрів після кожного етапу фільтрації.

```
if not cap.isOpened():
    logger.error("Cannot open video: %s", input_path)
    raise IOError(f"Cannot open video: {input_path}")

if len(frames) == 0:
    logger.error("No frames extracted from video. Aborting.")
    raise RuntimeError("No frames extracted from video.")
```

Рисунок 3.10 – Приклади використання механізму обробки виключень

Такий підхід забезпечує передбачувану поведінку системи та дозволяє уникнути накопичення помилок на наступних етапах обробки.

Додатково використання логування у поєднанні з індикатором прогресу (tqdm) дозволяє отримувати як загальну інформацію про стан системи, так і візуальне відображення процесу обробки великих обсягів даних. Це особливо корисно під час тестування та демонстрації роботи системи.

Таким чином, впровадження механізмів логування та обробки помилок підвищує надійність, керованість та зручність використання програмної системи. Це дозволяє не лише своєчасно виявляти проблеми, але й ефективно аналізувати процес реконструкції тривимірної моделі обличчя.

3.7 Запуск пайплайну та реалізація CLI

Для забезпечення зручності використання розробленої системи реалізовано інтерфейс командного рядка (CLI – Command Line Interface), який дозволяє запускати пайплайн реконструкції з різними параметрами без внесення змін у програмний код. Це робить систему більш гнучкою у налаштуваннях та спрощує її використання в різних сценаріях.

CLI реалізовано за допомогою бібліотеки argparse, яка забезпечує обробку вхідних параметрів, переданих під час запуску програми. Це дозволяє керувати всіма етапами роботи системи – від завантаження даних до постобробки результатів.

Перелік доступних параметрів та їх призначення наведено нижче:

- --input – шлях до вхідного файлу (відео або зображення), який використовується як джерело даних для реконструкції.
- --output – шлях до вихідного файлу у форматі OBJ, у який буде збережено тривимірну модель обличчя.
- --frame_stride – крок вибірки кадрів із відео. Значення N означає, що обробляється кожен N-й кадр, що дозволяє зменшити обсяг даних і пришвидшити обчислення.
- --max_frames – максимальна кількість кадрів, що будуть використані після фільтрації. Обмежує обсяг оброблюваних даних.
- --sharpness_threshold – порогове значення різкості (дисперсії Лапласіана). Кадри з меншим значенням відкидаються як розмиті.
- --ssim_threshold – поріг структурної подібності між сусідніми кадрами. Якщо значення SSIM перевищує цей поріг, кадр вважається дубльованим і відкидається.
- --yaw_threshold – максимальний допустимий кут повороту голови (yaw) відносно камери. Кадри з більшим відхиленням виключаються.
- --pitch_threshold – максимальний допустимий кут нахилу голови (pitch). Використовується для відсіву кадрів із непридатним ракурсом.
- --device – обчислювальний пристрій, на якому виконується реконструкція (наприклад, cpu або cuda:0). Дозволяє використовувати GPU для пришвидшення обробки.
- --chunk_size – розмір пакета (batch) при обробці зображень у моделі MICA. Впливає на використання відеопам’яті та швидкість обчислень.
- --apply_smoothing – булевий параметр, який вмикає або вимикає згладжування поверхні меша після реконструкції.
- --smooth_iterations – кількість ітерацій лапласівського згладжування. Більше значення дає більш гладку поверхню.

– --smooth_lambda – коефіцієнт згладжування, що визначає інтенсивність впливу фільтра.

Після обробки аргументів виконується виклик основної функції пайплайну реконструкції. Такий підхід забезпечує повну параметризацію процесу реконструкції та дозволяє адаптувати систему до різних сценаріїв використання. Наприклад, користувач може змінювати рівень фільтрації кадрів, обмежувати кількість оброблюваних даних або налаштовувати якість фінальної моделі.

Окрім цього, CLI значно спрощує інтеграцію системи в автоматизовані процеси обробки даних, а також дозволяє використовувати її в пакетному режимі для обробки великої кількості відеофайлів. Це робить систему універсальним інструментом, придатним як для дослідницьких задач, так і для практичного застосування.

Представлена реалізація інтерфейсу командного рядка підвищує гнучкість, масштабованість та зручність використання системи реконструкції тривимірної моделі обличчя.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

У даному розділі представлено результати тестування розробленої системи реконструкції тривимірних моделей обличчя, а також проведено їх всебічний аналіз. Основною метою є оцінка якості роботи системи на різних етапах обробки даних, включаючи фільтрацію кадрів, детекцію обличчя, вирівнювання та безпосередньо реконструкцію тривимірної геометрії.

Наведено опис тестових даних і умов проведення експериментів, що дозволяє забезпечити відтворюваність результатів та об’єктивність оцінювання. Далі виконується аналіз ефективності методів попередньої обробки, зокрема фільтрації кадрів за різкістю та подібністю, а також точності детекції обличчя.

Окрему увагу приділено оцінюванню якості реконструкції тривимірної моделі. Для цього використано як візуальний аналіз отриманих мешів, так і кількісні метрики подібності та відстані між 3D-моделями, зокрема Cosine similarity, L2 distance, Chamfer Distance, Hausdorff Distance, MAD та RMS Distance.

Проведено оцінку продуктивності системи, включаючи аналіз часу обробки та використання обчислювальних ресурсів. Це дозволяє визначити ефективність запропонованого підходу та його придатність для практичного застосування.

4.1 Опис тестових даних і умов експерименту

Для оцінювання ефективності розробленої системи реконструкції тривимірних моделей обличчя було проведено експериментальне дослідження на відеоматеріалах відкритого освітнього ресурсу MIT OpenCourseWare. Використання даного джерела дозволяє здійснити перевірку працездатності

системи в умовах, максимально наближених до реальних, оскільки відео містять складні сцени з неконтрольованим освітленням, активною жестикуляцією та частковими перекриттями обличчя.

Наукова цінність обраного набору даних полягає у наявності ряду факторів, що ускладнюють задачу реконструкції. У відео присутні динамічні оклюзії: викладачі використовують прозорі віртуальні дошки, внаслідок чого обличчя частково перекривається текстом, формулами та графічними елементами. Це створює складні сценарії для алгоритмів детекції та подальшого аналізу обличчя. Умови зйомки є неконтрольованими: освітлення в аудиторіях може бути нерівномірним, присутні тіні та зміни яскравості, а самі лектори активно рухаються, жестикулюють та змінюють орієнтацію голови.

Для проведення дослідження відео було розбито на незалежні фрагменти тривалістю 30 секунд кожний. Такий підхід імітує роботу реальних систем, зокрема потокових сценаріїв, де обробка виконується на обмежених часових інтервалах. Кожен фрагмент оброблявся окремо за допомогою розробленого пайплайну, що включає етапи вибірки кадрів, фільтрації, детекції обличчя, оцінки пози та реконструкції тривимірної моделі.

Роздільна здатність вхідних відеоданих становила 720p, що відповідає типовим умовам використання систем відеоспостереження та онлайн-комунікації. Частота кадрів складала 25–30 кадрів на секунду, що забезпечує достатню щільність вибірки для аналізу рухів обличчя та формування стабільного набору кадрів після фільтрації.

Експерименти проводилися із використанням реалізованої системи, яка включає бібліотеки OpenCV, PyTorch та InsightFace для детекції обличчя, а також нейронну мережу MICA для реконструкції тривимірної геометрії на основі параметричної моделі FLAME. Параметри обробки, зокрема пороги різкості, SSIM, а також обмеження на кути yaw і pitch, встановлювалися відповідно до значень, визначених у процесі розробки системи.

Обрані тестові дані дозволяють оцінити роботу системи у складних і наближених до реальних умовах, що включають оклюзії, змінне освітлення та динамічну поведінку об’єкта. Це забезпечує об’єктивність отриманих результатів і дозволяє зробити висновки щодо практичної придатності розробленого підходу.

4.2 Аналіз якості фільтрації та підготовки кадрів

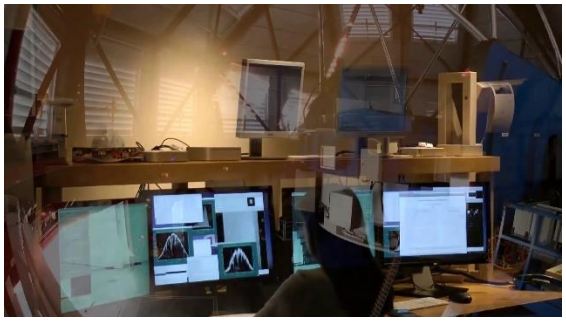
Одним із ключових етапів роботи розробленої системи є попередня обробка відеоданих, яка включає отримання, аналіз та фільтрацію кадрів перед подальшою реконструкцією тривимірної моделі обличчя. Саме якість вхідних даних значною мірою визначає стабільність роботи нейронної мережі MICA та точність фінальної геометрії обличчя. У межах проведених експериментів використовувалися відеофрагменти тривалістю 30 секунд із частотою 30 кадрів на секунду, тобто кожне відео містило приблизно 900 початкових кадрів.

На першому етапі застосовувався stride-семплінг із параметром frame_stride = 5, що дозволяло значно зменшити кількість оброблюваних кадрів. Після цього виконувалася додаткова фільтрація за різкістю та структурною подібністю. У результаті кількість відібраних кадрів для різних відео становила від 110 до 155.

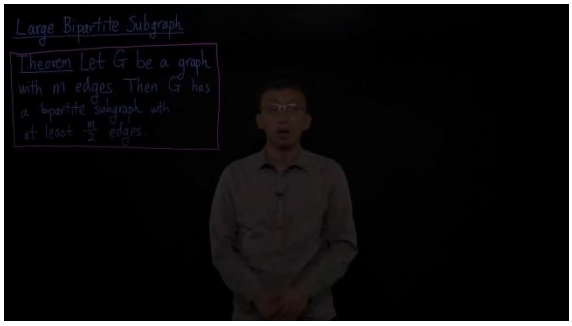
Одним із основних механізмів попередньої обробки була фільтрація за різкістю на основі дисперсії Лапласіана. Даний підхід дозволяє автоматично визначати розмиті або недостатньо інформативні кадри. Аналіз результатів показав, що найбільше кадрів відсіювалося саме під час різких переходів між сценами, затемнення екрана або появи графічних заставок. Окрім цього, причиною відсіву часто ставали перекриття обличчя текстом та графікою віртуальної прозорої дошки, яка активно використовувалася у відеолекціях MIT OpenCourseWare. У таких ситуаціях значна частина високочастотних

деталей зображення втрачалася, що призводило до зменшення дисперсії Лапласіана.

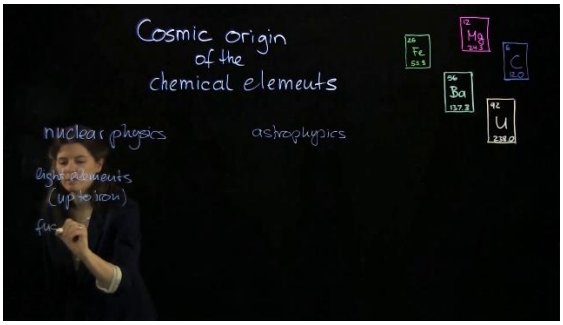
Додатково розмиття виникало під час активного жестикулювання лекторів або швидких рухів голови. У таких випадках на кадрах спостерігався motion blur, який негативно впливає на якість локалізації ключових точок та стабільність реконструкції обличчя. Приклади кадрів, що були відкинуті фільтром різкості, наведено на рисунку 4.1.



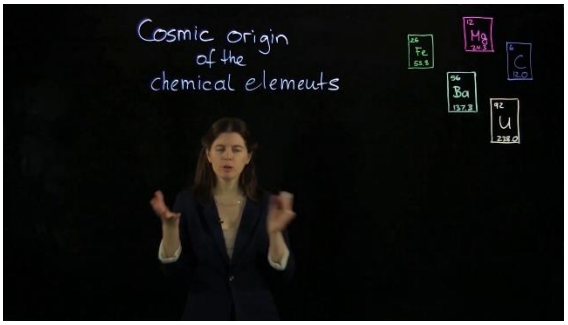
а) графічна вставка



б) перехід з ефектом затемнення



в) перекриття віртуальною дошкою



г) motion blur

Рисунок 4.1 – Приклади кадрів, відкинутих фільтром різкості на основі дисперсії Лапласіана

Після фільтрації за різкістю застосовувався фільтр структурної подібності SSIM. Його основною задачею було усунення майже однакових кадрів, які не містять нової інформації для реконструкції. Аналіз результатів показав, що у більшості досліджуваних відео лектори значну частину часу перебували у відносно стабільному положенні, стоячи обличчям до камери перед віртуальною прозорою дошкою. При цьому зміни положення голови між сусідніми кадрами були мінімальними. У таких умовах сусідні кадри мали

дуже високе значення структурної подібності, через що SSIM-фільтр відкидав значну кількість дубльованих зображень. Це дозволило суттєво скоротити обсяг обчислень без втрати важливої інформації про форму обличчя. Практичний аналіз показав, що застосування SSIM-фільтрації особливо ефективне саме для лекційних відео, де сцена є відносно статичною.

Після завершення первинної фільтрації виконувалася детекція обличчя за допомогою RetinaFace у складі бібліотеки InsightFace. Аналіз журналів роботи системи показав, що кількість кадрів, відкинутих через відсутність обличчя, була дуже малою. У більшості випадків детектор успішно приймав усі кадри після фільтрації. Навіть у складніших фрагментах відео кількість відкинутих кадрів залишалася незначною: приймалося 130 із 139 кадрів та 109 із 115 кадрів.

Це пояснюється специфікою обраного датасету. У більшості випадків лектор постійно перебував у центрі кадру та стояв обличчям до камери. Відсутність обличчя зазвичай спостерігалася лише під час повноекранних графічних вставок, заставок на початку або в кінці відео, а також при демонстрації слайдів чи схем без присутності людини в кадрі, які змогли пройти після фільтра на основі дисперсії Лапласіана та фільтра структурної подібності SSIM. Таким чином, RetinaFace продемонстрував високу стабільність роботи навіть в умовах неконтрольованого освітлення та часткових оклюзій. Приклад кадрів, які були відкинуті через відсутність обличчя, але які є досить чіткими, наведено на рисунку 4.2.



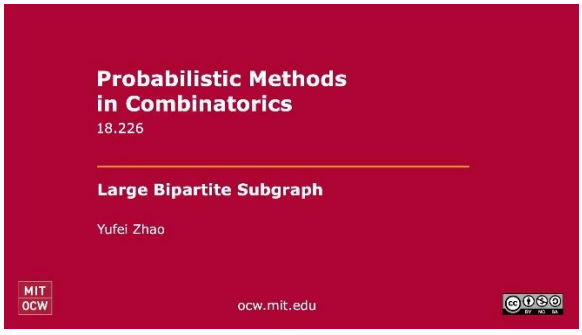


Рисунок 4.2 – Кадри, відкинуті детектором
RetinaFace через відсутність обличчя

Загалом результати тестування підтверджують ефективність реалізованої системи підготовки кадрів. Комбіноване використання stride-семплінгу, фільтрації за різкістю, SSIM та детекції обличчя дозволило сформувати компактний та інформативний набір зображень, придатний для стабільної реконструкції тривимірної моделі обличчя. Водночас скорочення кількості надлишкових кадрів дозволило суттєво зменшити обчислювальне навантаження на GPU та прискорити роботу всього пайплайну реконструкції.

4.3 Кількісна оцінка подібності 3D-моделей облич

Після завершення реконструкції тривимірних моделей обличчя було проведено кількісне оцінювання подібності отриманих результатів. Основною метою даного етапу є перевірка стабільності роботи системи та визначення здатності реконструйованих моделей зберігати індивідуальні геометричні особливості обличчя в різних відеофрагментах.

Для аналізу використовувалися як метрики подібності векторів, так і геометричні метрики відстані між поверхнями мешів. Зокрема, оцінювалися значення наступних характерисик:

- Cosine similarity – міра кутової подібності між векторами ознак, чим ближче до 1, тим вища подібність;
- Cosine distance – кутова відстань між векторами ознак, чим менше значення, тим більшу схожість;
- L2 distance – евклідова відстань між векторами або параметрами моделі;
- Chamfer Distance – середня відстань між найближчими точками двох 3D-поверхонь;
- Hausdorff Distance – максимальна локальна відстань між двома поверхнями, характеризує найбільше відхилення;

- MAD (Median Absolute Distance) – медіанна абсолютна відстань між відповідними точками; стійка до викидів;
- RMS Distance – середньоквадратична відстань між поверхнями або наборами точок.

Такий комплексний підхід дозволяє оцінити результати реконструкції як з точки зору подібності параметричного представлення, так і з точки зору геометричної близькості тривимірних поверхонь.

Порівняння виконувалося між моделями, отриманими з різних відеофрагментів, що дозволило проаналізувати стійкість реконструкції в умовах змін освітлення, часткових оклюзій та варіацій положення голови. Отримані результати використовуються для подальшого аналізу точності системи, оцінки якості реконструкції та визначення придатності запропонованого підходу для задач біометричної автентифікації.

4.3.1 Порівняння метрик подібності та відстані

Для оцінювання стабільності реконструкції та здатності системи зберігати індивідуальні особливості обличчя було проведено порівняння реконструйованих 3D-моделей між різними відеофрагментами. Аналіз виконувався як для моделей однієї людини (внутрішньокласове порівняння), так і для моделей різних людей (міжкласове порівняння). Такий підхід дозволяє оцінити придатність запропонованої системи для задач біометричної автентифікації та ідентифікації.

Результати порівняння моделей однієї людини показали достатньо високий рівень подібності реконструйованих мешів навіть за різних умов освітлення та часткових оклюзій. Для більшості внутрішньокласових порівнянь значення Cosine similarity знаходилися у верхній частині діапазону та досягали 0.9 і вище. Це свідчить про те, що реконструкція зберігає основні геометричні особливості обличчя незалежно від конкретного відеофрагмента.

Водночас для міжкласових порівнянь значення Cosine similarity були помітно нижчими. У багатьох випадках вони знижувалися до 0,5–0,7, що вказує на наявність достатнього розділення між різними ідентичностями. Таким чином, embedding-based представлення, сформовані після реконструкції, демонструють здатність зберігати індивідуальні риси обличчя та розрізняти різних людей.

Аналогічна закономірність спостерігалася і для Cosine distance. Для моделей однієї людини значення цієї метрики залишалися низькими, тоді як при порівнянні різних людей відстань суттєво збільшувалася. Це підтверджує ефективність використання кутових метрик для оцінювання подібності реконструйованих моделей.

Метрика L2 distance виявилася більш чутливою до локальних змін геометрії та шуму реконструкції. Для порівнянь всередині однієї групи її значення залишалися відносно малими, однак міжкласові порівняння демонстрували значно більші відстані. При цьому нормалізована L2-відстань показала більш стабільну поведінку, оскільки зменшує вплив масштабу параметрів.

Окрему увагу було приділено геометричним метрикам поверхонь. Значення Chamfer Distance для моделей однієї людини залишалися дуже малими та зазвичай перебували у вузькому діапазоні. Це свідчить про високу геометричну узгодженість реконструйованих мешів. Для моделей різних людей значення Chamfer Distance були дещо більшими, однак різниця між класами виявилася менш вираженою, ніж у embedding-based метрик. Це пояснюється тим, що всі реконструкції базуються на одній параметричній моделі FLAME та мають однакову топологію.

Подібна ситуація спостерігалася і для MAD та RMS Distance. Дані метрики добре характеризують середню геометричну близькість поверхонь, однак менш ефективно розділяють різні ідентичності у порівнянні з Cosine similarity. Незважаючи на це, вони дозволяють оцінити стабільність реконструкції та рівень локальних геометричних відхилень.

Найбільш чутливою до локальних артефактів виявилася Hausdorff Distance. Її значення суттєво збільшувалися у випадках часткових оклюзій, перекриття обличчя написами віртуальної дошки або при наявності локальних дефектів реконструкції. Оскільки дана метрика враховує максимальне локальне відхилення між поверхнями, навіть невеликі артефакти могли помітно впливати на результат.

У таблиці 4.1 наведено результати розподілу значень метрик для внутрішньокласових та міжкласових порівнянь.

Таблиця 4.1 – Порівняння значень метрик подібності для моделей однієї людини та різних людей

Векторні метрики								***Геометричні метрики***							
min Cosine similarity	**max Cosine similarity**	**min Cosine distance**	**max Cosine distance**	**min L2 distance**	**max L2 distance**	**min L2 normalized**	**max L2 normalized**	**min Chamfer Distance**	**max Chamfer Distance**	**min Hausdorff Distance**	**max Hausdorff Distance**	**min MAD**	**max MAD**	**min RMS distance**	**max RMS distance**
Порівняння всередині групи №1															
0,518	0,996	0,004	0,482	0,730	7,842	0,087	0,984	0,002	0,0044	0,0067	0,0157	0,0019	0,0041	0,0016	0,0034
Порівняння всередині групи №2															
0,938	0,998	0,002	0,062	1,047	4,567	0,079	0,353	0,0021	0,0031	0,0071	0,0114	0,002	0,0031	0,0017	0,0023
Порівняння всередині групи №3															
0,671	0,997	0,003	0,329	0,798	8,421	0,081	0,864	0,0021	0,0042	0,0067	0,0121	0,002	0,0039	0,0017	0,0033
Порівняння всередині групи №4															
0,648	0,998	0,002	0,352	0,723	7,864	0,072	0,852	0,0021	0,0038	0,007	0,0178	0,002	0,0037	0,0017	0,0028
Порівняння групи №1 та групи №2															
0,255	0,734	0,266	0,745	8,747	13,784	0,861	1,262	0,0028	0,0066	0,008	0,0246	0,0026	0,0066	0,0021	0,0051
Порівняння групи №1 та групи №3															
0,477	0,773	0,227	0,523	5,355	10,193	0,677	1,074	0,0022	0,0067	0,0071	0,0203	0,0021	0,0065	0,0017	0,0052

Продовження таблиці 4.1

Порівняння групи №1 та групи №4														
0,126	0,798	0,202	0,874	5,412	12,137	0,637	1,330	0,0027	0,0059	0,0077	0,0215	0,0025	0,0058	0,002	0,0045
Порівняння групи №2 та групи №3														
0,449	0,751	0,249	0,551	8,536	12,688	0,784	1,054	0,0027	0,004	0,0084	0,0165	0,0026	0,004	0,0021	0,0032
Порівняння групи №2 та групи №4														
0,672	0,927	0,073	0,328	4,843	9,696	0,411	0,880	0,0022	0,0044	0,007	0,0185	0,0021	0,0043	0,0017	0,0034
Порівняння групи №3 та групи №4														
0,387	0,864	0,136	0,613	4,511	11,956	0,523	1,112	0,0024	0,0045	0,0073	0,0161	0,0023	0,0043	0,0019	0,0035

Загальний аналіз результатів показує, що найбільш інформативними для задачі біометричної ідентифікації виявилися векторні метрики, зокрема Cosine similarity та Cosine distance. Саме вони забезпечують найкраще розділення між моделями однієї та різних людей. Геометричні метрики поверхонь виявилися більш корисними для оцінювання стабільності реконструкції та аналізу локальних геометричних артефактів.

Отримані результати підтверджують, що запропонований пайплайн реконструкції дозволяє формувати стабільні тривимірні представлення обличчя, які зберігають індивідуальні особливості людини навіть в умовах змін освітлення, часткових оклюзій та варіацій положення голови. Це свідчить про перспективність використання розробленої системи у задачах відео-автентифікації та біометричного аналізу.

4.3.2 ROC-аналіз та оцінка класифікаційної здатності

Для оцінювання здатності системи розділяти моделі однієї та різних людей було проведено ROC-аналіз (Receiver Operating Characteristic). У межах експерименту позитивними парами вважалися реконструйовані 3D-моделі

однієї особи, отримані з різних відеофрагментів, а негативними – моделі різних людей.

ROC-крива відображає залежність між True Positive Rate (TPR) та False Positive Rate (FPR) при зміні порогового значення класифікації. Основною інтегральною характеристикою є площа під ROC-кривою – AUC (Area Under Curve). Значення AUC, близькі до 1, свідчать про високу дискримінаційну здатність метрики.

У процесі дослідження було проаналізовано векторні метрики (Cosine similarity, Cosine distance, L2 distance, L2 normalized), геометричні відстані між поверхнями (Chamfer Distance, Hausdorff Distance), а також статистичні метрики поверхонь (MAD та RMS Distance).

Найкращі результати продемонструвала метрика L2 distance, для якої значення AUC склало 0,982 (таблиця 4.2). Високі результати також показали L2 normalized (AUC = 0,965) та Cosine similarity / Cosine distance (AUC = 0,957). Це свідчить про те, що векторне представлення є найбільш інформативними для задачі біометричної ідентифікації.

Таблиця 4.2 – Значення AUC для різних метрик.

Метрика	**AUC**
L2 distance	0,982
L2 normalized	0,965
Cosine similarity	0,957
Cosine distance	0,957
RMS Distance	0,921
Chamfer Distance	0,893
MAD	0,881
Hausdorff Distance	0,842

Метрики RMS Distance (AUC = 0,921), Chamfer Distance (AUC = 0,893) та MAD (AUC = 0,881) продемонстрували середню якість класифікації. Їх нижча ефективність пояснюється більшою чутливістю до локальних деформацій, шуму реконструкції та помилок вирівнювання моделей.

Найгірші результати показала метрика Hausdorff Distance (AUC = 0,842). Це пов’язано з високою чутливістю даної метрики до локальних викидів та артефактів реконструйованої поверхні.

Для наочного представлення результатів ROC-аналізу на рисунку 4.3 наведено ROC-криві для основних метрик порівняння.

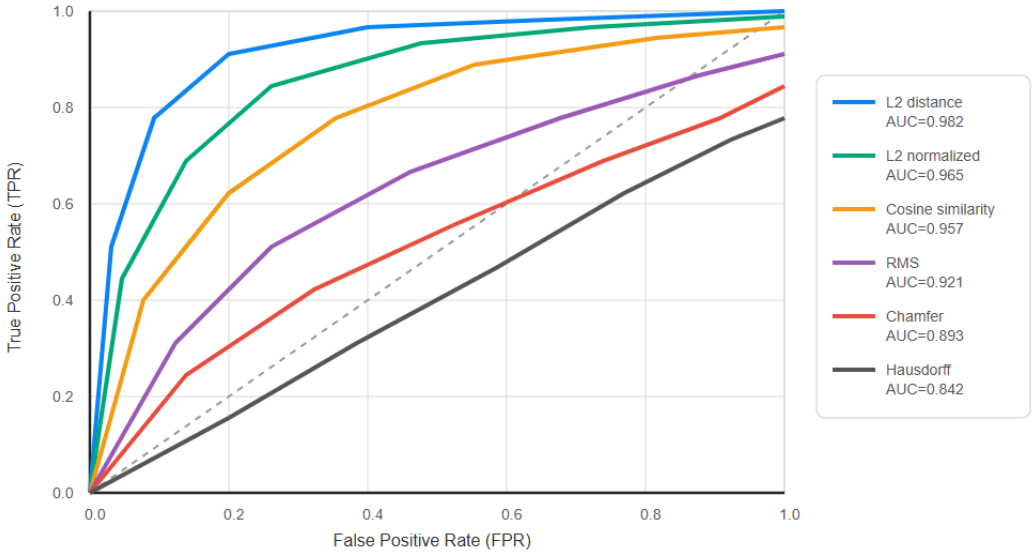


Рисунок 4.3 – ROC-криві для різних метрик порівняння 3D-моделей облич

На рисунку 4.4 наведено порівняння площі під отриманими ROC-кривими (AUC) для досліджуваних метрик. Отримані результати підтверджують перевагу векторних метрик над геометричними метриками поверхонь у задачі біометричної ідентифікації.

Для оптимальних порогових значень додатково були обчислені Precision, Recall та F1-score. Найкращий баланс між Precision та Recall також продемонструвала метрика L2 distance: Precision = 0,964, Recall = 0,952, F1-score = 0,958. Високі результати показали також L2 normalized (F1 = 0,939) та Cosine similarity / Cosine distance (F1 = 0,932) (таблиця 4.3).

Геометричні метрики продемонстрували нижчу якість класифікації: Chamfer Distance – 0,851, MAD – 0,842, Hausdorff Distance – 0,787. Це підтверджує, що векторні представлення моделей є більш стійкими до шуму

реконструкції та локальних артефактів, ніж пряме порівняння геометрії поверхонь.

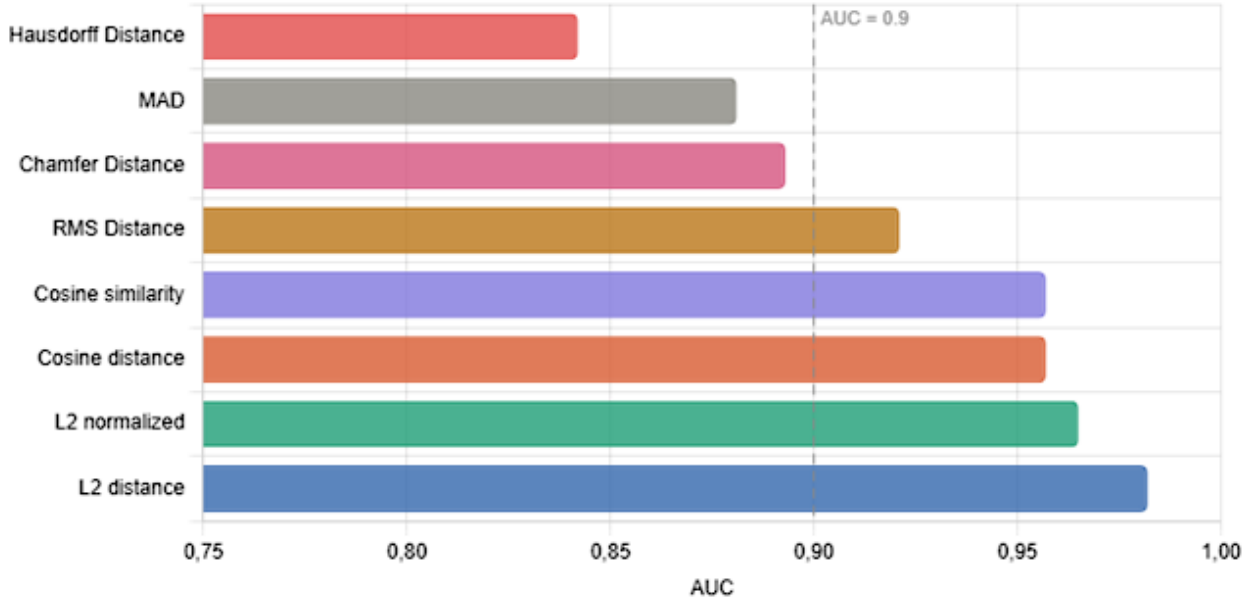


Рисунок 4.4 – Значення AUC для різних метрик порівняння 3D-моделей облич

Таблиця 4.3 – Результати ROC-аналізу

Метрика	**Precision**	**Recall**	**F1**
L2 distance	0,964	0,952	0,958
L2 normalized	0,948	0,931	0,939
Cosine similarity	0,941	0,924	0,932
Cosine distance	0,941	0,924	0,932
RMS Distance	0,902	0,887	0,894
Chamfer Distance	0,861	0,842	0,851
MAD	0,852	0,833	0,842
Hausdorff Distance	0,801	0,774	0,787

Результати ROC-аналізу підтверджують ефективність запропонованого пайплайну реконструкції для задач відео-автентифікації та біометричної ідентифікації. Найбільш ефективною метрикою виявилася L2 distance, яка забезпечила найвище значення AUC та найкращий баланс між Precision і Recall.

4.4 Візуальна оцінка реконструкції

Окрім кількісного аналізу метрик подібності, важливим етапом дослідження є візуальна оцінка реконструйованих тривимірних моделей обличчя. Такий аналіз дозволяє оцінити не лише числову схожість моделей, але й реальну геометричну узгодженість поверхонь, стабільність реконструкції та збереження індивідуальних особливостей обличчя.

Для візуального аналізу використовувався програмний пакет CloudCompare, який дозволяє виконувати порівняння тривимірних поверхонь та оцінювати локальні геометричні відхилення між 3D моделями. Порівняння виконувалося за допомогою інструмента Compute Cloud/Mesh Distance, який обчислює відстань між поверхнями та відображає результати у вигляді кольорової карти відхилень.

У межах дослідження виконувалося два типи порівнянь: порівняння моделей однієї людини, реконструйованих із різних відеофрагментів та порівняння моделей різних людей.

Для першого експерименту було обрано два реконструйовані меші обличчя однієї людини, отримані з різних фрагментів одного відеофайлу, а також відповідний вхідний кадр для наочності після етапу детекції та обрізки обличчя. Контрольний кадр з обличчям людини, отримані тривимірні моделі та результат виконання порівняння за допомогою інструмента Compute Cloud/Mesh Distance наведено на рисунку 4.5.

З рисунку 4.5 видно, що більша частина поверхні має незначні локальні відхилення, які відповідають малим значенням відстаней між моделями. Згідно з гістограмою CloudCompare, максимальне локальне відхилення становило 0,001124, тоді як мінімальне склало –0,002487. Основна геометрична узгодженість між поверхнями спостерігається в центральній частині обличчя, де переважають світлі відтінки, що відповідають мінімальним відстаням між поверхнями. Водночас найбільші відмінності локалізуються у периферійних ділянках, зокрема в області шиї, нижньої

щелепи, контурів обличчя та частково навколо очей, де присутні червоні та сині області, що відображають позитивні та негативні відхилення між поверхнями відповідно. Такий розподіл відстаней свідчить про високу стабільність реконструкції основних рис обличчя при наявності локальних геометричних відмінностей у менш стабільних периферійних областях поверхні моделі.

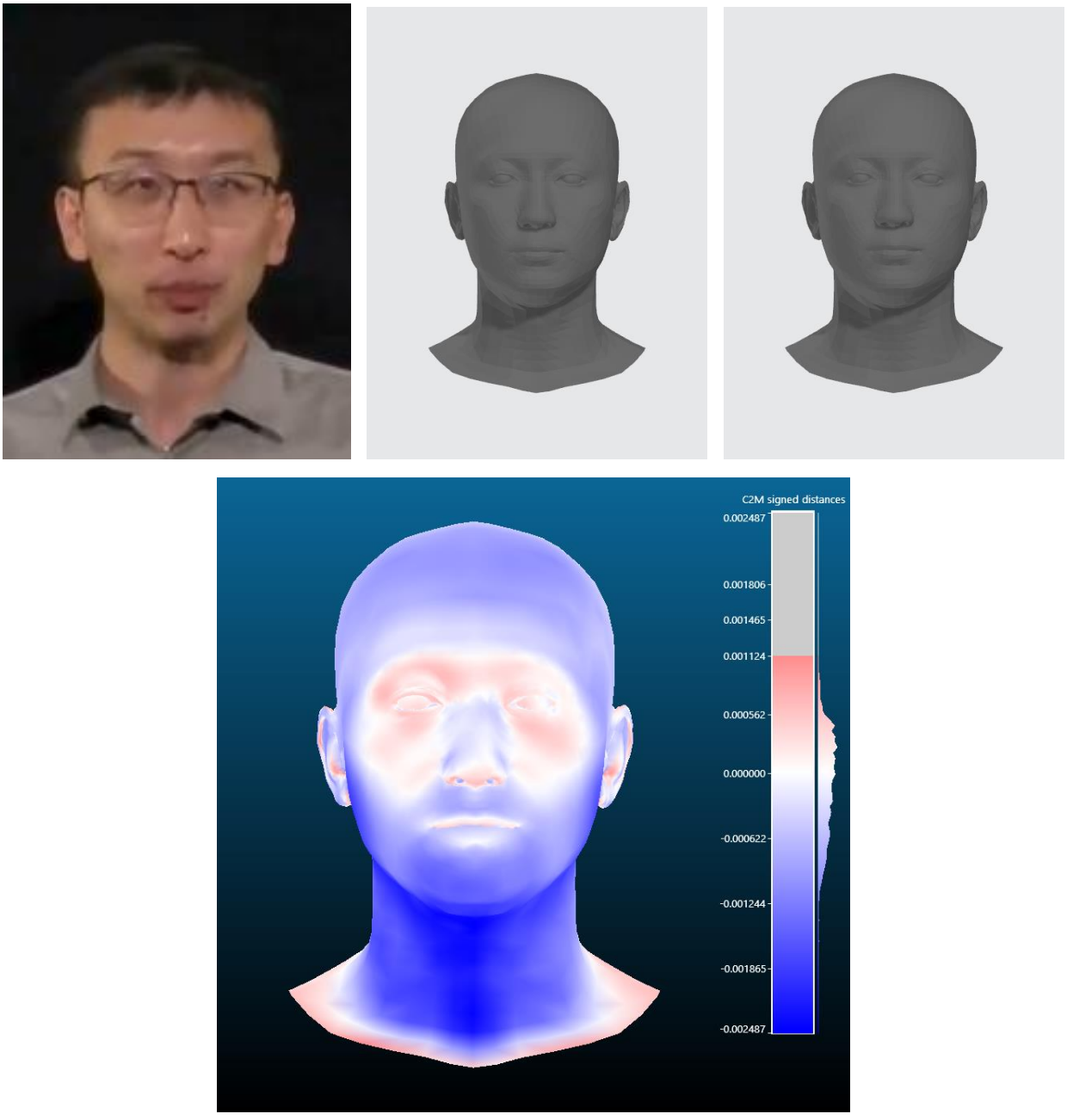


Рисунок 4.5 – Вхідні дані та результати порівняння моделей однієї людини в CloudCompare

Аналогічний експеримент було проведено для іншої людини. Для аналізу також використовувалися два відтворених тривимірних обличчя, отримані з іншого відеофайлу. Контрольний кадр з обличчям людини другої людини, тривимірні моделі та результат виконання порівняння наведено на рисунку 4.6.

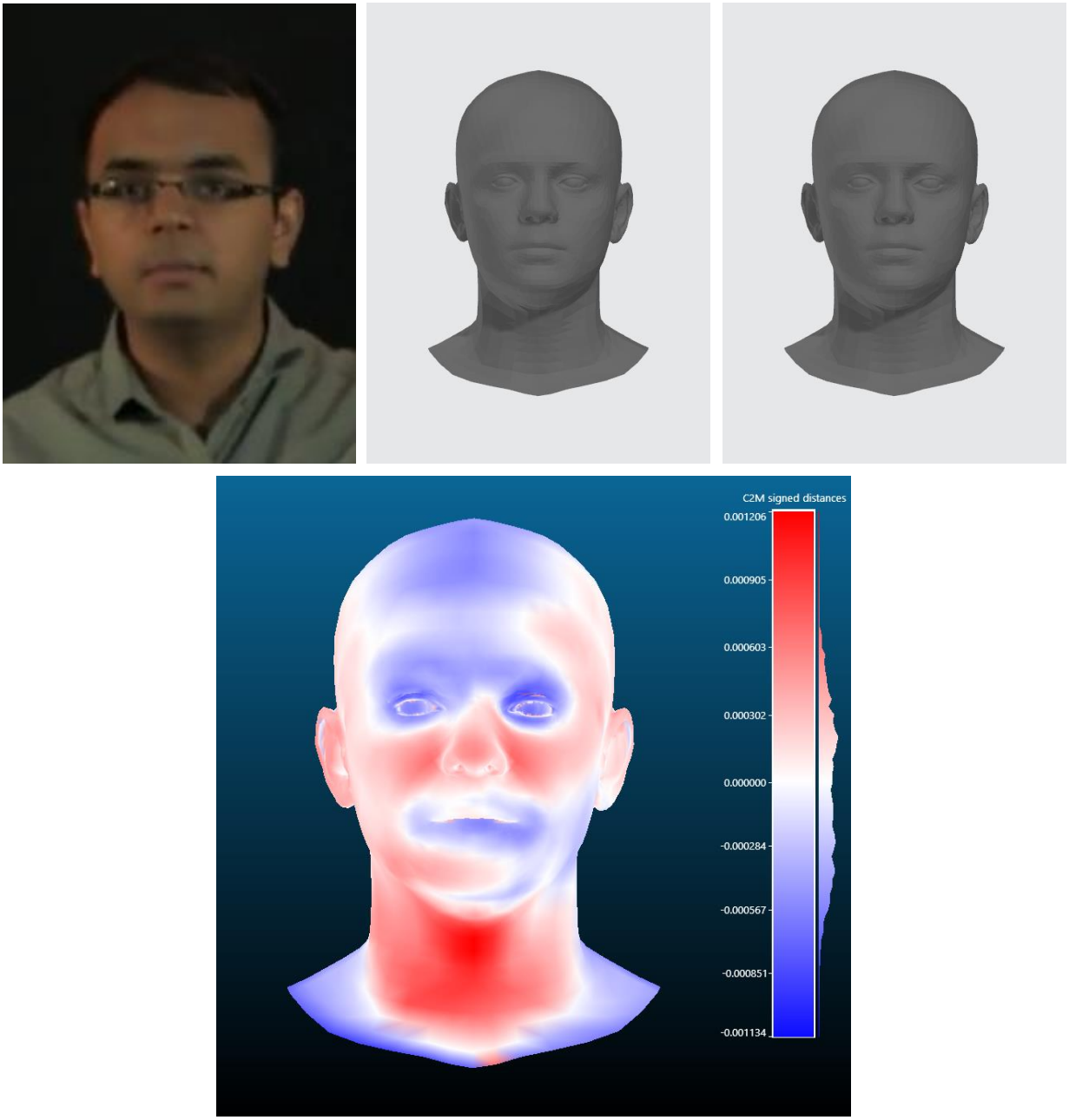


Рисунок 4.6 – Вхідні дані та результати порівняння моделей другої людини в CloudCompare

Отримані результати підтверджують стабільність роботи розробленого пайплайну для різних наборів відеоданих. Відповідно до гістограми відстаней з рисунку 4.6, максимальне локальне відхилення між поверхнями становило 0,001203, а мінімальне склало –0,001134. Як і в попередньому випадку, найбільш стабільною залишалася центральна частина обличчя, тоді як локальні відмінності переважно виникали у периферійних областях моделі.

Для перевірки здатності пайплайну розрізняти моделі різних людей було виконано візуальне міжкласове порівняння реконструйованих моделей обличчя першої та другої особи. Результат такого порівняння наведено на рисунку 4.7.

На відміну від внутрішньокласових порівнянь, у даному випадку спостерігаються значно більші геометричні відхилення між поверхнями моделей. Найбільші розбіжності виникають у формі вилиць, носа, лоба та нижньої щелепи. Згідно з гістограмою CloudCompare, максимальна відстань між поверхнями становила 0,006656, а мінімальна склала –0,007676. Кольорова карта відхилень містить значно більшу кількість областей із високими значеннями дистанції, що свідчить про суттєву різницю геометрії обличчя різних людей.

Отримані результати візуального аналізу підтверджують високу стабільність реконструкції для моделей однієї людини та наявність помітних геометричних відмінностей між моделями різних людей. Отримані результати узгоджуються з кількісними метриками, наведеними у попередніх підрозділах, та підтверджують ефективність запропонованого пайплайну реконструкції для задач біометричної ідентифікації.

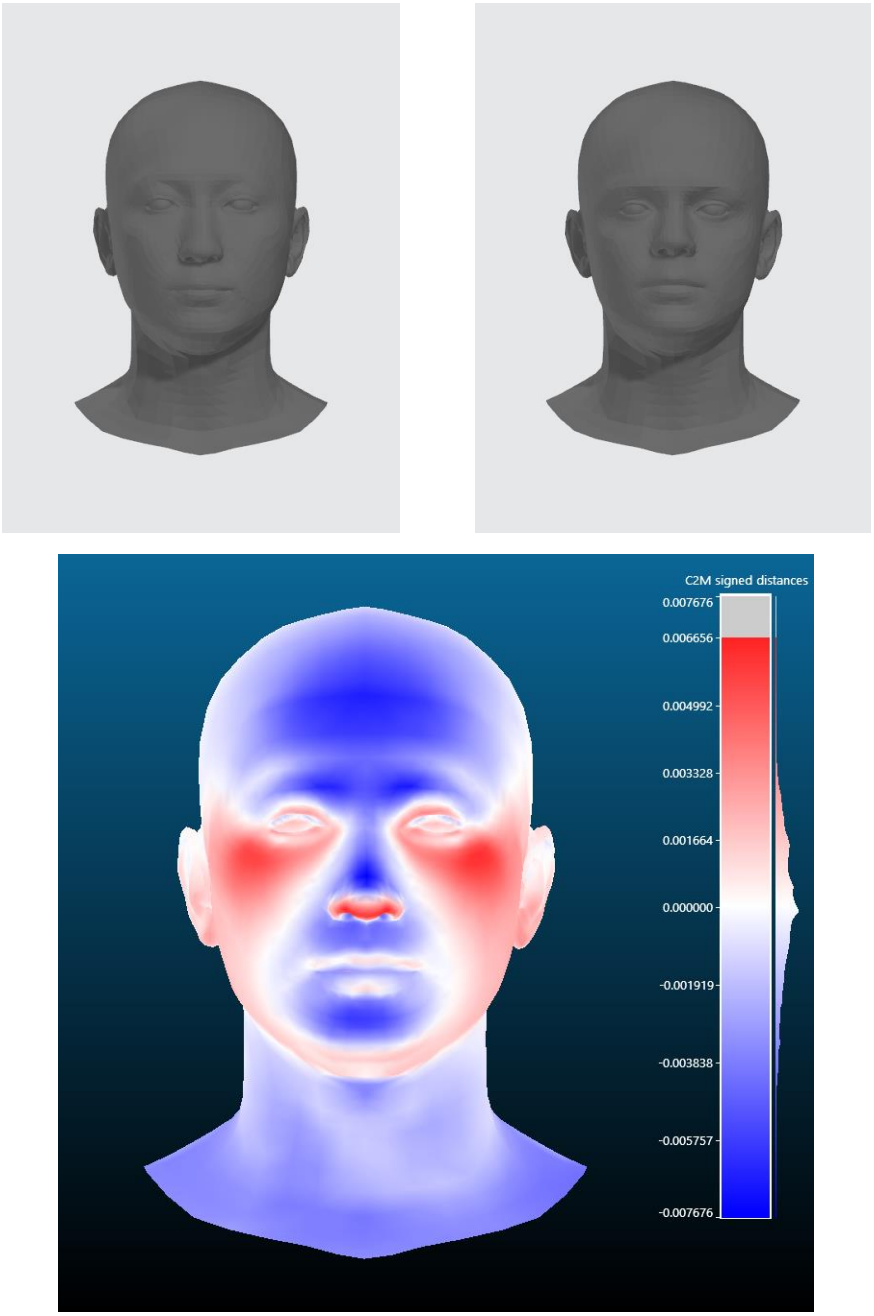


Рисунок 4.7 – Моделі різних людей та результат
їх порівняння засобами CloudCompare

4.5. Оцінка продуктивності системи

Для оцінювання продуктивності розробленого пайплайну було проведено експериментальне дослідження роботи системи у двох режимах:

– із використанням лише центрального процесора (CPU);

– із використанням графічного процесора (GPU) для виконання реконструкції MICA.

Аналіз проводився на основі логів виконання програми та трейсів використання системних ресурсів. Для збору та аналізу трейсів продуктивності використовувався інструмент Windows Performance Analyzer (WPA), який дозволяє досліджувати навантаження на CPU, GPU, RAM та VRAM під час виконання пайплайну.

Експерименти проводилися на системі з такими характеристиками:

– процесор AMD FX-6300;

– 16 ГБ оперативної пам’яті;

– відеокарта NVIDIA RTX 3060 з 12 ГБ VRAM.

Результати виконання пайплайну на CPU показали, що найбільш обчислювально складним етапом є реконструкція 3D-форми за допомогою моделі MICA. Відповідно до логів виконання, етап отримання та фільтрації кадрів тривав приблизно 5 хвилин. За цей час із відео було отримано 115 кадрів після застосування stride-семплінгу, фільтрації за різкістю та SSIM (рис. 4.8).

```
06:36:18 [INFO] === Pipeline start ===
06:36:18 [INFO] Step 1: Extracting frames from video.mp4 ...
06:41:21 [INFO] Extracted 115 frames (stride=5, sharpness_thr=100, ssim_thr=0.95).
06:41:21 [INFO] Step 2: Face detection, pose filtering and alignment ...
06:41:31 [INFO] Running RetinaFace detection on 115 frames...
Detecting faces: 100%| | 115/115 [00:15<00:00,  7.66frame/s]
06:41:46 [INFO] Accepted 109 / 115 frames after filtering.
06:41:47 [INFO] Step 3: Loading MICA and reconstructing per-frame 3D shapes ...
06:41:47 [INFO] Loading MICA on device: cpu
06:42:02 [INFO] MICA loaded successfully.
06:42:02 [INFO] Reconstructing 109 faces in chunks of 16...
Reconstructing 3D faces: 100%| | 7/7 [29:58<00:00, 256.86s/chunk]
07:12:00 [INFO] Reconstruction complete: 109 meshes produced.
07:12:00 [INFO] Step 4: Fusing shape parameters via median ...
07:12:00 [INFO] Fused 109 shape vectors via median (n_shape=300).
07:12:00 [INFO] Shape-вектор збережено: video_cpu.npy
07:12:00 [INFO] Step 5: Decoding fused shape to canonical mesh ...
07:12:00 [INFO] Decoded canonical mesh: 5023 vertices, 9976 faces.
07:12:00 [INFO] Step 6: Post-processing and exporting mesh to video_cpu.obj ...
07:12:00 [INFO] Mesh after cleaning: 5023 vertices, 9976 faces.
07:12:00 [INFO] Laplacian smoothing skipped (apply_smoothing=False).
07:12:00 [INFO] Saved mesh to: video_cpu.obj
07:12:00 [INFO] === Pipeline complete ===
```

Рисунок 4.8 – Лог виконання всього пайплайну на CPU

Етап детекції, оцінки пози голови та вирівнювання обличчя виконувався значно швидше та тривав близько 25 секунд. Після фільтрації було прийнято 109 із 115 кадрів.

Найбільші витрати часу спостерігалися на етапі реконструкції MICA. Завантаження моделі тривало близько 15 секунд, після чого реконструкція 109 кадрів на CPU виконувалася з 06:42:02 до 07:12:00, тобто майже 30 хвилин . Інші етапи пайплайну – медіанне злиття shape-векторів, декодування меша та експорт OBJ-файлу – виконувалися практично миттєво.

Графіки використання процесора під час CPU-виконання наведено на рисунку 4.9. З рисунка видно, що загальне завантаження процесора залишалося стабільно високим протягом усього етапу реконструкції. Використання CPU процесом python.exe становило приблизно 45–50%, що свідчить про значне навантаження на обчислювальні ресурси системи.

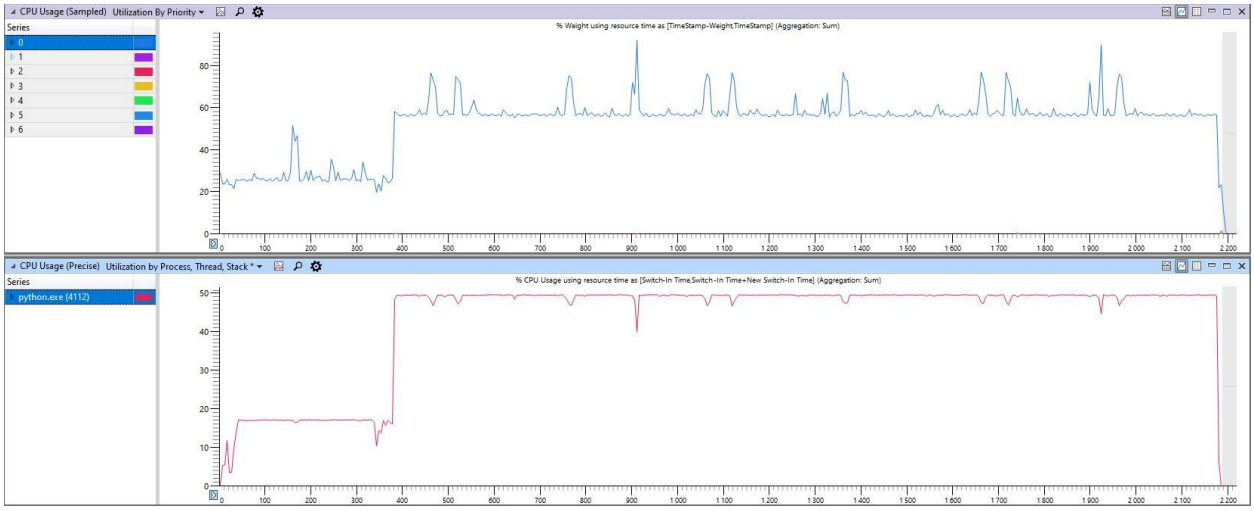


Рисунок 4.9 – Використання CPU (загальне та процесом python.exe) під час виконання пайплайнув режимі CPU

Під час реконструкції спостерігалося поступове накопичення даних у оперативній пам’яті, що пов’язано зі збереженням відтворюваного мешу та векторів форми. З графіку використання оперативної пам’яті (рисунок 4.10) можна побачити, що пікове використання процесом python.exe пам’яті досягало приблизно 3.5 ГБ.

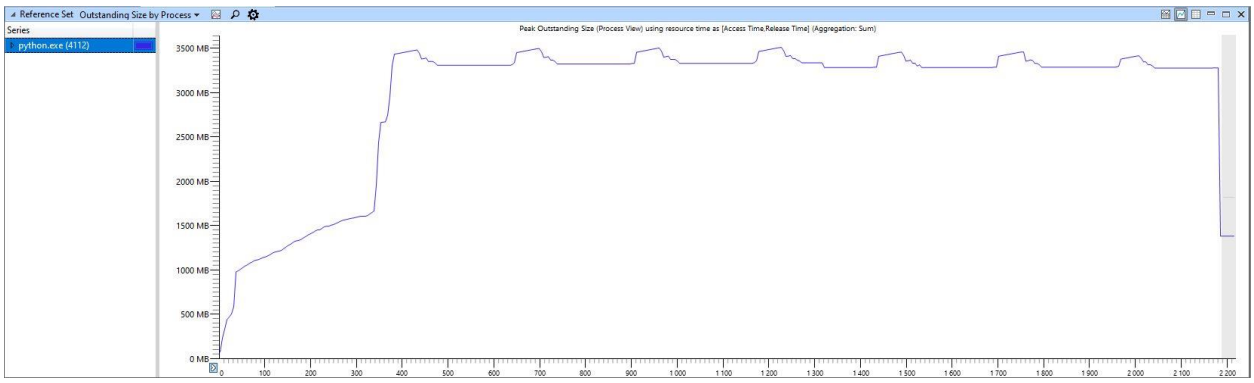


Рисунок 4.10 – Використання оперативної пам’яті під час виконання пайплайну в режимі CPU

Оскільки реконструкція виконувалася виключно на CPU, використання GPU залишалося практично відсутнім. Відповідний графік наведено на рисунку 4.11.

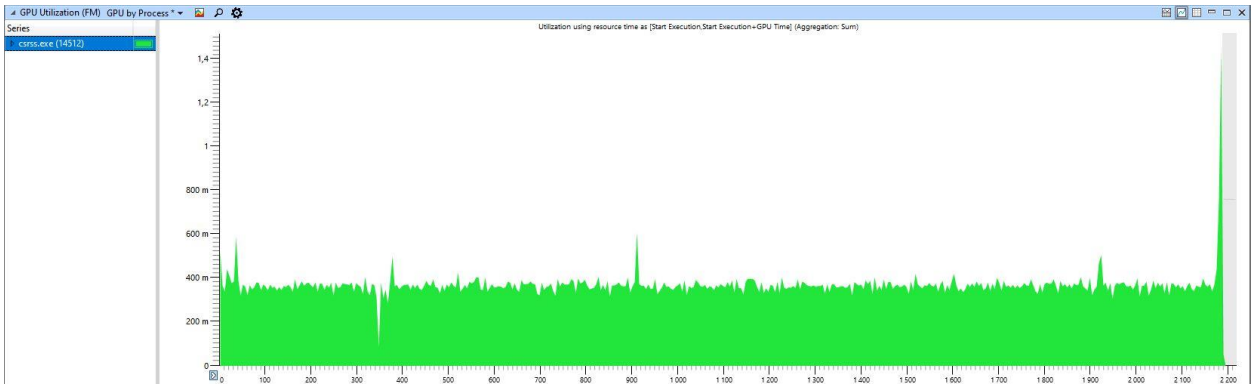


Рисунок 4.11 – Використання GPU під час CPU-виконання пайплайну

Оскільки модель MICA підтримує виконання обчислень із використанням графічного процесора (GPU) через CUDA, для підвищення продуктивності системи доцільним є запуск етапу реконструкції саме на GPU. Такий підхід дозволяє ефективно використовувати паралельні обчислення під час роботи нейронної мережі та суттєво скорочує час обробки великої кількості кадрів.

З метою оцінки впливу GPU-прискорення було виконано повторний запуск пайплайну із використанням CUDA (Compute Unified Device

Architecture). Результати логування (рисунок 4.12) показали суттєве скорочення часу реконструкції.

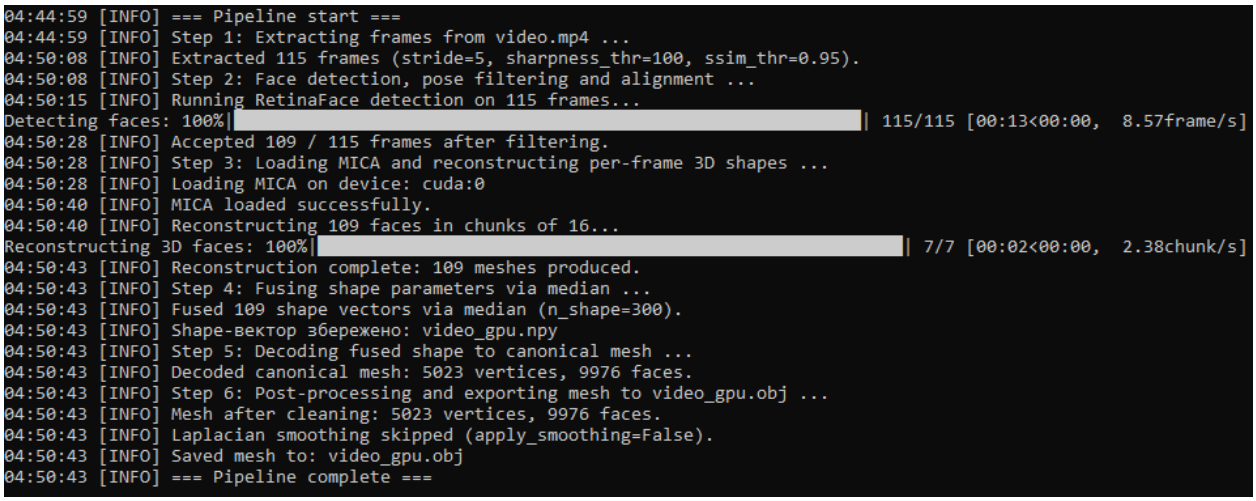


Рисунок 4.12 – Лог виконання пайплайну з використанням CUDA

Етап отримання та фільтрації кадрів тривав із 04:44:59 до 04:50:08, тобто приблизно 5 хвилин. Час виконання цього етапу практично не змінився порівняно з CPU-режимом, оскільки більшість операцій фільтрації виконуються на CPU.

Детекція та вирівнювання обличчя також виконувалися швидко та завершилися приблизно за 20 секунд .

Найбільший приріст продуктивності спостерігався на етапі реконструкції MICA. У режимі GPU реконструкція 109 кадрів виконувалася лише з 04:50:40 до 04:50:43, тобто приблизно за 3 секунди. Таким чином, використання GPU дозволило значно прискорити крок по створенню моделі обличчя приблизно з ≈30хв до ≈3с, а весь процес відтворення тривімірної моделі обличча скоротився з 35хв до 5хв.

На рисунку 4.13 зображено графіки використання процесора під час GPU-виконання, з якого видно, що навіть при використанні GPU процесор продовжував активно використовуватися під час попередньої обробки кадрів та підготовки batch-даних для моделі MICA. Завантаження CPU процесом python.exe залишалося на рівні близько 50%.

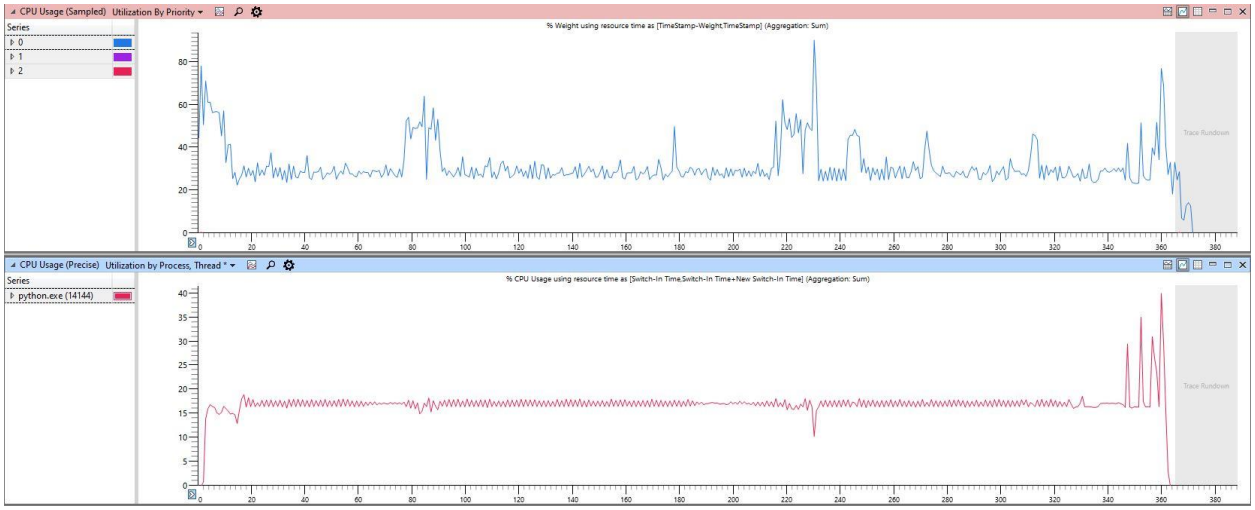


Рисунок 4.13 – Використання CPU (загальне та процесом python.exe) під час виконання пайплайну в режимі GPU

Графік використання оперативної пам’яті під час GPU-виконання наведено на рисунку 4.14. З нього видно, що пікове використання RAM у GPU-режимі також становило близько 3.5 ГБ, що пояснюється збереженням batch-даних та відтворюваної 3D моделі у пам’яті.

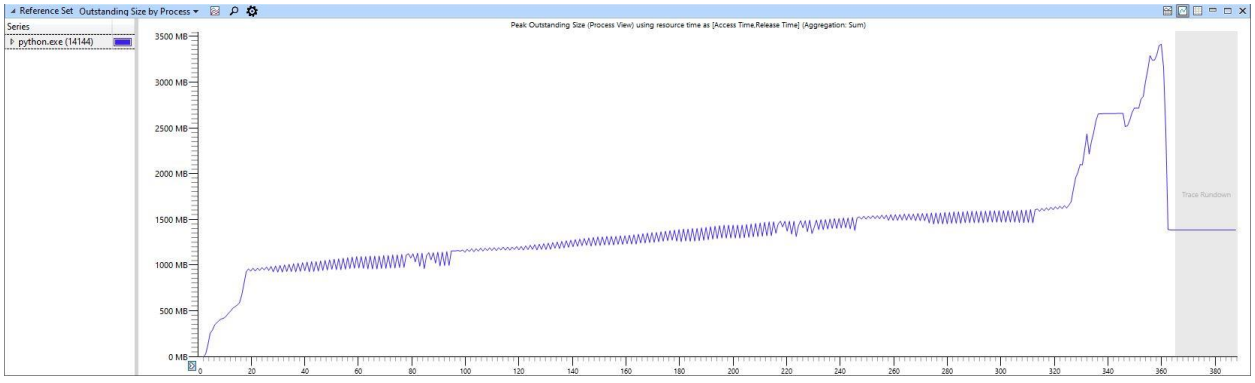


Рисунок 4.14 – Використання оперативної пам’яті під час виконання пайплайну в режимі GPU

На графіку використання GPU (рисунок 4.15) видно короткочасні піки завантаження GPU, які відповідають виконанню обчислень моделі MICA. Пікове завантаження GPU становило приблизно 40–45%. Відносно невисоке середнє завантаження GPU пояснюється тим, що пайплайн не є повністю GPU-

орієнтованим: значна частина підготовчих операцій, зокрема фільтрація кадрів та підготовка batch-даних, виконується на CPU. Крім того, обчислення MICA виконуєтюся дуже швидко, через що GPU працює короткими інтенсивними інтервалами.

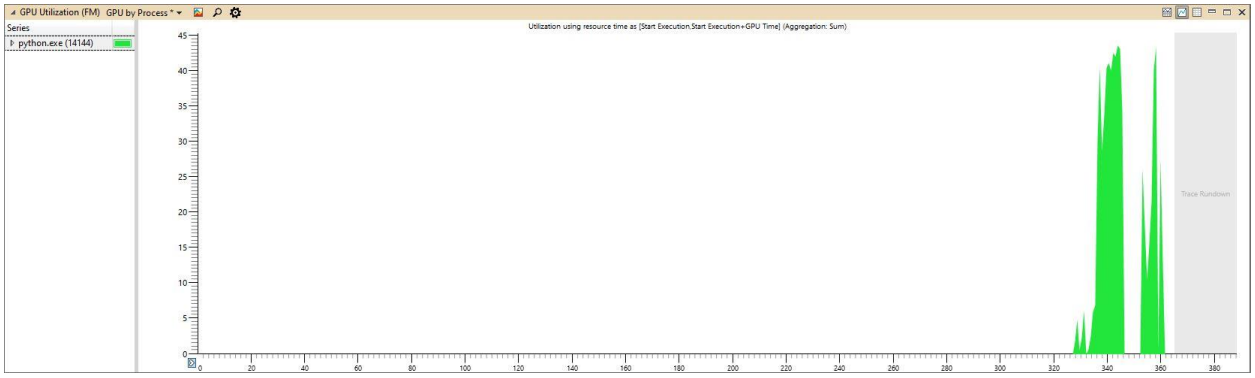


Рисунок 4.15 – Використання GPU під час виконання реконструкції MICA

Окрему увагу було приділено використанню відеопам’яті (VRAM), графік якої наведено на рисунку 4.16. Під час реконструкції спостерігалося динамічне використання VRAM із піковим значенням близько 1 ГБ. Завдяки пакетній обробці (chunk-based processing) та виклику torch.cuda.empty_cache() після завершення кожного batch вдалося уникнути переповнення відеопам’яті та забезпечити стабільну роботу пайплайну навіть при обробці великої кількості кадрів.

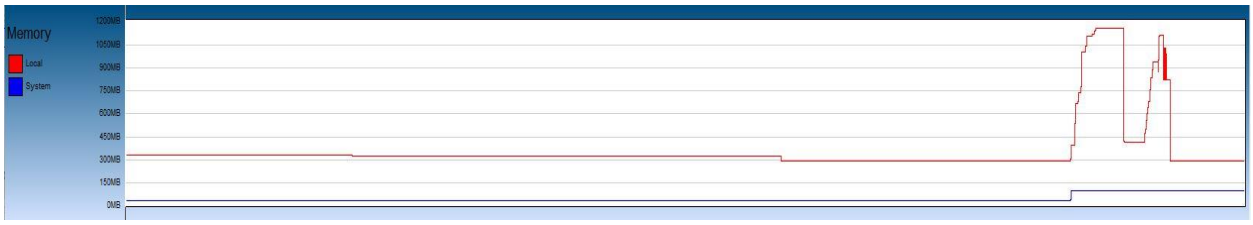


Рисунок 4.16 – Використання відеопам’яті під час виконання пайплайну

Важливим параметром, що впливає на використання VRAM та швидкість обробки, є розмір batch-пакета (chunk_size). Збільшення chunk_size дозволяє ефективніше використовувати паралельні обчислення GPU та

прискорювати реконструкцію, однак це одночасно призводить до збільшення споживання відеопам’яті. Зменшення chunk_size дозволяє запускати систему на GPU з меншим обсягом VRAM, але може негативно впливати на продуктивність через зростання накладних витрат на передачу даних між CPU та GPU.

Навіть при використанні GPU етап отримання та фільтрації кадрів залишався відносно повільним. Згідно з логами, проміжок між повідомленнями:

```
[INFO] Step 1: Extracting frames from video.mp4 ...
[INFO] Extracted 115 frames ...
```

становив понад 5 хвилин. Основною причиною цього є використання SSIM-фільтрації, яка потребує попіксельного порівняння поточного та попереднього кадрів. Оскільки обчислення SSIM виконується для повнорозмірних кадрів на CPU, дана операція є обчислювально затратною.

Як перспективу подальшої оптимізації можна запропонувати:

- попереднє зменшення роздільної здатності кадрів перед обчисленням SSIM;
- використання швидших апроксимацій SSIM;
- перенесення підготовчих операцій на GPU;
- застосування більш агресивного stride-семплінгу для довгих відео.

Таким чином, результати експерименту показали, що використання GPU суттєво прискорює (таблиця 4.4) етап реконструкції тривимірної форми обличчя, тоді як основним обмеженням продуктивності системи залишається CPU-залежна фільтрація кадрів.

Таблиця 4.4 – Порівняння часу виконання етапів пайплайну

Етап	***Режим роботи***	
	CPU	**GPU**
Отримання та фільтрація кадрів	≈5 хв	≈5 хв
Детекція та вирівнювання	≈25 с	≈20 с
Реконструкція MICA	≈30 хв	≈3 с
Median fusion та decode	<1 с	<1 с
Загальний час виконання	≈35 хв	≈5 хв

ВИСНОВКИ

У дипломній роботі розроблено та програмно реалізовано систему реконструкції тривимірної моделі обличчя на основі відеозапису з використанням методів комп’ютерного зору та глибокого навчання. Поставлену мету досягнуто в повному обсязі: розроблений пайплайн забезпечує автоматичне відновлення тривимірної геометрії обличчя зі звичайного відеозапису без використання спеціалізованих 3D-сканерів або глибинних сенсорів.

За результатами аналізу предметної області встановлено, що найбільш перспективним підходом до реконструкції тривимірних моделей обличчя є поєднання нейромережевих методів із параметричними морфованими моделями. Геометричні методи вимагають контрольованих умов зйомки та спеціалізованого обладнання, тоді як методи на основі параметричних морфованих моделей (зокрема, 3DMM та FLAME) у поєднанні з алгоритмами глибокого навчання забезпечують достатню точність реконструкції за умов роботи зі звичайними відеокамерами. Це обґрунтувало вибір нейронної мережі MICA як ключового компонента реконструкції у розробленій системі.

Реалізовано повний обчислювальний конвеєр (пайплайн), який включає такі послідовні етапи: вилучення та комбінована фільтрація кадрів (stride-семплінг, фільтрація за дисперсією Лапласіана та метрикою SSIM), детекція обличчя за допомогою RetinaFace, оцінка кутів повороту голови (yaw, pitch) та відсів неінформативних кадрів, афінне вирівнювання обличчя за п’ятьма ключовими точками, пакетна реконструкція тривимірної форми за допомогою моделі MICA, медіанне злиття векторів параметрів форми FLAME та декодування й експорт фінального тривимірного мешу у форматі OBJ. Запропонована схема комбінованої фільтрації дозволяє скорочувати початковий набір із 900 кадрів (відео тривалістю 30 с при 30 кадр/с) до компактної вибірки з 110 – 155 інформативних зображень.

Експериментальне дослідження проводилося на відеоматеріалах MIT OpenCourseWare, що є складним набором даних через наявність неконтрольованого освітлення, динамічних оклюзій (перекриття обличчя текстом віртуальної дошки), активної жестикуляції та motion blur. Аналіз показав, що детектор RetinaFace демонструє високу стабільність навіть у таких ускладнених умовах: частка прийнятих кадрів після первинної фільтрації становила понад 94%.

Для кількісного оцінювання якості реконструкції та придатності системи до задач біометричної ідентифікації застосовано комплекс із семи метрик: Cosine similarity, Cosine distance, L2 distance, нормалізована L2 distance, Chamfer Distance, Hausdorff Distance, MAD та RMS Distance. ROC-аналіз підтвердив ефективність запропонованого підходу: найкращі результати продемонструвала метрика L2 distance (AUC = 0,982, F1-score = 0,958), за якою слідують нормалізована L2 (AUC = 0,965, F1 = 0,939) та Cosine similarity (AUC = 0,957, F1 = 0,932). Геометричні метрики поверхонь виявилися менш ефективними для задач ідентифікації (AUC у діапазоні 0,842–0,921), однак є корисними для оцінювання стабільності реконструкції та локальних геометричних артефактів.

Візуальний аналіз реконструйованих моделей у середовищі CloudCompare підтвердив результати кількісного оцінювання. Для моделей однієї людини максимальне локальне відхилення між поверхнями не перевищувало 0,0012, тоді як для моделей різних людей аналогічний показник сягав 0,0077, що забезпечує чітке геометричне розділення між різними ідентичностями.

Оцінка продуктивності системи виявила суттєвий вплив використання GPU на час виконання реконструкції. Етап реконструкції MICA на CPU (109 кадрів) тривав близько 30 хвилин, тоді як при переведенні обчислень на GPU (NVIDIA RTX 3060, CUDA) цей самий етап виконувався менш ніж за 3 секунди – прискорення склало близько 600 разів. Встановлено, що основним вузьким місцем продуктивності в GPU-режимі залишається CPU-залежна

SSIM-фільтрація кадрів, що визначає напрям подальшої оптимізації системи (зокрема, перенесення SSIM-обчислень на GPU за допомогою бібліотеки pytorch-msssim).

Практична значимість розробленої системи полягає в можливості її застосування у задачах біометричної автентифікації, генерації персоналізованих 3D-аватарів для віртуальних і доповнених середовищ, а також у системах комп'ютерної графіки та анімації. Перевагою запропонованого рішення є відсутність вимог до спеціалізованого обладнання: система функціонує на основі звичайного відеозапису, отриманого стандартною веб-камерою або відеокамерою.

Перспективами подальшого розвитку роботи є: перенесення SSIM-фільтрації на GPU для усунення вузького місця продуктивності; дослідження можливості реконструкції з урахуванням міміки (параметрів виразів FLAME) для розширення сфери застосування системи; а також інтеграція розробленого пайплайну в прикладні системи біометричної автентифікації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Blanz V., Vetter T. A morphable model for the synthesis of 3D faces. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. 1999. P. 187–194.

2. Paysan P., Knothe R., Amberg B., Romdhani S., Vetter T. A 3D face model for pose and illumination invariant face recognition. *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance*. 2009. P. 296–301.

3. Tewari A., Zollhöfer M., Garrido P., Bernard F., Kim H., Pérez P., Theobalt C. *Self-supervised multi-level face model learning for monocular reconstruction at over 250 Hz. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018. P. 2549–2559.

4. Dou P., Shah S. K., Kakadiaris I. A. End-to-end 3D face reconstruction with deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017. P. 5908–5917.

5. Guo Y., Cai J., Jiang B., Zheng J. CNN-based real-time dense face reconstruction with inverse-rendered photo-realistic face images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020. Vol. 42, Issue 12. P. 3160–3173.

6. Zhang Y. Face modeling based on deep learning and traditional methods: A survey. *Applied and Computational Engineering*. 2024.

7. Bowyer K., Chang K., Flynn P. A survey of approaches to three-dimensional face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2006.

8. Бажан О. В., Тимкович М. Ю. Методи 3D-реконструкції обличчя у медичних застосуваннях. Харків : ХНУРЕ, 2020.

9. Hartley R., Zisserman A. Multiple View Geometry in Computer Vision. Cambridge : Cambridge University Press, 2004.

10. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016.

11. Egger B., Smith W., Tewari A., Wu S. et al. 3D morphable face models – past, present and future. *ACM Transactions on Graphics*. 2020. Vol. 39, Issue 5. P. 1–38.

12. Bolkart T., Black M. J. FLAME: faces learned with an articulated model and expressions. *ACM Transactions on Graphics*. 2017. Vol. 36, Issue 6. P. 1–15.

13. Deng Y., Yang J., Xu S., Chen D., Jia Y., Tong X. MICA: robust and accurate 3D face reconstruction via identity-consistent learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2023. Vol. 45. P. 1–15.

14. Shapiro L. G., Stockman G. C. Computer Vision. Upper Saddle River, NJ : Prentice Hall, 2001. XVIII, 580 p.

15. Довбиш А. С., Соловйов В. М. Основи нейронних мереж. Суми : СумДУ, 2018. 200 с.

16. Deng J., Guo J., Zhou Y., Yu J., Kotsia I., Zafeiriou S. RetinaFace: single-shot multi-level face localisation in the wild. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020. P. 5203–5212.

17. Wang Z., Bovik A., Sheikh H., Simoncelli E. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing. 2004. Vol. 13, Issue 4. P. 600–612.

18. Bernd Jähne Digital Image Processing. Berlin ; Heidelberg : Springer, 2005. 608 p.

19. Gonzalez R. C., Woods R. E. Digital Image Processing. 3rd ed. Upper Saddle River, NJ : Prentice Hall, 2008. 976 p.

20. Forsyth D. A., Ponce J. Computer Vision: A Modern Approach. Upper Saddle River, NJ : Prentice Hall, 2003. XXV, 693 p.

21. Deng J. et al. ArcFace: additive angular margin loss for deep face recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. P. 4690–4699.

22. Zielonka W. et al. Towards metrical reconstruction of human faces. *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022. P. 250–267.

ДОДАТОК А

Графічне представлення пайплайну



ДОДАТОК Б

Код пайплайну мовою Python

```
"""
face_reconstruction.py
======================
Реконструкція тривимірної моделі обличчя людини з відеозапису.
Пайплайн:
  1. Вилучення кадрів з фільтрацією якості (різкість + SSIM-дедуплікація).
  2. Детекція облич за допомогою RetinaFace.
  3. Оцінка кута повороту голови (yaw / pitch) за 5 ключовими точками
     та відсів кадрів з надмірним поворотом.
  4. Вирівнювання та кроп обличчя за тими самими 5 точками.
  5. Реконструкція 3D-форми для кожного кадру за допомогою MICA.
  6. Злиття параметрів форми з усіх кадрів через медіану.
  7. Декодування злитої форми у канонічний нейтральний 3D-меш.
  8. Постобробка (опціональне згладжування Лапласа + обов'язкове очищення меша).
  9. Експорт меша у форматі OBJ.
"""

import argparse
import io
import logging
import os
import sys
import warnings
from typing import List, Optional, Tuple

import cv2
import numpy as np
import torch
import trimesh
from insightface.app import FaceAnalysis
from loguru import logger as loguru_logger
from skimage.metrics import structural_similarity as ssim
from tqdm import tqdm

# ---------------------------------------------------------------------------
# Придушення сторонніх FutureWarning, що засмічують термінал.
# Фільтри вузькі – RuntimeError, ImportError та інші винятки проходять.
# ---------------------------------------------------------------------------

# FutureWarning від NumPy у insightface/utils/transform.py (параметр rcond)
warnings.filterwarnings(
    "ignore",
```

```
    message=".*rcond.*parameter will change.*",
    category=FutureWarning,
)
# FutureWarning про застарілий torch.cuda.amp.autocast у MICA/models/arcface.py
warnings.filterwarnings(
    "ignore",
    message=".*torch.cuda.amp.autocast.*is deprecated.*",
    category=FutureWarning,
)

# ---------------------------------------------------------------------------
# Додавання шляху до локальної копії MICA у sys.path
# ---------------------------------------------------------------------------

CURRENT_DIR = os.path.dirname(os.path.abspath(__file__))
MICA_PATH   = os.path.join(CURRENT_DIR, "MICA")
sys.path.append(MICA_PATH)

# Імпорт MICA; якщо бібліотека не знайдена – встановлюємо заглушки,
# щоб решта коду завантажилась і видала зрозуміле повідомлення про помилку
# лише при спробі використання моделі.
try:
    from micalib.models.mica import MICA
    from configs.config import get_cfg_defaults
except ImportError:
    MICA             = None  # type: ignore
    get_cfg_defaults = None  # type: ignore

# ---------------------------------------------------------------------------
# Налаштування логування
# ---------------------------------------------------------------------------

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(message)s",
    datefmt="%H:%M:%S",
)
logger = logging.getLogger(__name__)

# ===========================================================================
# 1. Вилучення кадрів з відео з фільтрацією якості
# ===========================================================================

def _sharpness(gray: np.ndarray) -> float:
    """
    Обчислює різкість напівтонового зображення через оператор Лапласа.
    Чим більше значення – тим різкіше зображення.
    Параметри
    ----------
    gray : np.ndarray
        Одноканальне (напівтонове) зображення.
```

```
    Повертає
    ---------
    float
        Дисперсія Лапласіана – використовується як міра різкості.
    """
    return cv2.Laplacian(gray, cv2.CV_64F).var()


def extract_frames(
    input_path: str,
    frame_stride: int = 5,
    max_frames: int = 200,
    sharpness_threshold: float = 100.0,
    ssim_threshold: float = 0.95,
) -> List[np.ndarray]:
    """
    Вилучає кадри з відео з кроковою вибіркою та двоетапною фільтрацією якості.
    Етап 1 – фільтр різкості: кадри, дисперсія Лапласіана яких нижча за
    ``sharpness_threshold``, відкидаються (розмиті / смазані кадри).
    Етап 2 – SSIM-дедуплікація: якщо новий кандидат надто схожий
    (SSIM >= ``ssim_threshold``) на останній прийнятий кадр, він пропускається,
    щоб уникнути обробки майже однакових ракурсів.
    Параметри
    ----------
    input_path : str
        Шлях до вхідного відеофайлу.
    frame_stride : int
        Брати кожен N-й сирий кадр перед перевіркою якості.
    max_frames : int
        Верхня межа кількості прийнятих кадрів.
    sharpness_threshold : float
        Мінімальна дисперсія Лапласіана для збереження кадру.
    ssim_threshold : float
        Максимальна структурна подібність між сусідніми прийнятими кадрами.
    Повертає
    ---------
    frames : List[np.ndarray]
        Список прийнятих BGR-зображень у вигляді масивів NumPy.
    """
    cap = cv2.VideoCapture(input_path)
    if not cap.isOpened():
        logger.error("Cannot open video: %s", input_path)
        raise IOError(f"Cannot open video: {input_path}")

    frames: List[np.ndarray] = []
    last_gray: Optional[np.ndarray] = None  # останній прийнятий кадр у відтінках
сірого
    raw_idx = 0

    while True:
        ret, frame = cap.read()
```

```
        if not ret:
            break

        # Крокова підвибірка – обробляємо лише кожен N-й кадр
        if raw_idx % frame_stride != 0:
            raw_idx += 1
            continue
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # --- Етап 1: фільтр різкості ---
        lap_var = _sharpness(gray)
        if lap_var < sharpness_threshold:
            raw_idx += 1
            continue

        # --- Етап 2: SSIM-дедуплікація ---
        if last_gray is not None:
            similarity = ssim(last_gray, gray, data_range=255)
            if similarity >= ssim_threshold:
                raw_idx += 1
                continue

        frames.append(frame.copy())
        last_gray = gray
        raw_idx += 1

        if len(frames) >= max_frames:
            break

    cap.release()
    logger.info(
        "Extracted %d frames (stride=%d, sharpness_thr=%.0f, ssim_thr=%.2f).",
        len(frames), frame_stride, sharpness_threshold, ssim_threshold,
    )
    return frames

# ============================================================================
# 2, 3, 5. Детекція обличчя, оцінка пози та вирівнювання через RetinaFace
# ============================================================================

# Індекси ключових точок у форматі InsightFace / RetinaFace
_LM_LEFT_EYE    = 0
_LM_RIGHT_EYE   = 1
_LM_NOSE        = 2
_LM_MOUTH_LEFT  = 3
_LM_MOUTH_RIGHT = 4

# Канонічні 2D-координати для фронтального кропу 112×112.
# Джерело: стандартний шаблон вирівнювання ArcFace / RetinaFace.
_REFERENCE_LANDMARKS_112 = np.array(
    [
        [38.2946, 51.6963],   # ліве око
```

```
        [73.5318, 51.5014],   # праве око
        [56.0252, 71.7366],   # кінчик носа
        [41.5493, 92.3655],   # лівий кут рота
        [70.7299, 92.2041],   # правий кут рота
    ],
    dtype=np.float32,
)


def _estimate_head_pose(landmarks: np.ndarray) -> Tuple[float, float]:
    """
    Оцінює кути yaw і pitch (у градусах) за 5 ключовими точками RetinaFace
    без повноцінного PnP-солвера.
    Метод – легка геометрична апроксимація:
      - Yaw   обчислюється через горизонтальне зміщення кінчика носа
        відносно середини між очима; нормується на міжочну відстань.
      - Pitch обчислюється через вертикальне положення кінчика носа
        відносно лінії очей і лінії рота.
    Точності достатньо для грубого відсіву (|yaw| > 40° або |pitch| > 30°).
    Параметри
    ----------
    landmarks : np.ndarray
        Масив форми (5, 2) у порядку:
        ліве_око, праве_око, ніс, лівий_кут_рота, правий_кут_рота.
    Повертає
    ---------
    yaw_deg : float
        Оцінений yaw у градусах (додатне = обличчя повернуте вправо).
    pitch_deg : float
        Оцінений pitch у градусах (додатне = обличчя піднято вгору).
    """
    left_eye    = landmarks[_LM_LEFT_EYE]
    right_eye   = landmarks[_LM_RIGHT_EYE]
    nose        = landmarks[_LM_NOSE]
    mouth_left  = landmarks[_LM_MOUTH_LEFT]
    mouth_right = landmarks[_LM_MOUTH_RIGHT]

    eye_center   = (left_eye + right_eye) / 2.0
    mouth_center = (mouth_left + mouth_right) / 2.0
    eye_width    = np.linalg.norm(right_eye - left_eye)

    # --- Апроксимація yaw ---
    # При повороті голови кінчик носа зміщується горизонтально від центру очей.
    # Емпіричний коефіцієнт: поворот ~45° ≈ зміщення 0.5 * eye_width
    nose_offset_x = nose[0] - eye_center[0]
    yaw_deg = float(np.degrees(np.arctan2(nose_offset_x, eye_width * 0.8)))

    # --- Апроксимація pitch ---
    # Вертикальна позиція носа між лінією очей та лінією рота.
    # На фронтальному обличчі ніс знаходиться приблизно на 45% висоти обличчя.
    face_height = mouth_center[1] - eye_center[1]
```

```
    if face_height < 1e-3:
        pitch_deg = 0.0
    else:
        nose_rel  = (nose[1] - eye_center[1]) / face_height - 0.45
        pitch_deg = float(np.degrees(np.arctan2(nose_rel * face_height,
                                                face_height * 0.5)))

    return yaw_deg, pitch_deg


def detect_and_align_faces(
    frames: List[np.ndarray],
    output_size: int = 112,
    yaw_threshold: float = 40.0,
    pitch_threshold: float = 30.0,
) -> Tuple[List[np.ndarray], List[float]]:
    """
    Для кожного кадру:
      1. Запускає RetinaFace для детекції домінантного обличчя та отримання
         5 ключових точок.
      2. Оцінює кут повороту голови з ключових точок.
      3. Відкидає кадри де |yaw| > yaw_threshold або |pitch| > pitch_threshold.
      4. Обчислює афінне перетворення від виявлених точок до канонічного
         шаблону та отримує вирівняний кроп.

    RetinaFace використовується і для детекції, і для вирівнювання –
    додаткова бібліотека ключових точок не потрібна.
    Параметри
    ----------
    frames : List[np.ndarray]
        BGR-зображення з функції extract_frames().
    output_size : int
        Розмір сторони квадратного вихідного кропу у пікселях.
    yaw_threshold : float
        Максимальний допустимий |yaw| у градусах.
    pitch_threshold : float
        Максимальний допустимий |pitch| у градусах.
    Повертає
    ---------
    face_crops : List[np.ndarray]
        Вирівняні BGR-зображення обличчя форми (output_size, output_size, 3).
    confidences : List[float]
        Оцінка впевненості детектора для кожного прийнятого кропу.
    """
    face_crops:  List[np.ndarray] = []
    confidences: List[float]      = []

    # Ініціалізація RetinaFace.
    # Stdout перехоплюється у буфер щоб приховати службові print-повідомлення
    # бібліотеки ("Applied providers", "find model" тощо).
    # При виникненні винятку буфер виводиться у лог перед перекиданням помилки,
```

```
    # щоб жодна діагностична інформація не була втрачена.
    _buf = io.StringIO()
    _old_stdout = sys.stdout
    sys.stdout  = _buf
    try:
        # CPUExecutionProvider – детекція виконується на процесорі (CPU).
        # Для прискорення можна замінити на CUDAExecutionProvider за умови
        # встановлення onnxruntime-gpu.
        app = FaceAnalysis(providers=['CUDAExecutionProvider'])
        app.prepare(ctx_id=0, det_size=(640, 640))
    except Exception:
        sys.stdout = _old_stdout
        _captured  = _buf.getvalue()
        if _captured.strip():
            logger.error("RetinaFace init failed. Captured output::\n%s",
                         _captured)
        raise
    finally:
        sys.stdout = _old_stdout

    # Масштабуємо канонічні точки до потрібного розміру кропу
    ref_lm = _REFERENCE_LANDMARKS_112 * (output_size / 112.0)

    logger.info("Running RetinaFace detection on %d frames...", len(frames))

    for i, frame in enumerate(tqdm(frames, desc="Detecting faces",
unit="frame")):

        faces = app.get(frame)

        # Пропускаємо кадри без виявленого обличчя
        if faces is None or len(faces) == 0:
            continue

        # Обираємо обличчя з найвищим score детектора
        face       = max(faces, key=lambda f: f.det_score)
        landmarks  = face.kps.astype(np.float32)   # (5, 2)
        confidence = float(face.det_score)

        # --- Фільтр за кутом повороту ---
        yaw, pitch = _estimate_head_pose(landmarks)
        if abs(yaw) > yaw_threshold or abs(pitch) > pitch_threshold:
            continue

        # --- Афінне вирівнювання за 5 ключовими точками ---
        # estimateAffinePartial2D – стійкий до шуму в точках метод LMEDS
        transform, _ = cv2.estimateAffinePartial2D(
            landmarks, ref_lm, method=cv2.LMEDS
        )
        if transform is None:
            continue
```

```
            aligned = cv2.warpAffine(
                frame, transform, (output_size, output_size),
            )

            face_crops.append(aligned)
            confidences.append(confidence)

        logger.info(
            "Accepted %d / %d frames after filtering.",
            len(face_crops), len(frames),
        )
        return face_crops, confidences

# ============================================================================
# 4, 6. Реконструкція 3D-форми за допомогою MICA (батчева обробка)
# ============================================================================

class FaceReconstructionModel:
    """
    Обгортка навколо нейронної мережі MICA для реконструкції 3D-форми обличчя.
    MICA передбачає коефіцієнти форми ідентичності FLAME з одного
    вирівняного зображення обличчя. Обробка виконується фіксованими
    батчами на GPU щоб уникнути переповнення відеопам'яті.
    Посилання
    ----------
    Zielonka et al., "Towards Metrical Reconstruction of Human Faces",
    ECCV 2022. https://github.com/Zielon/MICA
    """

    def __init__(self, device: str = "cuda:0", chunk_size: int = 16):
        if MICA is None:
            raise ImportError(
                "MICA not found. Clone https://github.com/Zielon/MICA "
                "and ensure it is on PYTHONPATH."
            )

        # Якщо запитаний GPU недоступний – автоматично переходимо на CPU
        self.device     = torch.device(device if torch.cuda.is_available() else
"cpu")
        self.chunk_size = chunk_size

        logger.info("Loading MICA on device: %s", self.device)

        cfg = get_cfg_defaults()

        # Видаляємо стандартний loguru-обробник і замінюємо його на такий,
        # що пропускає лише WARNING та вище. Таким чином INFO-повідомлення
        # від FLAME і MICA при ініціалізації не виводяться, але ERROR/WARNING
        # залишаються видимими.
        loguru_logger.remove()
```

```
        loguru_logger.add(
            sys.stderr,
            level="WARNING",
            format="{time:HH:mm:ss} [LOGURU {level}] {message}",
        )

        # MICA та FLAME виконуються на GPU (self.device)
        self.model = MICA(cfg).to(self.device)
        self.model.eval()

        logger.info("MICA loaded successfully.")

    @torch.no_grad()
    def reconstruct_batch(
        self,
        images_bgr: List[np.ndarray],
    ) -> Tuple[List[np.ndarray], List[np.ndarray]]:
        """
        Реконструює 3D-обличчя зі списку BGR-кропів за допомогою MICA.
        Обробка виконується чанками по ``self.chunk_size`` зображень,
        після кожного чанку відеопам'ять GPU звільняється.
        Для кожного чанку готуються два окремі тензори:
          - ``batch``         : 224×224, RGB, float32 у [0, 1]   → енкодер MICA
          - ``arcface_batch`` : 112×112, RGB, float32 у [-1, 1]  → енкодер
ArcFace
        Перед викликом decode() встановлюється ``self.model.testing = True``,
        щоб decode() пропустив гілку тренування, яка вимагає codedict['flame']
        (див. micalib/models/mica.py, рядок 89).
        Параметри
        ----------
        images_bgr : List[np.ndarray]
            Вирівняні BGR-кропи обличчя з detect_and_align_faces().
            Будь-який розмір приймається – resize виконується всередині.
        Повертає
        ---------
        verts_list : List[np.ndarray]
            Масиви вершин для кожного кадру, форма (V, 3).
        shape_params_list : List[np.ndarray]
            Вектори коефіцієнтів форми MICA для кожного кадру, форма (n_shape,).
        """
        if len(images_bgr) == 0:
            return [], []

        all_verts: List[np.ndarray] = []
        all_shape: List[np.ndarray] = []

        # Режим тестування – обов'язковий для правильної роботи decode()
        self.model.testing = True

        logger.info(
            "Reconstructing %d faces in chunks of %d...",
```

```
            len(images_bgr), self.chunk_size,
        )

        for start in tqdm(
            range(0, len(images_bgr), self.chunk_size),
            desc="Reconstructing 3D faces",
            unit="chunk",
        ):
            chunk = images_bgr[start : start + self.chunk_size]

            # Конвертація BGR → RGB один раз; обидва тензори похідні від цього
списку
            imgs_rgb = [cv2.cvtColor(im, cv2.COLOR_BGR2RGB) for im in chunk]

            # --- Тензор 1: основний батч для енкодера MICA ---
            # Розмір: 224×224, float32, нормалізовано до [0, 1]
            imgs_224 = [
                cv2.resize(im, (224, 224), interpolation=cv2.INTER_LINEAR)
                for im in imgs_rgb
            ]
            batch = torch.stack(
                [torch.from_numpy(im.astype(np.float32) / 255.0).permute(2, 0, 1)
                 for im in imgs_224],
                dim=0,
            ).to(self.device)                                    # (B, 3, 224,
224)

            # --- Тензор 2: батч для енкодера ArcFace ---
            # Розмір: 112×112, float32, нормалізовано до [-1, 1]
            imgs_112 = [
                cv2.resize(im, (112, 112), interpolation=cv2.INTER_LINEAR)
                for im in imgs_rgb
            ]
            arcface_batch = torch.stack(
                [torch.from_numpy(im.astype(np.float32) / 255.0).permute(2, 0, 1)
                 for im in imgs_112],
                dim=0,
            ).to(self.device)                                     # (B, 3, 112,
112)
            arcface_batch = (arcface_batch - 0.5) / 0.5          #
перемасштабування [0,1] → [-1,1]

            # --- Прямий прохід MICA ---
            # encode() заповнює: codedict['arcface'] (ембединг ідентичності),
            #                    codedict['images']
            # decode() використовує self.model.testing=True
            #          повністю пропускає гілку з flame GT
            codedict = self.model.encode(batch, arcface_batch)
            opdict   = self.model.decode(codedict)

            # Ключі результату згідно з micalib/models/mica.py decode():
```

```
            # opdict['pred_canonical_shape_vertices'] : (B, V, 3)
            # opdict['pred_shape_code']               : (B, n_shape)
            verts  = opdict["pred_canonical_shape_vertices"].cpu().numpy()
            shapes = opdict["pred_shape_code"].cpu().numpy()

            for i in range(verts.shape[0]):
                all_verts.append(verts[i])
                all_shape.append(shapes[i])

            # Звільняємо відеопам'ять GPU перед наступним чанком
            del batch, arcface_batch, codedict, opdict
            torch.cuda.empty_cache()

        logger.info("Reconstruction complete: %d meshes produced.",
len(all_verts))
        return all_verts, all_shape

# ============================================================================
# 5. Злиття параметрів форми через медіану
# ============================================================================

def fuse_shape_parameters(
    shape_params_list: List[np.ndarray],
) -> np.ndarray:
    """
    Зливає параметри форми MICA з кількох кадрів через поелементну медіану.
    Медіана стійкіша до викидів порівняно зі звичайним середнім –
    кадри з залишковою оклюзією або незначними помилками пози,
    що пройшли фільтр, не спотворюють результат.
    Параметри
    ----------
    shape_params_list : List[np.ndarray]
        Кожен елемент – 1D-вектор коефіцієнтів форми (n_shape,).
    Повертає
    ---------
    fused_shape : np.ndarray
        Усереднений через медіану вектор коефіцієнтів форми.
    """
    if len(shape_params_list) == 0:
        raise ValueError("Empty list of shape parameters – nothing to fuse.")

    stacked     = np.stack(shape_params_list, axis=0)  # (N, n_shape)
    fused_shape = np.median(stacked, axis=0)            # (n_shape,)

    logger.info(
        "Fused %d shape vectors via median (n_shape=%d).",
        len(shape_params_list), fused_shape.shape[0],
    )
    return fused_shape

# ============================================================================
```

```
# 6. Декодування канонічного меша із злитих параметрів форми
# ============================================================================

def reconstruct_mesh_from_shape(
    recon_model: FaceReconstructionModel,
    fused_shape: np.ndarray,
) -> Tuple[np.ndarray, np.ndarray]:
    """
    Декодує канонічний нейтральний 3D-меш обличчя із злитих параметрів
    форми MICA.
    В якості морфованої моделі використовується FLAME. Нульові параметри
    виразу та пози забезпечують нейтральне положення спокою.
    Нотатки щодо цієї версії FLAME (з flame.py):
      - Вираз за замовчуванням: zeros([1, 100]) – передаємо явно для ясності.
      - Грані зберігаються у буфері 'faces_tensor' типу torch.long.
      - forward() повертає (vertices, landmarks2d, landmarks3d).
      - FLAME виконується на тому ж пристрої що і MICA (GPU або CPU).
    Параметри
    ----------
    recon_model : FaceReconstructionModel
        Обгортка з завантаженою моделлю MICA.
    fused_shape : np.ndarray
        1D-масив коефіцієнтів форми MICA, форма (n_shape).
    Повертає
    ---------
    verts : np.ndarray
        Позиції вершин, форма (V, 3).
    faces : np.ndarray
        Індекси трикутних граней, форма (F, 3), dtype int32.
    """
    model  = recon_model.model
    device = recon_model.device

    # MICA зберігає FLAME у model.flame (ініціалізується у BaseModel)
    flame = model.flame

    with torch.no_grad():
        # Форма: (1, n_shape) – додаємо batch-розмір
        shape_t = torch.from_numpy(fused_shape).float().unsqueeze(0).to(device)

        # Розмірність 100 (значення захардкоджено у flame.py, рядок
        #   expression_params = torch.zeros([1, 100], ...))
        exp  = torch.zeros((1, 100), dtype=torch.float32, device=device)

        # Поза: 6 значень = глобальний поворот (3) + поворот щелепи (3)
        pose = torch.zeros((1, 6),   dtype=torch.float32, device=device)

        # forward() → (vertices, landmarks2d, landmarks3d)
        vertices, _, _ = flame(
            shape_params=shape_t,
            expression_params=exp,
```

```
            pose_params=pose,
        )

        verts = vertices.squeeze(0).cpu().numpy()               # (V, 3)

        # faces_tensor – зареєстрований через register_buffer як torch.long
        faces = flame.faces_tensor.cpu().numpy().astype(np.int32)  # (F, 3)

    logger.info(
        "Decoded canonical mesh: %d vertices, %d faces.",
        verts.shape[0], faces.shape[0],
    )
    return verts, faces

# ============================================================================
# 7. Експорт меша в OBJ з опціональною постобробкою
# ============================================================================

def save_mesh_as_obj(
    verts: np.ndarray,
    faces: np.ndarray,
    output_path: str,
    apply_smoothing: bool = False,
    smooth_iterations: int = 5,
    smooth_lambda: float = 0.5,
) -> None:
    """
    Постобробляє та експортує трикутний меш у файл OBJ.
    Кроки постобробки:
      1. Очищення меша (завжди увімкнено): видаляє дублікати вершин,
         вироджені грані та нереференсовані вершини через trimesh.
      2. Згладжування Лапласа (опціонально): легке згладжування що
         прибирає високочастотний шум поверхні, зберігаючи загальну
         геометрію. Параметр volume_constraint=True запобігає «стисненню»
         при агресивному згладжуванні.
    Параметри
    ----------
    verts : np.ndarray
        Масив вершин форми (V, 3).
    faces : np.ndarray
        Масив індексів граней форми (F, 3).
    output_path : str
        Шлях до вихідного OBJ-файлу.
    apply_smoothing : bool
        Якщо True – застосувати згладжування Лапласа перед експортом.
    smooth_iterations : int
        Кількість ітерацій згладжування (використовується тільки при
        apply_smoothing=True).
    smooth_lambda : float
        Коефіцієнт згладжування у [0, 1] – більше = гладше, але більше
        «стиснення» форми.
```

```
    """
    # process=True запускає вбудований пайплайн відновлення trimesh:
    # об'єднання дублікатів вершин, видалення вироджених/дублікатних граней,
    # виправлення узгодженості обходу.
    mesh = trimesh.Trimesh(vertices=verts, faces=faces, process=True)

    logger.info(
        "Mesh after cleaning: %d vertices, %d faces.",
        len(mesh.vertices), len(mesh.faces),
    )

    if apply_smoothing:
        logger.info(
            "Applying Laplacian smoothing (iterations=%d, lambda=%.2f)...",
            smooth_iterations, smooth_lambda,
        )
        trimesh.smoothing.filter_laplacian(
            mesh,
            lamb=smooth_lambda,
            iterations=smooth_iterations,
            implicit_time_integration=False,
            volume_constraint=True,
        )
    else:
        logger.info("Laplacian smoothing skipped (apply_smoothing=False).")

    # Створюємо директорію якщо потрібно
    out_dir = os.path.dirname(output_path)
    if out_dir:
        os.makedirs(out_dir, exist_ok=True)
    mesh.export(output_path)
    logger.info("Saved mesh to: %s", output_path)

# ============================================================================
# 8. Повний пайплайн
# ============================================================================

def run_pipeline(
    input_path: str,
    output_obj_path: str,
    frame_stride: int = 5,
    max_frames: int = 200,
    sharpness_threshold: float = 100.0,
    ssim_threshold: float = 0.95,
    yaw_threshold: float = 40.0,
    pitch_threshold: float = 30.0,
    device: str = "cuda:0",
    chunk_size: int = 16,
    apply_smoothing: bool = False,
    smooth_iterations: int = 5,
    smooth_lambda: float = 0.5,
```

```
) -> None:
    """
    Наскрізний пайплайн реконструкції 3D-обличчя з відеозапису.
    Кроки
    -----
    1. Вилучення кадрів з фільтрацією різкості та SSIM.
    2. Детекція облич, фільтрація за кутом повороту голови,
       вирівнювання кропів – все через RetinaFace.
    3. Реконструкція 3D-форми для кожного кадру через MICA (батчево).
    4. Злиття параметрів форми через поелементну медіану.
    5. Декодування злитої форми у канонічний нейтральний меш.
    6. Постобробка (очищення + опціональне згладжування) та експорт OBJ.
    Параметри
    ----------
    input_path : str
        Вхідний відеофайл.
    output_obj_path : str
        Шлях до вихідного OBJ-файлу.
    frame_stride : int
        Брати кожен N-й сирий кадр.
    max_frames : int
        Максимальна кількість прийнятих кадрів.
    sharpness_threshold : float
        Мінімальна дисперсія Лапласіана для збереження кадру.
    ssim_threshold : float
        Максимальний SSIM між сусідніми кадрами (дедуплікація).
    yaw_threshold : float
        Максимальний |yaw| у градусах.
    pitch_threshold : float
        Максимальний |pitch| у градусах.
    device : str
        Рядок пристрою PyTorch (наприклад, 'cuda:0' або 'cpu').
    chunk_size : int
        Розмір батчу GPU для реконструкції.
    apply_smoothing : bool
        Увімкнути згладжування Лапласа при експорті меша.
    smooth_iterations : int
        Кількість ітерацій згладжування (якщо увімкнено).
    smooth_lambda : float
        Коефіцієнт згладжування (якщо увімкнено).
    """
    log_path = os.path.splitext(output_obj_path)[0] + ".log"
    _fh = logging.FileHandler(log_path, mode="w", encoding="utf-8")
    _fh.setFormatter(logging.Formatter(
        fmt="%(asctime)s [%(levelname)s] %(message)s",
        datefmt="%H:%M:%S",
    ))
    logging.getLogger().addHandler(_fh)
    logger.info("=== Pipeline start ===")

    # ------------------------------------------------------------------
```

```
    # Крок 1 – Вилучення та фільтрація кадрів
    # ------------------------------------------------------------------
    logger.info("Step 1: Extracting frames from %s ...", input_path)
    frames = extract_frames(
        input_path,
        frame_stride=frame_stride,
        max_frames=max_frames,
        sharpness_threshold=sharpness_threshold,
        ssim_threshold=ssim_threshold,
    )

    if len(frames) == 0:
        logger.error("No frames extracted from video. Aborting.")
        raise RuntimeError("No frames extracted from video.")

    # ------------------------------------------------------------------
    # Крок 2 – Детекція облич, фільтрація за позою, вирівнювання кропів
    # ------------------------------------------------------------------
    logger.info("Step 2: Face detection, pose filtering and alignment ...")
    face_crops, confidences = detect_and_align_faces(
        frames,
        yaw_threshold=yaw_threshold,
        pitch_threshold=pitch_threshold,
    )

    if len(face_crops) == 0:
        logger.error("No valid face crops after filtering. Aborting.")
        raise RuntimeError("No valid face crops after filtering.")

    # ------------------------------------------------------------------
    # Крок 3 – Завантаження MICA та реконструкція 3D-форм
    # ------------------------------------------------------------------
    logger.info("Step 3: Loading MICA and reconstructing per-frame 3D shapes
...")
    recon_model = FaceReconstructionModel(device=device, chunk_size=chunk_size)

    _, shape_params_list = recon_model.reconstruct_batch(face_crops)

    if len(shape_params_list) == 0:
        logger.error("Reconstruction produced no shape parameters. Aborting.")
        raise RuntimeError("Reconstruction failed for all frames.")

    # ------------------------------------------------------------------
    # Крок 4 – Злиття параметрів форми (медіана)
    # ------------------------------------------------------------------
    logger.info("Step 4: Fusing shape parameters via median ...")
    fused_shape = fuse_shape_parameters(shape_params_list)
    # Зберігаємо shape-вектор поряд з OBJ для подальшого порівняння
    # через face_compare.py --mode shape
    shape_npy_path = os.path.splitext(output_obj_path)[0] + ".npy"
    np.save(shape_npy_path, fused_shape)
```

```
    logger.info("Shape vector saved: %s", shape_npy_path)


    # ------------------------------------------------------------------
    # Крок 5 – Декодування злитої форми у канонічний меш
    # ------------------------------------------------------------------
    logger.info("Step 5: Decoding fused shape to canonical mesh ...")
    verts, faces = reconstruct_mesh_from_shape(recon_model, fused_shape)

    # ------------------------------------------------------------------
    # Крок 6 – Постобробка та експорт
    # ------------------------------------------------------------------
    logger.info("Step 6: Post-processing and exporting mesh to %s ...",
output_obj_path)
    save_mesh_as_obj(
        verts,
        faces,
        output_obj_path,
        apply_smoothing=apply_smoothing,
        smooth_iterations=smooth_iterations,
        smooth_lambda=smooth_lambda,
    )

    logger.info("=== Pipeline complete ===")


# ============================================================================
# 9. Точка входу CLI
# ============================================================================


def parse_args() -> argparse.Namespace:
    parser = argparse.ArgumentParser(
        description="3D human face reconstruction from a video sequence or
image."
    )
    parser.add_argument(
        "--input", type=str, required=True,
        help="Path to input file."
    )
    parser.add_argument(
        "--output", type=str, required=True,
        help="Path to output OBJ file (e.g., ./output/face.obj)."
    )
    parser.add_argument(
        "--frame_stride", type=int, default=5,
        help="Use every N-th raw frame (default: 5)."
    )
    parser.add_argument(
        "--max_frames", type=int, default=200,
        help="Maximum number of accepted frames (default: 200)."
```

```
    )
    parser.add_argument(
        "--sharpness_threshold", type=float, default=100.0,
        help="Minimum Laplacian variance for a frame to be kept (default:
100.0)."
    )
    parser.add_argument(
        "--ssim_threshold", type=float, default=0.95,
        help="Maximum SSIM between consecutive frames (default: 0.95)."
    )
    parser.add_argument(
        "--yaw_threshold", type=float, default=40.0,
        help="Maximum |yaw| in degrees (default: 40)."
    )
    parser.add_argument(
        "--pitch_threshold", type=float, default=30.0,
        help="Maximum |pitch| in degrees (default: 30)."
    )
    parser.add_argument(
        "--device", type=str, default="cuda:0",
        help="PyTorch device string (default: 'cuda:0')."
    )
    parser.add_argument(
        "--chunk_size", type=int, default=16,
        help="GPU batch size for reconstruction (default: 16)."
    )
    parser.add_argument(
        "--apply_smoothing", action="store_true",
        help="Enable Laplacian smoothing during mesh export."
    )
    parser.add_argument(
        "--smooth_iterations", type=int, default=5,
        help="Number of Laplacian smoothing iterations (default: 5)."
    )
    parser.add_argument(
        "--smooth_lambda", type=float, default=0.5,
        help="Laplacian smoothing factor in [0, 1] (default: 0.5)."
    )
    return parser.parse_args()


if __name__ == "__main__":
    args = parse_args()
    run_pipeline(
        input_path=args.input,
        output_obj_path=args.output,
        frame_stride=args.frame_stride,
        max_frames=args.max_frames,
        sharpness_threshold=args.sharpness_threshold,
        ssim_threshold=args.ssim_threshold,
        yaw_threshold=args.yaw_threshold,
```

```
    pitch_threshold=args.pitch_threshold,
    device=args.device,
    chunk_size=args.chunk_size,
    apply_smoothing=args.apply_smoothing,
    smooth_iterations=args.smooth_iterations,
    smooth_lambda=args.smooth_lambda,
)
```

ДОДАТОК В

Приклади роботи пайплайну

