МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
Кафедра штучного інтелекту

До захисту допущено:
В. о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«___»_____2026 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи і методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Інтелектуальна система AI Motion Capture для анімації 3D-моделі людини з використанням методів комп'ютерного зору та аудіоаналізу»

Виконав:
студент IV курсу, групи КІ-21
Вареня Андрій Вячеславович _____

Керівник:
старший викладач кафедри штучного інтелекту,
PhD, Коломоєць Сергій Олексійович _____

Консультант з нормоконтролю:
фахівець першої категорії кафедри штучного інтелекту,
к.т.н., доцент, Комариста Богдана Миколаївна _____

Рецензент:
асистент кафедри математичних методів системного аналізу,
PhD, Канцедал Георгій Олегович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В. о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«15» січня 2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Вареня Андрій Вячеславович

1. Тема роботи «Інтелектуальна система AI Motion Capture для анімації 3D-моделі людини з використанням методів комп'ютерного зору та аудіоаналізу», керівник роботи Коломоєць Сергій Олексійович, старший викладач кафедри штучного інтелекту, PhD, затверджені наказом по НН ІПСА № 1944-с від «27» травня 2026 року.

2. Термін подання студентом роботи «05» червня 2026 року.

3. Вихідні дані до роботи: монокулярні відеозаписи руху людини з синхронним аудіосигналом, попередньо навчені моделі оцінки пози тіла, кистей, обличчя, аудіокерованої лицевої анімації та розпізнавання емоцій.

4. Зміст роботи: аналіз предметної області AI-захоплення руху; огляд моделей комп'ютерного зору й аудіоаналізу; розроблення канальної моделі даних і формату обміну AIMOCAP; формалізація producer- і consumer-алгоритмів системи; програмна реалізація бібліотеки, настільного редактора і тестового доповнення Blender; експериментальна перевірка системи.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): електронна презентація.

6. Дата видачі завдання «02» лютого 2026 року.

Календарний план

№ з/п	*Назва етапів виконання дипломної роботи*	*Термін виконання етапів роботи*	*Примітка*
1	Огляд літератури за темою	20.04.2026	Виконано
2	Підготовка першого розділу	27.04.2026	Виконано
3	Підготовка другого розділу	04.05.2026	Виконано
4	Розробка програмного продукту	11.05.2026	Виконано
5	Підготовка третього розділу	18.05.2026	Виконано
6	Оформлення дипломної роботи	25.05.2026	Виконано
7	Підготовка електронної презентації	01.06.2026	Виконано
8	Передання роботи на перевірку	05.06.2026	Виконано

Студент Андрій ВАРЕНЯ

Керівник Сергій КОЛОМОЄЦЬ

РЕФЕРАТ

Дипломна робота: 109 с., 31 рис., 16 табл., 53 посилання, 1 додаток.

AI-ЗАХОПЛЕННЯ РУХУ, КОМП'ЮТЕРНИЙ ЗІР, АУДІОАНАЛІЗ, SMPL-X, РЕГІОНАЛЬНЕ ЗЛИТТЯ, BLENDER, AIMOCAP.

Об'єкт дослідження – процес безмаркерного захоплення руху людини за монокулярним відео та синхронним аудіо. Предмет дослідження – методи багатоканального подання, синхронізації, очищення та регіонального злиття результатів нейромережевого інференсу для анімації персонажа Blender/DAZ. Мета роботи – підвищити ефективність і доступність створення анімації 3D-моделі людини шляхом розроблення системи AI-захоплення руху, у якій тіло, кисті, голова, відеоміміка, аудіоміміка та емоції зберігаються окремими каналами у форматі AIMOCAP. Для досягнення мети оглянуто сучасні методи безмаркерного захоплення руху, моделі SMPL-X, MediaPipe, Audio2Face-3D, HaMeR та засоби розпізнавання емоцій; проаналізовано обмеження готових програмних рішень; розроблено канальну модель декомпозиції, самоописний формат .aimocap, producer-алгоритми калібрування, очищення і кодування та consumer-алгоритми декодування, синхронізації, регіонального злиття і емоційного шару. Реалізовано бібліотеку aimocap, настільний редактор на Qt 6 і тестове доповнення Blender/DAZ. Виконано експериментальну перевірку на відеофрагменті з послідовним увімкненням каналів, порівнянням джерел кистей і політик злиття рота. Встановлено, що спеціалізоване джерело кистей HaMeR зменшує джитер порівняно з NLF, а аудіоканал стабілізує рух рота за деградації відео; отримані файли AIMOCAP відтворюються у Blender з окремими редагованими доріжками тіла, обличчя й емоцій. Практичне значення розробки полягає у можливості використовувати систему в навчальних і дослідницьких сценаріях створення анімації без маркерів, інерційного костюма та багатокамерної сцени.

ABSTRACT

Bachelor's thesis: 109 p., 31 figures, 16 tables, 53 references, 1 appendix.

AI MOTION CAPTURE, COMPUTER VISION, AUDIO ANALYSIS, SMPL-X, REGIONAL FUSION, BLENDER, AIMOCAP.

The object of the study is the process of markerless human motion capture from monocular video and synchronized audio. The subject of research is the set of methods for multi-channel representation, synchronization, cleaning and regional fusion of neural-network inference results for Blender/DAZ character animation. The purpose of the work is to improve the efficiency and accessibility of 3D human animation by developing an AI motion capture system in which body motion, hands, head pose, video-driven facial motion, audio-driven facial motion and emotions are stored as separate channels in the AIMOCAP format. To achieve this purpose, modern markerless motion-capture methods, SMPL-X, MediaPipe, Audio2Face-3D, HaMeR and emotion-recognition models were reviewed; limitations of ready-made software solutions were analyzed; a channel decomposition model, the self-describing .aimocap format, producer algorithms for calibration, cleaning and encoding, and consumer algorithms for decoding, synchronization, regional fusion and the emotion layer were developed. The aimocap library, a Qt 6 desktop editor and a test Blender/DAZ add-on were implemented. Experimental verification was performed on a video fragment with sequential channel activation, comparison of hand-pose sources and mouth-fusion policies. The experiments showed that the specialized HaMeR hand source reduces jitter compared with NLF, while the audio channel stabilizes mouth motion under video degradation; the resulting AIMOCAP files are reproduced in Blender as separate editable tracks for body, face and emotions. The practical value of the system is its use in educational and research scenarios for creating animation without markers, inertial suits or a multi-camera stage.

ЗМІСТ

ВСТУП

Тут і нижче використані такі інструменти штучного інтелекту, як чат-бот з генеративним штучним інтелектом Claude, виключно для написання програмного коду, корегування та редагування тексту, створеного автором цієї дипломної роботи, на основі автоматизованої перевірки граматики, структури та стилю, що відповідає Політиці використання штучного інтелекту для академічної діяльності в КПІ ім. Ігоря Сікорського (протокол №11 Вченої ради КПІ ім. Ігоря Сікорського від 11 грудня 2023 р.).

Тривимірна анімація персонажа-людини довгий час була доступна лише великим студіям: промислові оптичні системи захоплення руху Vicon, OptiTrack, Qualisys коштують десятки тисяч доларів і потребують обладнаного павільйону з командою операторів [1], а інерційні костюми Xsens MVN та Rokoko Smartsuit Pro залишаються поза бюджетом окремого розробника чи освітньої лабораторії. Розвиток глибокого навчання змінив ситуацію: з’явилися архітектури, які реконструюють тривимірну позу людини з єдиного RGB-зображення без маркерів. У 2015 році параметрична модель тіла SMPL (Skinned Multi-Person Linear Model) [2] дала нейронним мережам компактну ціль регресії замість мільйонів координат вершин; у 2017 році OpenPose [3] показала детекцію ключових точок тіла у реальному часі з однієї камери; у 2018 році HMR (Human Mesh Recovery) [4] стала першою впливовою наскрізною моделлю регресії параметрів SMPL.

Паралельно розвивалися інші модальності. Apple зробила стандарт ARKit з 52 лицевими блендшейпами фактичним інтерфейсом лицевої анімації [5]; Google підтримує MediaPipe з моделями пози та обличчя для клієнтських пристроїв [6; 37; 38]; NVIDIA опублікувала стек Audio2Face-3D для анімації обличчя за аудіо [7]. Для пози кистей з’явилися спеціалізовані моделі HaMeR [45] та WiLoR [46] поверх параметричної моделі кисті MANO (hand Model with Articulated and Non-rigid defOrmations) [44].

Попри доступність готових моделей, кожен відкритий інструмент вирішує лише частину задачі. FreeMoCap [8] працює з тілом і потребує кількох камер; BlendArMocap захоплює пози та обличчя, але ігнорує аудіо й емоції; ThreeDPoseTracker підтримує лише тіло; Audio2Face-3D закриває обличчя й аудіо без тіла. У межах проведеного огляду не виявлено інструмента, який поєднує чотири сигнали – повне тіло з кистями, обличчя, аудіокерований рот, емоції – і зберігає результати окремих моделей у форматі, придатному для незалежного редагування каналів.

Цей розрив і визначає тему дипломної роботи. Розроблена система складається з трьох частин: бібліотеки aimocap, настільного редактора на Qt 6 і тестового вендорського доповнення Blender 4.x. Бібліотека послідовно запускає попередньо навчені моделі за чотирма модальностями, очищає їхні результати і зберігає кожен канал спостережень окремо у власному самоописному форматі .aimocap. Редактор керує обробкою: дозволяє обрати діапазон кадрів, переглянути кожен канал у viewport-ах і на таймлайні, експортувати файл і повторно його відкрити. Споживач читає файл, синхронізує канали до власної частоти кадрів, виконує регіональне злиття лицевих каналів і передає узгоджені значення у вендорський адаптер цільової програми. У роботі таким адаптером є доповнення Blender для персонажа DAZ Genesis (DAZ Studio Genesis): воно демонструє, як тіло, міміка та емоційний шар розкладаються на окремі доріжки нелінійної анімації. Об’єднання каналів відбувається на боці споживача в момент застосування, тоді як файл зберігає очищені спостереження кожної моделі разом із рекомендованою політикою їхнього поєднання.

Метою дипломної роботи є підвищення ефективності та доступності процесу створення анімації 3D-моделі людини шляхом інтелектуального багатоканального AI-захоплення руху з монокулярного відео і синхронного аудіо, за якого очищені спостереження різних нейромережевих моделей зберігаються у власному форматі .aimocap, їхня синхронізація та регіональне злиття виконуються на боці споживача, а практична придатність

підтверджується еталонним адаптером Blender/DAZ Studio. Для досягнення цієї мети виконано такі завдання:

- проаналізовано сучасний стан безмаркерного захоплення руху та моделі за окремими модальностями: поза тіла, поза кистей, лицеві рухи, аудіокерована анімація обличчя, розпізнавання емоцій;
- обґрунтовано канальну модель декомпозиції та шарову модель даних, за якою файл зберігає спостереження моделей, а рішення про їхнє поєднання ухвалює споживач;
- спроєктовано самоописний формат обміну .aimocap з покадровими довірами, регіональним словником ARKit-52 та рекомендованою політикою злиття;
- формалізовано producer-алгоритми калібрування, очищення і кодування та consumer-алгоритми декодування, синхронізації, регіонального злиття, емоційного шару і контракту адаптера цільового рига;
- реалізовано бібліотеку aimocap, настільний редактор на Qt 6 і тестове вендорське доповнення Blender 4.x;
- проведено експериментальну перевірку з послідовним увімкненням каналів, порівнянням джерел пальців та політик злиття рота.

Об’єктом дослідження є процес безмаркерного захоплення повного руху людини (тіло, кисті, міміка, мовлення, емоції) з монокулярного відео та синхронного звуку.

Предметом дослідження є методи багатоканального подання, синхронізації, очищення та регіонального злиття результатів нейромережевого інференсу з подальшою практичною перевіркою через вендорський адаптер Blender/DAZ.

Методи дослідження – аналіз наукової літератури, порівняння опублікованих характеристик моделей, формалізація алгоритмічних послідовностей, програмна реалізація та експериментальна перевірка абляційними експериментами на тестових відеофрагментах.

Практична цінність роботи полягає у створенні завершеного програмного комплексу, що покриває чотири модальності одночасно й працює на побутовому апаратному забезпеченні. Формат .aimocap виконує роль контракту між моделями і споживачами: кожен канал анімації можна переглянути, вимкнути або замінити незалежно від інших, а політика поєднання каналів залишається повністю керованою на боці споживача. Такої організації даних не дає жоден з розглянутих готових інструментів і жоден з поширених форматів обміну анімацією.

Структура роботи. Розділ 1 присвячено аналізу актуальності задачі, поясненню принципів 3D-анімації, історичному розвитку напряму, огляду доступних моделей за всіма модальностями та формулюванню постановки задачі. Розділ 2 описує канальну модель декомпозиції, словники даних, формат обміну .aimocap, producer-алгоритми та consumer-ланцюг від декодування до передачі даних у вендорський адаптер. У третьому розділі описано реалізований програмний комплекс і наведено експериментальну перевірку з послідовним увімкненням каналів.

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ AI-ЗАХОПЛЕННЯ РУХУ

1.1 Захоплення руху як інженерна задача

Захоплення руху (motion capture) – це процес запису рухів живого виконавця та перенесення цих рухів на цифрового персонажа. Технологія народилася на перетині кіноіндустрії та комп'ютерної графіки у 1990-х роках і стала індустріальним стандартом анімації людиноподібних персонажів в іграх, кіно та віртуальному виробництві.

Інженерна складність задачі випливає з кількості ступенів свободи. Поза людини в одному кадрі описується приблизно 70–100 числовими параметрами: суглоби тіла, пальці обох кистей, щелепа, очі, м'язи обличчя. Запис руху вимагає фіксувати ці параметри 30–60 разів на секунду, а потім переносити їх на персонажа з іншими пропорціями та іншою топологією скелета без артефактів на кшталт ковзання стоп або вивернутих колін.

Класичних підходів два. Оптичне маркерне захоплення тріангулює позиції ретрорефлективних маркерів десятками інфрачервоних камер (Vicon Shōgun, OptiTrack Motive, Qualisys Track Manager) з міліметровою точністю і частотою 120–240 Гц; ціна – спеціально обладнане приміщення, камери від 5000 доларів США за одиницю і години підготовки на сесію [1]. Інерційне захоплення кріпить сенсори IMU (Inertial Measurement Unit – інерційний вимірювальний блок) на сегменти тіла і відновлює орієнтації фільтром Калмана або комплементарним фільтром (Xsens MVN, Rokoko Smartsuit Pro); воно мобільне, але страждає від дрейфу магнітометра у приміщеннях з металом, потребує повторного калібрування і коштує від 2500 доларів США за костюм [9].

Третій підхід, предмет цієї роботи, – безмаркерне AI-захоплення з одного відеофайла побутової камери та мікрофона: попередньо навчені нейронні мережі оцінюють тривимірну позу безпосередньо за зображенням. Точність нижча за оптичні системи, але достатня для незалежного розробника, освітньої лабораторії чи дослідника, а вхідний бар'єр знижують промислові екосистеми:

стандарт ARKit на 52 блендшейпи, компоненти MediaPipe, набір NVIDIA Audio2Face-3D [5; 6; 7; 37; 38]. Окрім пози тіла, ця робота охоплює ще дві модальності, що зазвичай випадають з оглядів захоплення руху: аудіокеровану лицеву анімацію та розпізнавання емоцій – саме поєднання тіла, обличчя і голосу відрізняє персонажа, який говорить, від персонажа, який лише рухається.

1.2 Як влаштована тривимірна анімація персонажа

Подальші підрозділи постійно оперують поняттями блендшейпів, ригів і ретаргетингу, тому коротко зафіксуємо технічний мінімум тривимірної анімації персонажа.

Цифровий персонаж складається з меша (поверхня з тисяч вершин), скелета (ієрархічне дерево кісток) і матеріалів; останні поза розглядом роботи. Деформацію меша за рухом скелета задає скінінгове зважування: кожна вершина має ваги прив’язки до 2–4 кісток і рухається як зважена сума їхніх трансформацій. Ця процедура – лінійне скінінгове змішування (LBS) – є стандартом усіх 3D-двигунів і повторно зустрінеться у розділі 2 як алгоритмічний примітив.

Скелет має корінь у тазу, а кожна кістка зберігає локальне обертання відносно батьківської. Сукупність локальних обертань утворює позу: для тіла з 24 кістками у поданні вісь-кут це вектор з 72 чисел. Саме такі дані має видавати система захоплення руху для тіла.

Міміка керується інакше – блендшейпами (ключі форми у Blender, морф-цілі у Maya). Блендшейп – заздалегідь створений варіант деформації меша обличчя під один вираз («рот відкрито», «ліве око закрите»); фінальна форма обличчя є зваженою сумою зміщень з вагами від 0 до 1. Фактичним індустріальним словником стали 52 блендшейпи стандарту ARKit від Apple (iPhone X, 2017): jawOpen, mouthSmile_L, browDownLeft тощо. Номенклатура

узгоджена з системою кодування мімічних рухів FACS (Facial Action Coding System) Пола Екмана (1978), яка виділяє елементарні анатомічні рухи обличчя – Action Units [5; 10]. Стан обличчя у кадрі, отже, описується вектором з 52 чисел.

Ретаргетинг – перенесення руху на персонажа з іншими пропорціями. Локальні обертання кісток переносяться без масштабування, але коренева трансляція має бути узгоджена зі зростом цільової моделі, а розбіжність нейтральних поз джерела і цілі – скомпенсована. У цій роботі ретаргетинг не заявляється як окремий науковий внесок: формат .aimocap і consumer-алгоритми лише подають синхронізовані канали та політику застосування, а конкретне перенесення на кістки, morph/FACS-властивості або animation curves виконує адаптер цільової програми. Практичним прикладом такого адаптера є доповнення Blender для рига DAZ Genesis у режимах native і MHX (MakeHuman eXchange).

Екосистема, в якій працює система: Blender – відкритий 3D-редактор з Python API (Application Programming Interface) для доповнень, у ньому й реалізовано тестове доповнення; Rigify і MHX – поширені відкриті конвенції ригів; Unreal Engine приймає 52 ARKit-блендшейпи через Live Link Face; персонажі DAZ Genesis постачаються з FACS-контролерами, на які ARKit-коефіцієнти переносяться таблицею відповідності. DAZ Genesis обрано тестовим ригом як найвимогливіший до коректності лицевих даних. Саме на такому ригу добре видно помилки узгодження каналів: неточності тіла проявляються у позі рук, а помилки лицевого шару – у роті, бровах і очах. Тому перевірка на DAZ Genesis є не найпростішим, а навпаки показовим сценарієм для оцінювання придатності формату і адаптера. Обмін анімацією між середовищами зазвичай іде через формат FBX (Filmbox), який зберігає одну уніфіковану позу на кадр без розділення джерел сигналу; для задачі з незалежно вимиканими каналами тіла, кистей, обличчя, аудіо-рота й емоцій цього недостатньо, що і мотивує власний формат .aimocap розділу 2.

1.3 Від маркерних систем до безмаркерного нейромережевого захоплення руху

Історія захоплення руху ділиться на три етапи, кожен з приблизно десятилітнім домінуванням; кожен попередній етап зберігався, поступово втрачаючи частку серед нових проєктів.

Перший етап – оптичні маркерні системи, що масово увійшли в індустрію наприкінці 1990-х. Камери з частотою 120–2000 Гц розставляються по периметру павільйону, актор одягає костюм з 30–50 ретрорефлективними маркерами, програмне забезпечення триангулює координати маркерів, асоціює їх зі скелетною моделлю і відновлює обертання суглобів зворотною кінематикою [1]. Точність цих систем еталонна, проте вони потребують спеціально обладнаного павільйону і тривалої підготовки кожної сесії.

Другий етап – інерційні костюми 2010-х. Сенсори IMU (гіроскоп, акселерометр, магнітометр) на сегментах тіла дають мобільність без камер, але потерпають від дрейфу магнітометра біля металу, потребують повторного калібрування кожні 10–15 хвилин і реконструюють лише пози без абсолютної позиції у просторі [9].

Третій етап – безмаркерне нейромережеве захоплення. Прорив зробила параметрична модель тіла SMPL: геометрія людини описується функцією двох коротких векторів – форми (β, 10 компонент) і пози (θ, 72 компоненти вісь-кут) – замість 6890 координат вершин, що звело задачу регресії до кількох десятків параметрів [2]. У 2017 році OpenPose показала детекцію 2D-ключових точок тіла у реальному часі на основі Part Affinity Fields [3], а у 2018 році HMR (Human Mesh Recovery) стала першою впливовою наскрізною моделлю регресії параметрів SMPL з RGB-кадру з адверсаріальним наглядом за правдоподібністю поз [4].

Подальший розвиток ішов поколіннями. VIBE (Video Inference for Body Pose and Shape Estimation, 2020) додала часовий GRU-блок (Gated Recurrent Unit

– рекурентна нейронна мережа з механізмом вентилів для обробки послідовностей) проти дрижання між кадрами [11]; PARE (Part Attention Regressor, 2021) – увагу до частин тіла заради стійкості до оклюзій [12]; HMR2.0 / 4D-Humans (2023) перевела регресію на ViT-H/16 (Vision Transformer – трансформерна модель для обробки зображень, варіант Huge з патчами 16×16 пікселів) з MPJPE (Mean Per Joint Position Error, середньою позиційною похибкою суглобів) 70,0 мм (2.0a) і 81,3 мм (2.0b) на 3DPW [14]. У 2024 році WHAM винесла рух у глобальну систему координат через SLAM-оцінку (Simultaneous Localization and Mapping) камери і контакти стоп (MPJPE 57,8 мм на 3DPW) [15]; поруч з'явилися одноетапна повнотіла Multi-HMR [16], NLF (Neural Localizer Fields) з неперервним полем локалізації [17], фундаментальна SMPLer-X на 32 датасетах [18] і спеціалізована модель кистей HaMeR [45], розглянута у підрозділі 1.5. Паралельно у лицевій анімації FaceFormer (2022) задала трансформерну схему поверх самонавчених аудіопредставлень [13].

У 2025–2026 роках напрям розвивають дифузійні моделі генерації руху, а NVIDIA опублікувала стек Audio2Face-3D – технічну публікацію і хаб з SDK (Software Development Kit), плагінами, NIM-сервісом (NVIDIA Inference Microservice) і попередньо навченими моделями з компонентними ліцензіями [42; 7]. За дев'ять років технологія дозріла, але інтеграція кількох модальностей в один інструмент залишається відкритою інженерною задачею.

1.4 Сучасні моделі оцінки тривимірної пози тіла

Підходи до оцінки тривимірної пози тіла зручно класифікувати за входами і виходами на три парадигми: двовимірна детекція ключових точок, ліфтинг 2D-поз у 3D-простір та наскрізна регресія параметричних моделей.

Двовимірні детектори ключових точок повертають координати (x, y) для 17–33 точок тіла. OpenPose (CMU, 2017) працює знизу вгору через Part Affinity

Fields і має академічну некомерційну ліцензію [3]; MediaPipe Pose Landmarker від Google повертає 33 орієнтири у координатах зображення і світових координатах та орієнтований на реальний час на клієнтських пристроях [6; 38]; AlphaPose, MoveNet і фреймворк MMPose доповнюють клас. Сильні сторони – швидкість і простота інтеграції; слабкі – відсутність тривимірної форми і неможливість відновити параметри SMPL для ретаргетингу, тому в сучасних системах 2D-детектори слугують або фінальним рішенням для плоских задач, або першим кроком двоетапного пайплайна.

Ліфтинг приймає послідовність 2D-точок і відновлює 3D-координати суглобів: VideoPose3D читає вікно з 243 кадрів дилатованою часовою згорткою [19], MotionBERT додає трансформер і вихід у параметри SMPL [20]. Підхід дешевий і повторно використовує готові 2D-детектори, але успадковує всі їхні помилки і деградує на складних кадрах з оклюзіями та рухомою камерою.

Наскрізна регресія параметрів SMPL / SMPL-X – найсильніша підгрупа: вхід – зображення або відео, вихід – параметри β і θ статистичної моделі тіла (SMPL з 24 кістками або SMPL-X з кистями і обличчям) разом з параметрами камери [2]. Розвиток покоління за поколінням: HMR (2018) – ResNet-50 з ітеративною регресією та адверсаріальним наглядом [4]; VIBE (2020) – часова узгодженість через GRU [11]; PARE (2021) – увага до частин тіла проти оклюзій [12]; HMR2.0 / 4D-Humans (2023) – ViT-H/16, MPJPE 70,0 мм (2.0a) і 81,3 мм (2.0b) на 3DPW при ≈15 FPS (Frames Per Second – кадрів за секунду) [14].

WHAM (2024) виводить рух у глобальну систему координат, поєднуючи кутову швидкість камери зі SLAM і контакти стоп: MPJPE 57,8 мм на 3DPW, W-MPJPE$_{100}$ (World MPJPE) 335,3 мм (еталонний гіроскоп) і 354,8 мм (DPVO, Deep Patch Visual Odometry) на EMDB (Electromagnetic Database – датасет глобальної 3D-пози і форми людини); код MIT, але залежності потребують окремої перевірки [15]. Multi-HMR (2024) – одноетапна модель усього тіла для кількох людей з виходом SMPL-X, проте з власною некомерційною ліцензією NAVER [16]. SMPLer-X (2023) – фундаментальна модель на ViT-Huge, навчена на 32 датасетах до 4,5 млн екземплярів; через обчислювальну вартість і ліцензії вона

слугує радше орієнтиром: для ролі базового модуля системи вона занадто вимоглива [18].

NLF (Neural Localizer Fields, 2024) використовує неперервне поле локалізації: природний вихід моделі непараметричний – тривимірні вершини і суглоби у запитаних точках поля. Перехід до параметрів SMPL-X виконує постпроцесингова бібліотека smplfitter того ж автора, яка підганяє параметричну модель Kabsch-оцінюванням орієнтацій частин тіла і лінійним методом найменших квадратів для форми зі швидкодією 9481 підгонка за секунду на RTX 3090 [43]. Код NLF і smplfitter мають MIT-ліцензію (Massachusetts Institute of Technology), ваги NLF – non-commercial research, файли SMPL-X – академічну ліцензію MPI (Max Planck Institute); для публічного розповсюдження це доведеться переглянути [17; 43].

Зведене порівняння наведено у Таблиці 1.1. Метрики різних статей вимірюють різне (MPJPE на 3DPW, PVE (Per-Vertex Error) на повнотілих бенчмарках, глобальна траєкторія), тому таблиця фіксує тип виходу, ліцензії і придатність до інтеграції, а не рейтинг точності.

Таблиця 1.1 – Порівняння методів регресії параметрів SMPL / SMPL-X

Метод	*Рік*	*Бекбон*	*Вихід*	*Глобальні координати*	*Авторський показник*	*FPS-клас*	*Ліцензія*
HMR	2018	ResNet-50	SMPL	–	–	> 30	MIT
VIBE	2020	ResNet + GRU	SMPL	–	–	> 30	MIT
PARE	2021	HRNet	SMPL	–	–	> 30	MIT
HMR2.0	2023	ViT-H/16	SMPL	–	3DPW MPJPE 70,0;81,3 [14]	≈ 15	MIT
WHAM	2024	ViT + SLAM	SMPL	+	3DPW MPJPE 57,8 [15]	≈ 30	MIT (код)
Multi-HMR	2024	ViT	SMPL-X	–	–	≈ 15	NAVER – некомерційне використання
NLF	2024	Transformer	3D-вершини /суглоби	–	–	≈ 10	NLF – non-commercial; smplfitter – MIT

SMPLer-X	2023	ViT-H	SMPL-X	–	UBody PVE 57,4 [18]	< 10	дослідницькі / проєктні умови використання

Із Таблиці 1.1 видно компроміс між типом виходу, швидкістю, глобальною системою координат і ліцензійним статусом. Multi-HMR привабливий як одноетапний метод усього тіла, але має некомерційні обмеження; WHAM сильний для глобальної траєкторії рухомої камери, але не закриває кисті й обличчя; NLF дає прийнятний баланс для дослідницького прототипу. NLF обрано основним обчислювальним модулем запропонованої системи з можливістю підключення альтернативних моделей у налаштуваннях. Додатковою перевагою NLF для цієї роботи виявився його непараметричний вихід для кистей: тривимірні точки кистей використовуються двічі – як грубе джерело поз пальців через SMPL-X-фітинг і як джерело двовимірних підказок для побудови кропів кистей, з якими далі працює спеціалізована модель. Моделі оцінки пози кистей розглянуто далі.

1.5 Моделі оцінки пози кистей

Кисті утворюють окрему підзадачу захоплення руху. При зйомці людини у повний зріст на 1080p на кисть припадає близько 50–80 пікселів; пальці постійно перекривають один одного, а кисть буває розвернута до камери ребром. Кінематична складність при цьому порівнянна з усім тілом: 15 суглобів на кисть, разом 90 ступенів свободи у поданні вісь-кут. Тому кисті мають найвищий рівень шуму серед усіх каналів, і якість їхньої оцінки найсильніше залежить від вибору моделі.

Спільним словником для подання пози кисті в сучасній літературі є параметрична модель MANO (hand Model with Articulated and Non-rigid defOrmations), опублікована Romero, Tzionas і Black у 2017 році [44]. MANO будується за тим самим принципом, що і SMPL: меш кисті з 778 вершин

описується функцією компактних векторів форми і пози, де поза задається локальними обертаннями 15 суглобів у поданні вісь-кут (15 × 3 = 45 чисел на кисть). Модель MANO інтегрована у SMPL-X як підмодель кистей, тому повнотіла модель і спеціалізована модель кисті можуть обмінюватися позами пальців без конвертації. Запропонована система спирається на цю властивість: канал кистей зберігає локальні пози пальців у форматі MANO незалежно від того, яка модель їх обчислила.

Перше джерело поз кистей – повнотілі регресори SMPL-X, розглянуті у підрозділі 1.4. NLF і Multi-HMR повертають пози пальців у складі повного тіла, тобто кисті отримуються «безкоштовно», без додаткових моделей і обчислень. Якість такої оцінки обмежена згаданою малою роздільністю кисті у кадрі повного тіла: пальці у виході повнотілих моделей зазвичай майже нерухомі або рухаються грубо, бо мережа фізично не бачить достатньо пікселів. Для сцен, де кисті не важливі, цього достатньо; для виразної жестикуляції – ні.

Друге джерело – спеціалізовані моделі, що працюють з вирізаним зображенням кисті (crop). Найвпливовіша з них – HaMeR (Hand Mesh Recovery), опублікована Pavlakos та ін. на CVPR 2024 [45]. HaMeR використовує великий Vision Transformer (ViT-H) і регресує параметри MANO безпосередньо з кропу кисті. Автори показали, що масштабування обсягу навчальних даних (об’єднання наявних датасетів з 2D- і 3D-розміткою кистей) і ємності мережі дає суттєвий приріст точності та стійкості порівняно з попередніми спеціалізованими моделями; додатково опубліковано датасет HInt з новою розміткою ключових точок кистей. Для роботи HaMeR потрібна попередня детекція кисті у кадрі – модель сама не шукає кисті.

Новішою альтернативою є WiLoR (Whole-image Localization and Reconstruction), опублікована Potamias та ін. і прийнята на CVPR 2025 [46]. WiLoR поєднує швидкий повнозгортковий детектор кистей і трансформерний реконструктор MANO в одному пайплайні, тобто усуває потребу у зовнішньому детекторі; для навчання детектора автори зібрали понад 2 млн зображень кистей у природних умовах. У запропонованій системі WiLoR розглянуто як

перспективну заміну HaMeR: інтерфейс каналу кистей від цього не змінюється, оскільки обидві моделі повертають пози у спільному словнику MANO. Порівняння джерел поз кистей наведено у Таблиці 1.2.

Таблиця 1.2 – Джерела поз кистей для системи захоплення руху

Джерело	*Рік*	*Вхід*	*Вихід*	*Роль у запропонованій системі*
Кисті у складі SMPL-X (NLF)	2024	повний кадр	пози пальців MANO у складі тіла	грубе базове джерело; резерв
HaMeR	2024	кроп кисті	MANO 15 × 3 на кисть + ключові точки	основне джерело пальців; потребує зовнішньої детекції кистей
WiLoR	2025	повний кадр	детекція кистей + MANO	перспективна заміна; в систему поки не інтегрована

Поєднання повнотілої і спеціалізованої моделей вимагає одного важливого інженерного рішення. Глобальна орієнтація зап’ястя, яку повертає crop-модель, визначена у системі координат кропу: модель не знає, де саме у кадрі та під яким кутом було вирізано кисть. Пряма заміна зап’ястя тіла орієнтацією з crop-моделі тому некоректна і призводить до вивернутих кистей. Коректна декомпозиція виглядає так: зап’ястя і передпліччя залишаються у каналі тіла, який оцінює їх у системі координат повного кадру, а зі спеціалізованої моделі беруться лише локальні пози пальців, інваріантні до положення кропу. Така декомпозиція й закладена у формат каналів системи (підрозділ 2.3).

Отже, основним джерелом пальців у системі обрано HaMeR, резервним і порівняльним – кисті з виходу NLF/SMPL-X. Обидва джерела пишуть канал у спільному словнику MANO, тому їх можна підміняти без зміни формату і споживачів. Це важливо для архітектури системи: вибір моделі пальців залишається рішенням producer-частини, а consumer бачить однаковий тип даних незалежно від джерела. Завдяки цьому можна оцінювати якість джерел пальців

без переписування адаптера Blender або алгоритмів синхронізації. Така взаємозамінність створює природний матеріал для експериментального порівняння джерел пальців у розділі 3.3.

1.6 Сучасні моделі захоплення лицевих рухів

Лицева анімація у системі формується двома незалежними джерелами: візуальним аналізом обличчя у кадрі та аудіокерованою регресією рухів губ; тут розглянуто перше, аудіокеровані моделі винесено у підрозділ 1.7.

Моделі лицевого захоплення видають один з двох виходів: орієнтири обличчя (точки на куточках очей, контурі губ тощо) або коефіцієнти блендшейпів. Перший вихід зручний для аналізу, але потребує конвертації для анімації; другий готовий до подачі на риг, тому індустрія рухається до моделей з прямим виходом у блендшейпи, найчастіше у форматі ARKit-52, пов’язаному з Action Units системи FACS [5; 10].

Основним візуальним джерелом системи обрано MediaPipe Face Landmarker від Google: компонент повертає сітку обличчя з 478 тривимірних орієнтирів, оцінки 52 блендшейпів та матрицю трансформації обличчя, з якої виводиться наближений поворот голови [37]. Переваги – готові пакети для Python та інших платформ, реальний час на клієнтських пристроях, дозвільні ліцензії прикладів коду; обмеження – деградація при сильних поворотах голови, рідкісних виразах і оклюзіях.

Вищу геометричну точність дають моделі на основі 3D Morphable Model. Базовою є FLAME (Faces Learned with an Articulated Model and Expressions, 2017): лінійна модель обличчя на 5023 вершини з розділеними параметрами ідентичності (300 компонент), виразу (100) і пози шиї та щелепи [21]. На FLAME побудовано родину регресорів: DECA (Detailed Expression Capture and Animation, 2021) додає карту зміщень для деталей шкіри [51], EMOCA (Emotion Driven Monocular Face Capture and Animation, 2022) – втрату емоційної узгодженості

[22], MICA (2022) – метричну точність ідентичності [23], SPECTRE (2022) – втрату читання губ для кращої реконструкції рота під час мовлення [24]. Ціна – конвертація FLAME-параметрів у блендшейпи і здебільшого академічні ліцензії.

Вибір MediaPipe замість 3DMM-регресора зумовлений прямим виходом у ARKit-коефіцієнти без конвертації, простотою інтеграції і ліцензіями, сумісними з прототипом [37]. Моделі з RGB-D-входом (RGB + Depth – кольорове зображення з картою глибини), як-от Kinect чи TrueDepth, виключено з розгляду через вимогу працювати з побутовою камерою без додаткових сенсорів; 3DMM-режим високої якості залишено як майбутнє розширення.

1.7 Аудіокерована анімація обличчя

Візуальний аналіз обличчя ненадійний на бічних ракурсах, у поганому освітленні та при низькій роздільності – саме там, де рот найважливіший. Аудіопотік натомість містить артикуляцію у явному вигляді: відповідність між фонемами і формами губ (віземи) дозволяє відновлювати рухи рота безпосередньо з мовлення. На цій відповідності ґрунтується напрям аудіокерованої лицевої анімації.

Модель цього класу приймає аудіопотік або аудіоознаки і видає параметри лицевої анімації: зміщення вершин сітки обличчя, FLAME-параметри або коефіцієнти блендшейпів. Для запропонованої системи найважливіший вихід – підмножина ARKit-коефіцієнтів щелепи і рота (jawOpen, mouthFunnel, mouthPucker, mouthSmile та інші), яка далі поєднується з відеоканалом алгоритмом регіонального злиття з розділу 2.

Напрям розвивався швидко. VOCA (Voice Operated Character Animation, MPI, 2019) – згорткова мережа, що відображає аудіоознаки у вершинні зміщення FLAME-сітки, навчена на 4D-сканах датасету VOCASET [25]. MeshTalk (2021) увів крос-модальне розділення рухів, пов’язаних і не пов’язаних з мовленням, що

дало реалістичні моргання у паузах [26]. FaceFormer (2022) став поворотною точкою: замість ручних аудіоознак – самонавчене представлення wav2vec 2.0 [47] з трансформерним декодером перехресної уваги [13]. CodeTalker (2023) додав дискретну апріорну модель руху на VQ-VAE (Vector-Quantized Variational Autoencoder) проти надмірного згладжування [27]; EmoTalk (2023) – явне керування емоцією мовлення [28]; SadTalker (2023) генерує відео обличчя з однієї фотографії і свідчить про зрілість напрямку [29].

NVIDIA Audio2Face-3D (2025) – офіційний стек генерації тривимірної лицевої анімації з аудіо: технічна публікація [42] і хаб з SDK, плагінами, тренувальним фреймворком, контейнерним NIM-сервісом і попередньо навченими моделями. Ліцензії компонентні: SDK і плагіни – MIT, тренувальний фреймворк – Apache, моделі – NVIDIA Open Model і власні умови [7].

Легший клас рішень – фонемні віземні генератори: фонемний розпізнавач плюс фіксована таблиця відповідності фонем до візем, розгорнутих у ARKit-підмножину рота. Якість нижча за повні моделі, зате не потрібен GPU-сервіс (Graphics Processing Unit – графічний процесор); для системи це технічний резерв на випадок недоступності повної моделі.

Основним аудіоджерелом системи обрано Audio2Face-3D: модель розгортається локально як контейнерний сервіс, приймає аудіодоріжку і повертає ARKit-сумісні коефіцієнти разом з оцінками емоційного стану; власне донавчання не передбачене – всі моделі використовуються як готові. Архітектурна вимога до будь-якого аудіокомпонента формулюється через покриття: видавати ARKit-сумісний шар на стабільній власній частоті (для Audio2Face-3D – 30 Гц) і явно заявляти довіру до кожного регіону обличчя. Повна модель впевнено покриває щелепу і рот та слабше – брови, щоки, ніс; віземний генератор – лише рот і щелепу. Споживачеві достатньо вектора довір за регіонами, формалізованого у підрозділі 2.2.

1.8 Розпізнавання емоцій з аудіо та відео

Емоція у системі – окремий канал, що накладається поверх базової міміки. Мотивація: однакові послідовності коефіцієнтів рота можуть супроводжувати і веселу, і сумну розмову, а різниця лежить у бровах, очах і тонусі щік. Винесена в окремий канал, ця різниця залишається редагованою і вимиканою без впливу на решту анімації.

Емоцію представляють двома способами: категоріально за Екманом (6 базових емоцій плюс нейтральний стан, softmax-розподіл за 7 класами) і неперервно за циркумплексною моделлю афекту Рассела [48] – скалярами Valence, Arousal, Dominance у діапазоні від −1 до +1. Система зберігає обидва подання: канал емоції містить 7 класових ймовірностей і 3 скаляри VAD (Valence–Arousal–Dominance), які за відсутності окремого регресора виводяться з класів фіксованою таблицею відповідності.

У локальному для КПІ/ІПСА науковому контексті близькою є робота Бодянського, Зайченка, Гамідова і Кулішової: багатошарова GMDH (Group Method of Data Handling) нейро-нечітка мережа на розширених нео-нечітких нейронах для потокового розпізнавання мімічних виразів [33]. Як базовий компонент метод не обрано – він передбачає побудову і навчання власної моделі, тоді як робота спирається на готові попередньо навчені модулі; водночас він підтверджує інженерну вимогу до емоційного каналу: потрібен легкий потоковий модуль, придатний до покадрової роботи.

Джерелом емоції у реалізованій системі є аудіо: Audio2Face-3D разом з лицевими коефіцієнтами повертає оцінки емоційного стану, які переводяться у 7 канонічних класів. Резервним класом моделей є wav2vec2-SER (Speech Emotion Recognition) – донавчені версії wav2vec 2.0 з класифікаційною головою; придатність цих представлень для розпізнавання емоцій показана у [30], хоча конкретна точність залежить від датасету і схеми донавчання. Аудіоджерело

незалежне від ракурсу й освітлення; його чутливість до доменного зсуву компенсується м’якістю ймовірностей на боці споживача.

Відеоканал емоцій у фінальну конфігурацію не увійшов: розглянута бібліотека HSEmotion [31] частково дублює інформацію каналу обличчя, а кожна додаткова модель збільшує час обробки на побутовому GPU. Архітектура до другого джерела готова – споживачі вибирають канали за роллю emotion, і другий канал не порушує ані формату, ані алгоритмів. Для майбутнього поєднання джерел застосовна таксономія мультимодального злиття Atrey та співавторів (раннє, пізнє, гібридне) [32]; природним для незалежних каналів є пізнє злиття розподілів класів.

Частоту оновлення формат не нав’язує: канал зберігається з частотою продюсера, а узгодження з кадрами сцени виконує синхронізація споживача. Емоційний стан змінюється повільно порівняно з кадровою частотою відео, тому високої частоти дискретизації канал не потребує; в експериментальній конфігурації він успадковує 30 Гц виходу Audio2Face-3D.

1.9 Готові програмні рішення та їхні обмеження

Ніша єдиного інструмента для всіх модальностей залишається порожньою: кожен відомий відкритий проєкт покриває одну-дві модальності з чотирьох і має обмеження за ретаргетингом, форматом обміну або ліцензією.

FreeMoCap [8] – Python-фреймворк багатокамерного захоплення тіла з тріангуляцією 2D-точок від 2–4 вебкамер; монокулярний режим через MediaPipe Pose деградує за якістю. Експортує у Blender через Rigify і VRM (формат гуманоїдних 3D-аватарів); обличчя у форматі ARKit, звук і емоції відсутні. Ліцензія AGPL-3.0 (GNU Affero General Public License).

BlendArMocap [39] – доповнення до Blender, що в реальному часі знімає тіло, кисті й обличчя з MediaPipe на риги Rigify та MHX. Звук і емоції відсутні;

ретаргетинг без компенсації нейтральної пози і масштабування кісток. Ліцензія GPL-3.0 (GNU General Public License).

ThreeDPoseTracker [40] – Unity-застосунок під сценарії VTuber (Virtual YouTuber): лише тіло, експорт у VRM, режим некомерційного використання. MocapNET [41] – швидке 2D→3D підняття поверх OpenPose з виходом у BVH (Biovision Hierarchy): лише тіло, ліцензія FORTH (Foundation for Research and Technology – Hellas) плюс академічна ліцензія OpenPose.

NVIDIA Audio2Face-3D [7] закриває протилежну частину спектра: обличчя й аудіо без тіла, з офіційною підтримкою вендора та інтеграцією з Unreal/MetaHuman, але з NVIDIA-орієнтованою інфраструктурою і компонентними ліцензіями. OpenPose та приклади MediaPipe є будівельними блоками: вони повертають числові масиви і не містять пайплайна, експорту чи інтерфейсу кінцевого продукту.

Зведений огляд наведено у Таблиці 1.3. Знак «+» означає підтримку модальності, знак «–» означає відсутність.

Таблиця 1.3 – Зведений огляд відкритих рішень AI-захоплення

Проєкт	*Тіло*	*Обличчя*	*Аудіо*	*Емоція*	*Цільові риги*	*Ліцензія*
FreeMoCap	+	–	–	–	Rigify, VRM	AGPL-3.0
BlendArMocap	+	+	–	–	Rigify, MHX	GPL-3.0
ThreeDPoseTracker	+	–	–	–	VRM (Unity)	некомерційне використання
MocapNET	+	–	–	–	BVH	ліцензія FORTH
NVIDIA A2F-3D	–	+	+	+ (емоція з аудіо)	MetaHuman	компонентні
OpenPose	+ (2D)	–	–	–	–	академічна
приклади MediaPipe	+	+	–	–	–	Apache 2.0

Аналіз розривів. З Таблиці 1.3 видно, що:

- жодне з відкритих рішень не підтримує всі чотири модальності одночасно (тіло + обличчя + аудіо + емоція);

- жодне з рішень не зберігає проміжні результати окремих моделей: усі вони експортують єдину уніфіковану позу, в якій джерела сигналу вже невіддільні;
- жодне з рішень не дає користувачеві керованого механізму поєднання візуального та аудіоджерела лицевої анімації;
- жодне з рішень не має явного інтерфейсу для діагностики окремих каналів захоплених даних перед застосуванням до персонажа.

Комерційні сервіси (Rokoko Vision, DeepMotion Animate3D, Move.ai, iClone Motion Live, Apple Live Link Face, RADiCAL Motion) працюють за підпискою через хмару, з пропрієтарними протоколами і без локального розгортання, що для академічних сценаріїв є суттєвим обмеженням.

Розрив, отже, лежить глибше за перелік підтримуваних модальностей. Жодне з розглянутих рішень не зберігає проміжні канали спостережень у відкритому форматі, придатному для незалежного перегляду, вимикання й заміни окремих джерел. Запропонована система заповнює цей розрив форматом .aimocap і поділом відповідальності між продюсером і споживачем даних. Далі формалізуємо постановку задачі, що випливає з цього аналізу.

1.10 Постановка задачі

Проведений аналіз показує, що сучасні нейромережеві моделі вже здатні оцінювати окремі складові руху людини з доступних джерел даних, однак практична задача створення повної анімації персонажа не зводиться до запуску однієї моделі. Необхідно узгодити тіло, кисті, обличчя, аудіокеровану артикуляцію рота та емоційний шар у спільній часовій шкалі, не втрачаючи при цьому можливості перевірити якість кожного джерела окремо. Тому постановка задачі цієї роботи полягає у розробленні інтелектуальної багатоканальної системи AI-захоплення руху, яка приймає монокулярне відео з синхронною

звуковою доріжкою та обраним користувачем діапазоном кадрів, а на виході формує придатний до повторного відкриття і практичного застосування файл спостережень руху.

Метою дослідження є підвищення ефективності та доступності процесу створення анімації 3D-моделі людини; її досягнення забезпечують архітектура і програмна реалізація системи, що послідовно запускає набір попередньо навчених моделей за основними модальностями руху, виконує поканальне очищення їхніх результатів і зберігає ці результати у самоописному форматі .aimocap. Замість однієї остаточно змішаної анімації формат має фіксувати окремі канали спостережень з часовими мітками, покадровими довірами, описом джерел, діапазоном експорту і рекомендованою політикою використання. Така організація потрібна для того, щоб редактор, експериментальний стенд або доповнення Blender могли незалежно читати ті самі дані, синхронізувати їх до власної частоти кадрів, змінювати ваги злиття і вимикати окремі джерела без повторного запуску всього конвеєра.

Вихідним матеріалом для системи вважається записаний відеофайл побутової камери з однією людиною в кадрі, синхронним аудіо та достатньою роздільною здатністю для виділення обличчя і кистей. Система не повинна вимагати маркерів, інерційного костюма, багатокамерної сцени або спеціально обладнаного приміщення. Обробка розглядається як офлайн-процес над вибраним фрагментом запису; вимога реального часу не ставиться. Окремим обмеженням є робота на побутовій робочій станції з одним GPU класу NVIDIA RTX з 16 ГБ відеопам’яті, тому важкі моделі мають запускатися послідовно, а повторне використання результатів має забезпечуватися кешуванням етапів.

Для досягнення поставленої мети необхідно сформувати канальну модель декомпозиції, у якій тіло, пальці, поза голови, відео-міміка, аудіо-міміка та емоція є незалежними спостереженнями з власними довірами і частотами дискретизації. На основі цієї моделі потрібно спроєктувати формат .aimocap, який зберігає дані як маніфестований контейнер, а також формалізувати producer-алгоритми калібрування, очищення і кодування та consumer-алгоритми

декодування, синхронної вибірки, регіонального злиття лицевих каналів, емоційного шару і контракту вендорського адаптера. Практична частина задачі передбачає реалізацію бібліотеки aimocap, настільного редактора на Qt 6 для керування обробкою і діагностики каналів та тестового доповнення Blender, яке демонструє можливість застосувати збережені дані до підготовленого персонажа DAZ Genesis.

Експериментальна перевірка має показати не лише працездатність кінцевого результату, а й внесок окремих компонентів системи. Для цього один і той самий тестовий фрагмент обробляється з послідовним увімкненням каналів і порівнянням альтернативних джерел: тіло без спеціалізованих пальців і з HaMeR, міміка лише з відео і з додаванням аудіоканалу, базова міміка без емоційного шару і з ним. Такий стенд дозволяє обґрунтувати практичну цінність багатоканального підходу порівняно з моно-модельним пайплайном, а також виміряти час обробки, ефект кешування, розмір файлу і межі застосовності системи.

Практична новизна роботи зосереджена в організації даних і поділі відповідальності між producer- та consumer-частинами. Нейромережеві моделі використовуються як готові компоненти, але їхні виходи не зливаються безповоротно всередині конвеєра. Файл .aimocap зберігає канали у відкритій для аналізу структурі, а остаточне поєднання виконується там, де дані застосовуються. Завдяки цьому система може бути використана як дослідницький інструмент для незалежних розробників, аніматорів, освітніх лабораторій і робіт з аналізу людино-комп’ютерної взаємодії.

Висновки до розділу 1

У розділі проаналізовано задачу AI-захоплення руху, історію переходу від маркерних і інерційних систем до безмаркерного нейромережевого захоплення

та сучасні моделі за п’ятьма напрямками: поза тіла, поза кистей, лицеві рухи, аудіокерована анімація обличчя, розпізнавання емоцій.

Обрано компоненти системи. Тіло: NLF з постпроцесингом smplfitter у SMPL-X-параметри – компромісний баланс точності, швидкості і ліцензій серед розглянутих регресорів (HMR, VIBE, PARE, HMR2.0, WHAM, Multi-HMR, SMPLer-X). Кисті: спільним словником є модель MANO; основне джерело пальців – crop-модель HaMeR, резервне – кисті з виходу NLF, перспективна заміна – WiLoR; зап’ястя в усіх конфігураціях залишається за каналом тіла. Обличчя: MediaPipe Face Landmarker з прямим виходом у 52 коефіцієнти; 3DMM-моделі відкладено як режим високої якості. Аудіо-рот і емоція: NVIDIA Audio2Face-3D як локальний контейнерний сервіс з компонентними ліцензіями; емоційний канал походить з аудіо, відеорозпізнавання і мультимодальне злиття емоцій – напрями розширення.

Огляд відкритих рішень (FreeMoCap, BlendArMocap, ThreeDPoseTracker, MocapNET, Audio2Face-3D) показав, що жодне не покриває чотири модальності в одному пайплайні і жодне не зберігає проміжні канали спостережень у відкритому форматі, придатному для незалежного редагування.

Сформульовано постановку задачі: вхід, шукану архітектуру, обмеження і підзадачі реалізації. З неї випливає структура розділу 2 – канальна модель декомпозиції, словники даних, формат обміну .aimocap, producer- і consumer-алгоритми.

РОЗДІЛ 2 АЛГОРИТМИ ТА АРХІТЕКТУРА ЗАПРОПОНОВАНОЇ СИСТЕМИ

2.1 Канальна модель декомпозиції та архітектура

Архітектурна ідея системи зводиться до однієї тези: кожна модальність обробляється як незалежний канал спостережень з власною моделлю-джерелом, частотою, складом запису і мірою впевненості. Канали взаємно незалежні; їхня взаємодія відбувається лише на боці споживача у момент застосування до персонажа. Це дає змогу замінювати окрему модель без зміни решти конвеєра: наприклад, джерело пальців може перейти з HaMeR на WiLoR, а лицевий канал – з MediaPipe на 3DMM-регресор, якщо вони записують дані в узгоджений канал. Також така схема спрощує діагностику, бо помилки можна локалізувати не в усьому результаті одразу, а в конкретному каналі: тілі, кистях, обличчі, аудіо-роті або емоціях. Монолітна альтернатива – єдина мережа всіх модальностей – дала б теоретично кращу якість за рахунок кросмодальних залежностей, але вимагала б власного навчання на десятках тисяч годин розмічених записів; канальна декомпозиція натомість повторно використовує готові опубліковані моделі.

Система зберігає шість каналів спостереження: body.pose (поза тіла), body.hands (пози пальців), head.pose (поворот голови), face.blend.video і face.blend.audio (лицеві коефіцієнти з відео та аудіо), emotion.audio (емоційний стан). Сьомий, діагностичний канал body.hands.debug містить службові дані ланцюга кистей і споживачами анімації ігнорується. Кожен канал має власні часові мітки у секундах від початку кліпу; узгодження осей виконує синхронізація споживача (підрозділ 2.6).

Канали належать до п’яти ролей: body, hands, head, face, emotion. Споживачі вибирають канали за роллю; ім’я каналу слугує лише ідентифікатором. Звідси розширюваність: другий аудіоканал обличчя наявний

споживач опрацює автоматично, а канал з невідомою роллю безпечно пропустить з попередженням.

Розподіл відповідальності показано на рисунку 2.1: producer (бібліотека aimocap) запускає моделі, очищає виходи і кодує канали у файл; consumer (редактор і доповнення Blender) декодує, синхронізує і зливає. Межею відповідальності слугує сам файл.

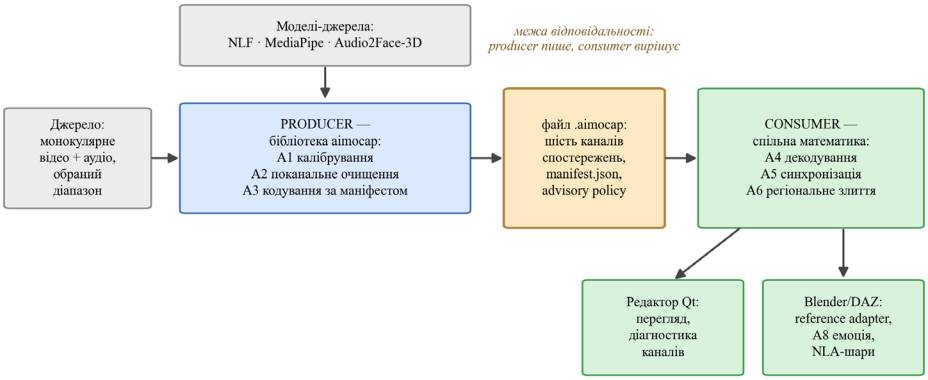

Рисунок 2.1 – Архітектура запропонованої системи

Другою опорою є шарова модель даних (рисунок 2.2). L0 – сирі виходи моделей: у файл не потрапляють, відтворюються повторним запуском з локальним кешем. L1 – очищені спостереження каналів тіла, кистей, голови та відео-обличчя. L2 – похідна семантика: аудіоканали і вектори довір за регіонами. L1 і L2 утворюють вміст файлу. Такий поділ фіксує межу відповідальності: producer відповідає за отримання й очищення спостережень, але не приймає остаточне рішення про їхнє поєднання на конкретному персонажі. Завдяки цьому один і той самий .aimocap файл можна повторно відкрити з іншою політикою споживача, іншим профілем рига або іншим алгоритмом злиття без повторного запуску важких моделей. L3 – синхронізація і злиття у споживача; L4 – зафіксована (бейк) анімація на ризі.

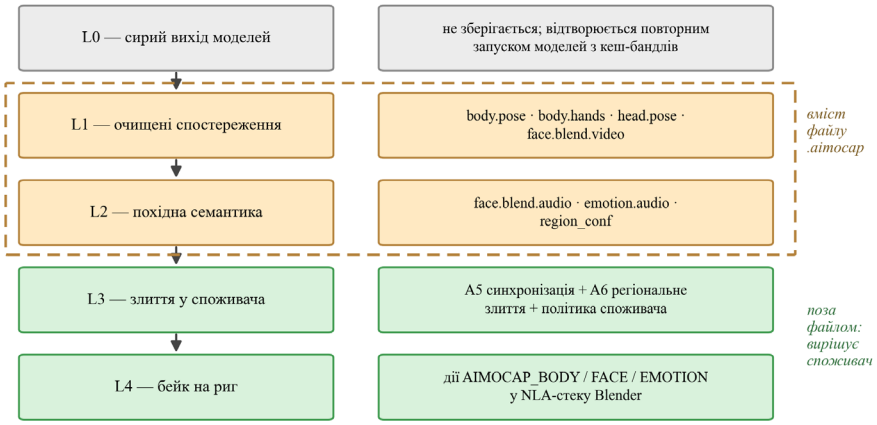

Рисунок 2.2 – Шарова модель даних

Отже: файл зберігає тільки спостереження. Необоротного об’єднання даних до запису не відбувається, тому зміна політики споживача (вимкнути аудіоканал, порівняти джерела пальців) не потребує повторного запуску моделей. Розділ 3 використовує цю властивість прямо: всі порівняльні експерименти виконуються над одним файлом зміною параметрів застосування.

Склад каналів відображає межі між підзадачами: голова відділена від міміки (кістковий ланцюг проти FACS-контролерів), кисті від тіла (окреме джерело і найвищий шум), аудіо-обличчя від відео-обличчя (різні модальності з різними зонами надійності), емоція від міміки (адитивний шар, який вимикається без сліду).

2.2 Існуючі параметричні моделі тіла й обличчя та регіональна модель ARKit-52

Канали системи спираються на три словники даних: параметричну модель тіла SMPL-X для каналу тіла, параметричну модель кисті MANO для каналу пальців і набір з 52 коефіцієнтів ARKit для лицевих каналів. Цей підрозділ фіксує

всі три і вводить регіональну модель обличчя, на якій будуються формат (підрозділ 2.3) і регіональне злиття (підрозділ 2.7).

SMPL (Skinned Multi-Person Linear Model) – параметрична модель людського тіла, опублікована Max Planck Institute for Intelligent Systems у 2015 році [2]. Модель описує форму та позу людини як параметричну функцію над сіткою з 6890 вершин і 24 кістками:

$$M(\beta, \theta) = W\left(T_p(\beta, \theta), J(\beta), \theta, W_{LBS}\right), \quad (2.1)$$

де T_p – шаблонна сітка з урахуванням деформацій форми і пози; $\beta \in R^{10}$ – вектор форми тіла (10 компонент PCA-розкладу, Principal Component Analysis); θ – параметри пози у поданні вісь-кут; $J(\beta)$ – регресор позицій суглобів залежно від форми; W – оператор лінійного скінінгового змішування з вагами W_{LBS}.

У спрощеному викладі модель (2.1) приймає 10 чисел форми і 72 числа пози та повертає 6890 координат вершин – повний меш тіла. SMPL-X розширює SMPL до повного тіла з кистями та обличчям (54 суглоби) [36]; кисті у SMPL-X – вбудована модель MANO [44], тому пози пальців SMPL-X і спеціалізованої моделі HaMeR записані в одному словнику. NLF у системі дає непараметричні 3D-вершини і суглоби, з яких SMPL-X-параметри відновлює бібліотека smplfitter (Kabsch-оцінювання орієнтацій, найменші квадрати для форми) [43]; цей крок – інтеграція готової MIT-бібліотеки.

Канал body.pose зберігає з SMPL-X-вектора лише тіло: 63-вимірну позу (21 суглоб × 3), кореневу трансляцію, кореневий кватерніон і вектор форми β. Пози шиї і голови у SMPL-X формально присутні, але споживач їх ігнорує – за голову відповідає канал head.pose від MediaPipe, який оцінює поворот голови надійніше за повнотілу модель. Пози пальців винесені у body.hands (по 45 чисел на кисть у поданні MANO). Вектор β зберігається покадрово; рішення про його колапс до константи винесене у підказку політики betas_export і ухвалюється споживачем. Модель обличчя FLAME [21] безпосередньо не використовується: лицеві канали працюють у словнику ARKit-52.

Новим елементом, введеним у цій роботі, є регіональна модель ARKit-52. Усі 52 коефіцієнти розбито на сім анатомічних регіонів; розбиття повне і без перетинів, що закріплено окремим тестом. Склад регіонів наведено у Таблиці 2.1.

Таблиця 2.1 – Регіональне розбиття 52 коефіцієнтів ARKit

Регіон	*Кількість коефіцієнтів*	*Приклади коефіцієнтів*
brow (брови)	5	browDownLeft, browInnerUp, browOuterUpRight
eyes (очі)	14	eyeBlinkLeft, eyeLookUpRight, eyeSquintLeft
cheek (щоки)	3	cheekPuff, cheekSquintLeft, cheekSquintRight
nose (ніс)	2	noseSneerLeft, noseSneerRight
jaw (щелепа)	4	jawOpen, jawForward, jawLeft, jawRight
mouth (рот)	23	mouthSmileLeft, mouthPucker, mouthFunnel, mouthClose
tongue (язик)	1	tongueOut

Кожен лицевий канал зберігає поряд із 52 коефіцієнтами вектор region_conf із семи чисел – довіру каналу до кожного регіону в поточному записі. Семантика цього вектора навмисно об’єднує два поняття: покриття і надійність. Модель, яка взагалі не оцінює певний регіон, ставить для нього нульову довіру; модель, яка оцінює регіон, але в поточному кадрі бачить його погано, ставить низьку довіру. Завдяки цьому формату не потрібні окремі канали для «повних» і «часткових» моделей обличчя: віземний генератор, що дає лише рот і щелепу, заповнює ненульову довіру тільки для цих двох регіонів, і той самий алгоритм злиття коректно працює з обома випадками. Приклад заповнення region_conf для двох фактичних джерел системи показано на рисунку 2.3.

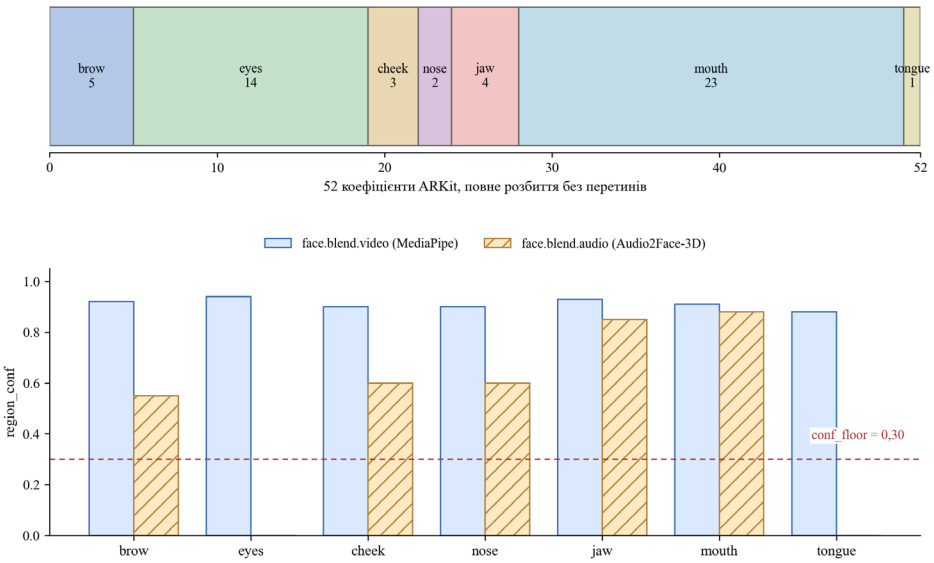

Рисунок 2.3 – Регіональне розбиття ARKit-52 і приклад векторів region_conf двох лицевих каналів

На регіональну модель спираються три подальші елементи роботи: формат (region_conf входить до складу запису лицевих каналів), регіональне злиття (підрозділ 2.7 обчислює ваги змішування саме за регіонами) та експериментальна перевірка (розділ 3 демонструє деградацію і відновлення довір на стрес-прикладах).

2.3 Формат обміну .aimocap

Стандартні формати обміну анімацією (FBX, BVH) зберігають одну уніфіковану позу на кадр без походження сигналу і довір джерел. Для шести різночастотних каналів спостережень, які мають залишатися незалежно вимиканими і замінними, потрібен власний формат.

Формат .aimocap – контейнер zstd(tar): tar-архів, стиснений алгоритмом Zstandard [50]. Першим членом іде manifest.json, який повністю описує вміст, тому читач дізнається структуру файлу без повного розпакування; далі – бінарні

файли каналів у каталозі channels/ та опційні артефакти. Структуру контейнера показано на рисунку 2.4.

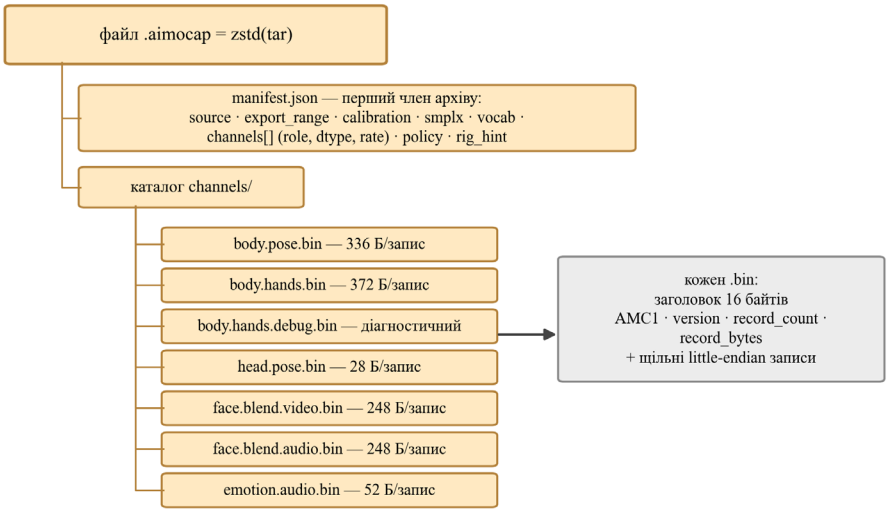

Рисунок 2.4 – Структура контейнера .aimocap

Кожен бінарний файл каналу починається 16-байтовим заголовком: магічна сигнатура AMC1, мінорна версія, кількість записів record_count і розмір запису record_bytes. Після заголовка йдуть щільно упаковані записи у порядку little-endian. Інваріант record_bytes = itemsize типу запису перевіряється і при кодуванні, і при декодуванні; розбіжність означає пошкоджений або несумісний файл.

Самоописність формату забезпечує маніфест: для кожного каналу він містить повний опис типу запису (імена полів, типи, розміри), роль, модальність, продюсера, шар і частоту. Читач будує структуру запису безпосередньо з маніфесту, без відомостей, зашитих у код споживача, тому додавання нового каналу не вимагає оновлення споживачів: канали з невідомою роллю пропускаються з попередженням. Файли старої розкладки (з окремим заголовковим файлом замість маніфесту) декодер відхиляє з явним повідомленням про потребу повторної генерації. Склад записів кожного каналу наведено у Таблиці 2.2.

Таблиця 2.2 – Канали формату .aimocap

Канал	*Роль*	*Склад запису*	*Байтів на запис*
body.pose	body	t (f8); conf (f4); betas(10); root_t(3); root_q(4); body_pose(63)	336
body.hands	hands	t; conf; left(45); right(45)	372
body.hands.debug	hands (діагностичний)	t; conf; success(2); crop_boxes(2×4); keypoints_2d(2×21×2); keypoints_3d(2×21×3)	892
head.pose	head	t; conf; head_q(4)	28
face.blend.video	face	t; blendshapes(52); region_conf(7)	248
face.blend.audio	face	t; blendshapes(52); region_conf(7)	248
emotion.audio	emotion	t; conf; classes(7); vad(3)	52

Маніфест, окрім описів каналів, містить блоки метаданих, перелічені у Таблиці 2.3. Особливе місце серед них займає блок policy – рекомендована політика споживача.

Таблиця 2.3 – Основні блоки manifest.json

Блок	*Призначення*
format, version	ідентифікація формату і версії для перевірки сумісності
source	шлях, розміри кадру, частота кадрів, тривалість, параметри аудіодоріжки джерела
export_range	межі обробленого діапазону у секундах та основна частота кадрів
calibration	матриця переходу до правоорієнтованого базису цільового середовища
smplx	rest-метадані моделі тіла: позиції суглобів спокою, відстань таз–голова
vocab	вбудований словник ARKit-52 і регіональне розбиття
channels[]	роль, модальність, продюсер, шар, частота, кількість записів, опис типу запису
policy	рекомендована політика споживача (Таблиця 2.4)
rig_hint, editor	підказка цільового рига та стан редактора для повторного відкриття

Політика у маніфесті має суто рекомендаційний характер: вона ніколи не застосовується до збережених значень, і споживач може перевизначити кожен її параметр. Так формально закріплено головну тезу роботи: продюсер записує, як він радить поєднувати канали, а саме поєднання відбувається лише в момент застосування. Параметри політики наведено у Таблиці 2.4.

Таблиця 2.4 – Параметри рекомендованої політики споживача

Параметр	*Призначення*	*Значення за замовчуванням*
conf_floor	поріг довіри, нижче якого регіон вважається непокритим	0,30
face_fusion_audio_weight	вектор ваг аудіоканалу за сімома регіонами обличчя	див. Таблицю 2.5
emotion_weight	глобальна вага адитивного емоційного шару	0,5
root_motion	режим кореневого руху: locked / planar / full	locked
betas_export	підказка згортання покадрової форми до константи	median_confident
hand_conf_floor	поріг довіри кистей для показу у попередньому перегляді	0,90

Файл .aimocap виконує у системі три ролі. Перша – кеш повторної обробки: редактор після завершення конвеєра читає лише декодований файл, а повторне відкриття відновлює діапазон і стан таймлайна з блоку editor. Друга – формат обміну: файл є єдиним входом доповнення Blender. Третя – одиниця відтворюваності експерименту: всі порівняльні експерименти розділу 3 виконуються над зафіксованими файлами, що робить кожен результат повторюваним. Фактичні розміри файлу і кількості записів на реальному тестовому кліпі наведено у підрозділі 3.2.

2.4 Повторно використовувані алгоритмічні примітиви

Перед описом алгоритмів коротко зафіксуємо примітиви, що використовуються у кількох місцях; подається лише сигнатура і місце застосування. Лінійне скінінгове змішування (LBS) деформує меш за рухом скелета зваженою сумою трансформацій кісток; працює всередині SMPL-X та у Blender, власної реалізації система не містить. Пряма кінематика (FK) поширює локальні обертання від кореня дерева кісток до листя і використовується вендорськими адаптерами під час застосування пози до конкретного рига (підрозділ 2.8).

Сферична лінійна інтерполяція кватерніонів (SLERP) [49]. Інтерполяція між двома одиничними кватерніонами q_a і q_b з параметром $\alpha \in [0, 1]$:

$$\mathrm{SLERP}(q_a, q_b, \alpha) = \frac{[\sin((1-\alpha)\Omega)\, q_a + \sin(\alpha\Omega)\, q_b]}{\sin(\Omega)}, \qquad (2.2)$$

де Ω – кут між кватерніонами, визначений їхнім скалярним добутком.

Покомпонентна лінійна інтерполяція кватерніонів, на відміну від (2.2), дала б змінну кутову швидкість і артефакти прискорень; SLERP гарантує сталу. У системі SLERP застосовується в синхронізації (підрозділ 2.6) для кватерніонних полів і, окремо для кожного суглоба, для поворотів вісь-кут, які переводяться у кватерніонний домен експоненціальним відображенням і назад.

Вирівнювання знаків кватерніонної послідовності (sign-unroll). Кватерніони q і $-q$ кодують одне обертання, тому покадрова послідовність може стрибати між еквівалентними поданнями. Перед покомпонентним згладжуванням кожен подальший кватерніон за потреби множиться на -1, щоб скалярний добуток із попереднім був невід'ємним; після згладжування компоненти ренормалізуються. Неперервність результату закріплена тестом.

Заповнення коротких розривів за довірою. Записи з довірою нижче порога вважаються відсутніми; розриви тривалістю до 0,5 с заповнюються лінійною

інтерполяцією між надійними краями. Довші розриви не заповнюються: вигадані дані на великих інтервалах дали б правдоподібну на вигляд, але хибну анімацію.

Фільтр Савицького–Голея. Згладжування локальним підгоном полінома низького степеня у вікні фіксованого розміру [35]; на відміну від ковзного середнього зберігає локальні піки виразних рухів. В офлайн-системі з доступом до всього діапазону симетричне вікно дає кращий результат за каузальні фільтри інтерактивних застосувань на кшталт One-Euro [34], тому конвеєр очищення обмежено єдиним офлайн-проходом.

Пальцеві ланцюги тестового рига наближаються послідовністю шарнірних суглобів, кути яких обчислюються у замкненій формі з цільових позицій суглобів MANO, без ітеративного розв’язувача (підрозділ 2.8).

2.5 Producer-алгоритми: калібрування, очищення, кодування

Producer-частина складається з трьох алгоритмів, що виконуються один раз під час обробки кліпу; далі вони позначаються A1–A3, а consumer-алгоритми, подані нижче, позначаються A4–A8.

A1 – калібрування. Моделі-джерела працюють у власних, здебільшого лівоорієнтованих екранних системах координат; Blender використовує правоорієнтовану з віссю Z догори. Алгоритм будує матрицю переходу до правоорієнтованого базису з визначником +1 (що унеможливлює віддзеркалення персонажа) і записує її у блок calibration маніфесту разом з rest-метаданими тіла: позиціями суглобів у позі спокою і відстанню таз–голова. До збережених даних матриця не застосовується – нормалізацію виконує споживач при декодуванні, а відстань таз–голова використовується адаптером для автоматичного масштабування кореневої трансляції. Це перший приклад принципу формату: продюсер записує параметри перетворення, тоді як його виконання відкладене до моменту декодування.

A2 – поканальне очищення. Сирі виходи моделей містять високочастотний джитер, короткі провали детекції та стрибки знаку кватерніонів. Очищення виконується для кожного каналу незалежно у три кроки: заповнення коротких розривів за довірою, згладжування Савицького–Голея з канально-специфічним вікном, фіналізація довір. Ширина вікна відображає шум джерела: 7 записів для скалярів і лицевих коефіцієнтів, 9 для поворотів тіла і голови, 11 для пальців з найвищим джитером. Повороти обробляються у кватерніонному домені з вирівнюванням знаків і ренормалізацією; довіри згладжуються поліномом степеня не вище першого і обрізаються у [0, 1]. Послідовність очищення одного каналу:

1) взяти записи каналу $(t_i, value_i, conf_i)$, поріг $conf_{floor}$, максимальний розрив $gap_{max} = 0{,}5$ с і вікно фільтра w;
2) позначити записи з $conf_i < conf_{floor}$ як відсутні;
3) для кожного короткого розриву між надійними моментами t_a і t_b, якщо $t_b - t_a \leq gap_{max}$, заповнити значення лінійною інтерполяцією між $value_a$ і $value_b$;
4) для поворотів у кватерніонах або поданні вісь-кут перейти до кватерніонного домену, вирівняти знаки сусідніх кватерніонів, згладити компоненти фільтром Савицького–Голея, ренормалізувати та повернути результат у вихідне подання;
5) для коефіцієнтів, довір і скалярів застосувати фільтр Савицького–Голея, причому для довір використати поліном степеня не вище першого та обрізання до діапазону $[0, 1]$;
6) видати очищені записи каналу з тими самими часовими мітками.

A3 – кодування. Алгоритм збирає маніфест (джерело, діапазон, калібрування, словник, описи каналів, політика), пакує кожен канал у бінарний файл із заголовком AMC1 і записує tar-архів, де маніфест іде першим членом, після чого архів стискається zstd. Під час кодування перевіряються два інваріанти: розмір запису кожного каналу має дорівнювати розміру його типу даних, а регіональне розбиття словника ARKit-52 має бути повним і без

перетинів. Порушення будь-якого з інваріантів зупиняє кодування з помилкою; формат не допускає файлів, що не проходять власну валідацію.

2.6 Consumer-алгоритми: декодування і синхронізація

A4 – декодування. Споживач читає маніфест першим членом архіву, будує типи записів кожного каналу з його опису і розпаковує бінарні дані. Вибір каналів відбувається за роллю: споживач запитує, наприклад, «всі канали ролі face», і отримує відео- та аудіоканали незалежно від їхніх імен і продюсерів. Канали з невідомими ролями пропускаються з попередженням; файли старої розкладки відхиляються з явним повідомленням. Після декодування виконується нормалізація до базису цільового середовища за матрицею з блоку calibration.

A5 – синхронізація. Канали мають різні частоти і різні часові мітки, а споживачеві потрібне значення кожного каналу у довільний момент часу t – кадр цільового середовища або позиція таймлайна редактора. Для кожного каналу алгоритм знаходить бракет – пару сусідніх записів $t_a \leq t < t_b$ – і інтерполює між ними за правилом, що залежить від типу поля. Кватерніонні поля (root_q, head_q) інтерполюються SLERP; повороти у поданні вісь-кут (body_pose, left, right) – SLERP окремо для кожного суглоба через кватерніонний домен; лицеві коефіцієнти, довіри, класи емоцій і VAD – лінійно. Якщо запитаний час лежить поза діапазоном каналу, повертається крайовий запис з довірою, помноженою на 0,5: споживач отримує сигнал, що дані екстрапольовані, і його подальші рішення (наприклад, регіональне злиття) автоматично знижують вагу такого каналу. Послідовність вибірки:

1) взяти декодовані канали $C_1, \ldots, C_n$ і час вибірки t;
2) для кожного каналу C_k, якщо t менше мінімальної часової мітки каналу, видати перший запис і зменшити його довіру вдвічі;

3) якщо t більше максимальної часової мітки каналу, видати останній запис і зменшити його довіру вдвічі;
4) інакше знайти бракет (t_a, t_b), для якого $t_a \leq t < t_b$, та обчислити $\alpha = \frac{(t - t_a)}{(t_b - t_a)}$;
5) для кватерніонів застосувати SLERP між значеннями сусідніх записів, а для поворотів вісь-кут виконати SLERP у кватерніонному домені окремо для кожного суглоба;
6) для коефіцієнтів, довір, класів емоцій і VAD застосувати лінійну інтерполяцію;
7) видати по одному узгодженому запису $S_1, \ldots, S_n$ для всіх каналів на час t.

У промислових системах захоплення руху синхронізацію розв'язує апаратний синхросигнал між камерами; тут її роль виконує спільна часова вісь кліпу, від якої походять мітки всіх каналів. Повний consumer-ланцюг показано на рисунку 2.5.

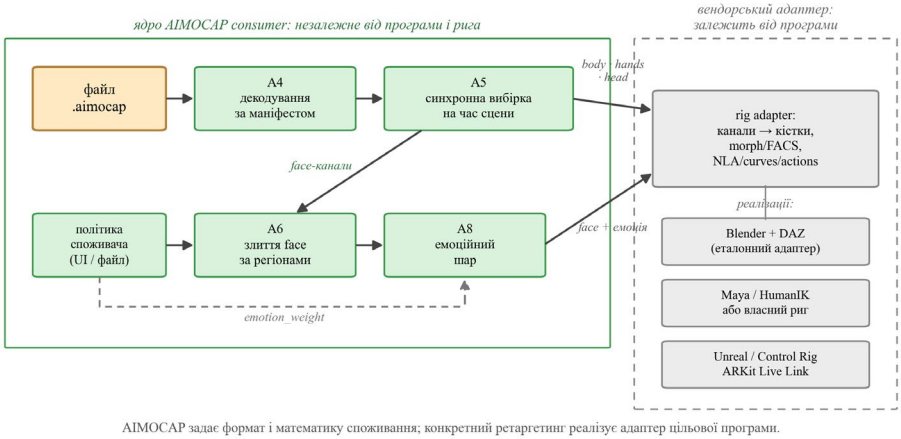

Рисунок 2.5 – Повний consumer-пайплайн

Ядро AIMOCAP завершується синхронізованими каналами, регіональним злиттям лицевих шарів, емоційним доданком і політикою споживача. Далі дані передаються у вендорський адаптер цільової програми: саме він знає, як записати

тіло у скелет, міміку у morph/FACS-властивості, а результати у доріжки нелінійної анімації (NLA), animation curves або інший механізм конкретного середовища. Такий адаптер може бути реалізований для Blender/DAZ, Maya/HumanIK, Unreal/Control Rig або будь-якого іншого сумісного рига; формат .aimocap не прив’язаний до жодного з них.

Принципово, що алгоритми A4 і A5 реалізовані один раз і використовуються різними споживачами: попередній перегляд редактора і доповнення Blender викликають один і той самий код вибірки (у доповненні Blender – вбудована копія модуля з тим самим алгоритмом, поведінка якої закріплена спільними тестами). Завдяки цьому те, що користувач бачить у редакторі, збігається з тим, що буде передано у цільовий адаптер.

2.7 Регіональне злиття лицевих каналів

Регіональне злиття – авторський алгоритм роботи (A6). Його мотивацію закладено в огляді предметної області: аудіоджерело є надійнішим за відео у нижній частині обличчя під час мовлення, оскільки безпосередньо відображає артикуляцію, тоді як відеоджерело краще зберігає міміку, положення голови, рухи очей і ті області обличчя, які не визначаються мовленнєвим сигналом. Тому просте усереднення двох джерел або жорсткий вибір одного з них є недостатнім: у різних анатомічних регіонах перевагу можуть мати різні канали. Реалізований алгоритм узагальнює це спостереження до зваженого змішування за сімома регіонами обличчя, у якому жорсткий пріоритет одного джерела залишається лише частковим випадком відповідного вибору ваг. Такий підхід також дає змогу змінювати поведінку системи без повторного обчислення каналів: достатньо змінити політику споживача. У результаті один і той самий файл .aimocap може давати різні варіанти артикуляції залежно від того, наскільки користувач довіряє відео- або аудіоджерелу в окремих регіонах.

Вхід алгоритму – один відеоканал обличчя, довільна кількість аудіоканалів обличчя (усі зі своїми region_conf) і політика споживача: вектор ваг w[r] за регіонами та поріг conf_floor. Для кожного регіону r обчислюються ефективні довіри відео- та аудіоканалу і частка аудіо:

$$\mathrm{fc_r} = \max(0, \mathrm{conf_v}[\mathrm{r}] - \mathrm{conf_{floor}}), \tag{2.3}$$

$$\mathrm{ac_r} = \max(0, \mathrm{conf_a}[\mathrm{r}] - \mathrm{conf_{floor}}) \cdot \mathrm{w}[\mathrm{r}], \tag{2.4}$$

$$\alpha_\mathrm{r} = \frac{\mathrm{ac_r}}{(\mathrm{fc_r} + \mathrm{ac_r})}, \tag{2.5}$$

де $\mathrm{conf_v}[\mathrm{r}]$ і $\mathrm{conf_a}[\mathrm{r}]$ – довіри відео- та аудіоканалу в регіоні r; $\mathrm{w}[\mathrm{r}]$ – вага аудіо для регіону з політики споживача; α_r – підсумкова частка аудіо у змішуванні коефіцієнтів регіону.

Коефіцієнти кожного регіону обчислюються лінійним змішуванням значень двох каналів з вагою α_r; за наявності кількох аудіоканалів їхні внески складаються у чисельнику і знаменнику. Загальну схему алгоритму показано на рисунку 2.6.

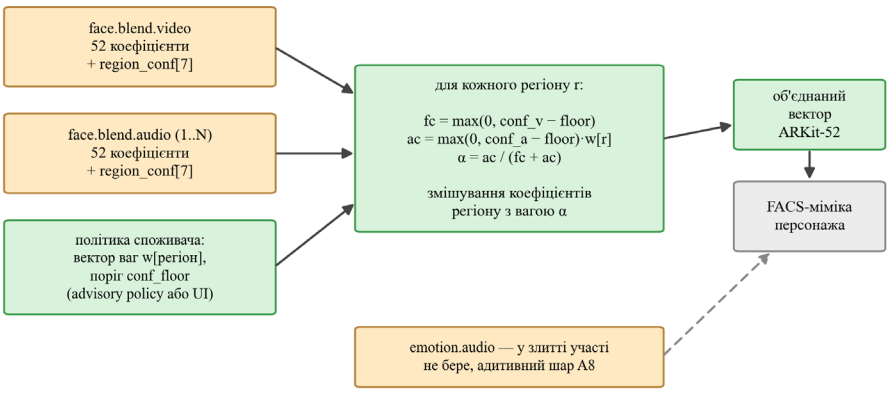

Рисунок 2.6 – Регіональне злиття лицевих каналів

Граничні випадки формул (2.3), (2.4) і (2.5): якщо довіра відеоканалу в регіоні впала нижче порога, $fc_r = 0$ і регіон повністю визначається аудіоканалом; якщо аудіоканал регіон не заявляє або $w[r] = 0$, регіон керується чисто відео – зокрема, за замовчуванням $w[eyes] = 0$, тому аудіо ніколи не торкається очей. Ваги за замовчуванням і їхнє обґрунтування наведено у Таблиці 2.5.

Таблиця 2.5 – Ваги аудіоканалу за регіонами обличчя за замовчуванням

Регіон	*Вага w[r]*	*Обґрунтування*
brow (брови)	0,25	брови видно у кадрі значно краще, ніж чутно; аудіо дає лише слабку просодичну підказку
eyes (очі)	0,00	погляд і моргання з аудіо не спостерігаються; регіон завжди за відео
cheek (щоки)	0,35	слабкі допоміжні рухи; аудіо корисне при поганому ракурсі
nose (ніс)	0,35	аналогічно щокам
jaw (щелепа)	0,65	артикуляцію щелепи під час мовлення краще чути, ніж видно
mouth (рот)	0,75	форма губ під час мовлення надійніше відновлюється з аудіо
tongue (язик)	1,00	відеомодель язика практично не бачить; регіон повністю віддається аудіоканалу, який його заявить

Жорсткий пріоритет аудіо для рота виражається у цих термінах одним рядком політики ($w[mouth] \rightarrow 1$), а вся поведінка алгоритму визначається одним вектором ваг, який зберігається у рекомендованій політиці файлу і перевизначається споживачем без повторної обробки. Канали тіла, кистей і голови у злитті участі не беруть – у кожного єдиний власник; емоційний канал застосовується окремим адитивним шаром (підрозділ 2.9), що усуває конфлікт між злиттям і емоційною модуляцією.

Поверх злиття споживач може застосувати посткорекцію під конкретний риг: тестове доповнення Blender підсилює апертуру рота персонажа DAZ (jawOpen × 1,45, притлумлення mouthClose 0,85, нижня губа × 1,20). Ці коефіцієнти компенсують консервативну мімічну механіку конкретного персонажа, належать ригу і тому не зберігаються у файлі.

2.8 Контракт вендорського адаптера цільового рига

A7 у межах роботи слід розуміти як контракт застосування синхронізованих каналів до довільного сумісного цільового середовища; претензій на новий універсальний метод ретаргетингу робота не висуває. Після A4–A6 система має значення body.pose, body.hands, head.pose, злитий face-вектор, emotion.audio і політику споживача у спільному часі. Далі вступає вендорський адаптер: він знає структуру конкретного рига, назви кісток, morph/FACS-властивості, механізм запису ключів, NLA-доріжки або animation curves. Такий адаптер може бути написаний для Blender, Maya, Unreal Engine або власного рушія; формат .aimocap і алгоритми синхронізації не залежать від цієї частини.

Практичною реалізацією в роботі є еталонний адаптер Blender/DAZ (reference adapter). Він потрібен для перевірки, що файл .aimocap достатній для реального 3D-середовища, але не заявляється як самостійний внесок у методи ретаргетингу. Адаптер інстанціюється профілем рига – JSON-ресурсом (JavaScript Object Notation) з таблицями відповідності кісток (bone_map), лицевих параметрів (face_map), режимом кореневого руху та правилами шарів. Такий профіль відокремлює загальну логіку читання .aimocap від особливостей конкретної моделі: назв кісток, локальних осей, набору блендшейпів, способу обробки кореневої кістки та структури NLA-доріжок. Для прикладу DAZ-профілів використано DAZ_GENESIS для native-рига Genesis і DAZ_MHX для

варіанта з MHX-структурою; інші риги потребують власних профілів і окремої перевірки у відповідній програмі.

В еталонному адаптері Blender/DAZ компенсація різниці нейтральних поз джерела і цілі для тіла реалізована формулою:

$$q_{tgt} = q'_{tgt} \cdot q_{src} \cdot (q'_{src})^{-1}, \quad (2.6)$$

де q_{src} – глобальне обертання кістки джерела; q'_{src} і q'_{tgt} – обертання відповідних кісток у нейтральних позах джерела і цілі.

Перенесення обертань за (2.6) доповнюється трансляцією кореня: коренева трансляція у цьому самому адаптері масштабується за відношенням зростів:

$$t_{tgt} = \frac{t_{src} \cdot h_{tgt}}{h_{src}}, \quad (2.7)$$

де h_{src} – відстань таз–голова джерела з rest-метаданих маніфесту; h_{tgt} – та сама відстань, виміряна на цільовому ризі.

Формули (2.6) і (2.7) описують конкретну тестову реалізацію і не є вимогою формату. Інший адаптер може використати IK-систему (Inverse Kinematics) рушія, HumanIK, Control Rig, власний solver-пайплайн або готовий ретаргетер програми, якщо він читає синхронізовані канали AIMOCAP і поважає політику споживача.

В еталонному адаптері Blender/DAZ використано два практичні рішення. По-перше, сегментна корекція напрямку кістки застосовується лише до кінцівок: для них додатково узгоджуються фактичні напрямки сегментів джерела і цілі, тоді як хребет переноситься без корекції. По-друге, шия і голова виключені з bone_map тіла: за них відповідає лицевий бік системи через окремий ланцюг голови, керований каналом head.pose. Таке розділення зменшує конфлікт між тілесним і лицевим каналами, коли обидва могли б одночасно змінювати орієнтацію голови. Пальці у цьому адаптері застосовуються через аналітичний

шарнірний IK (підрозділ 2.4), а лицева частина face_map переводить злитий вектор ARKit-52 у FACS-властивості підготовленого персонажа. Конкретна механіка драйверів або morph-властивостей належить адаптеру програми. Узагальнена послідовність A7:

1) взяти синхронізовані записи каналів на кадрі цільового середовища, політику споживача та профіль адаптера цільової програми P;
2) передати $\mathrm{body.pose}$, $\mathrm{body.hands}$ і $\mathrm{head.pose}$ у компонент $\mathrm{P.body_{mapper}}$;
3) передати злитий за A6 face-вектор у компонент $\mathrm{P.face_{mapper}}$;
4) передати $\mathrm{emotion.audio}$ у компонент $\mathrm{P.emotion_{mapper}}$ або пропустити цей крок, якщо цільове середовище не підтримує адитивний емоційний шар;
5) застосувати $\mathrm{root_{motion}}$, $\mathrm{conf_{floor}}$, $\mathrm{emotion_{weight}}$ та інші параметри політики споживача;
6) записати ключі, криві або доріжки у форматі цільової програми, а для еталонного адаптера Blender/DAZ сформувати доріжки AIMOCAP_BODY, AIMOCAP_FACE і AIMOCAP_EMOTION у NLA-стеку;
7) видати анімаційні дані цільового середовища.

2.9 Емоційний адаптер

Алгоритм A8 застосовує канал emotion.audio до персонажа. Канали тіла й обличчя керують геометрією прямо; емоція натомість працює як модуляційний шар, що додає до базової міміки прототипні вирази відповідно до поточного емоційного стану.

Зв’язок між класами емоцій і мімікою задається рецептами – редагованим JSON-ресурсом, у якому кожному з семи канонічних класів відповідає набір

FACS-властивостей з ваговими коефіцієнтами. Наприклад, рецепт радості піднімає кутики губ і щоки, рецепт суму опускає брови біля перенісся. На кожному кадрі адаптер обчислює адитивний доданок до кожної FACS-властивості:

$$\Delta_u(t) = \text{emotion}_{\text{weight}} \cdot \Sigma_{k=1\ldots7}\, p_k(t) \cdot R_k[u], \quad (2.8)$$

де $p_k(t)$ – ймовірність k-го класу з каналу emotion.audio; $R_k[u]$ – вага властивості u у рецепті класу k; $\text{emotion}_{\text{weight}}$ – глобальна вага шару з політики споживача. Результат обрізається у допустимі межі властивості.

У Blender доданок (2.8) записується в окрему дію AIMOCAP_EMOTION, розміщену у NLA-стеку поверх дії обличчя з режимом змішування ADD та параметром influence, що дорівнює emotion_weight. З такого розміщення випливають три властивості, які перевіряються експериментально у розділі 3: емоційний шар вимикається без сліду (вимкнення доріжки повертає базову міміку), масштабується одним параметром і редагується незалежно від базової анімації.

Рецепти не містять властивостей рота: за артикуляцію відповідає злиття A6, і виключення рота з рецептів усуває конфлікт між емоційним шаром і мовленням. Скаляри VAD зберігаються у каналі та доступні споживачам, але в поточній версії адаптера як модулятор не використовуються; це задокументоване обмеження поточної версії адаптера.

М’якість ймовірностей p_k робить поведінку шару неперервною: при плавній зміні емоційного стану мовця доданки сусідніх класів плавно перетікають одна в одну, без перемикань між дискретними станами.

2.10 Метрики оцінювання системи

Оцінювання спирається на дві групи метрик. Цитовані з літератури (MPJPE, PVE, W-MPJPE для тіла, SyncNet для обличчя; Unweighted Accuracy, Weighted F1) використовуються лише для опису сторонніх моделей за їхніми публікаціями: самостійне вимірювання потребувало б еталонних даних, яких для власних тестових кліпів не існує.

Власні метрики тому будуються як проксі-метрики без еталона; кожна вимірює одну з чотирьох властивостей – стабільність, узгодженість, покриття або керованість.

Темпоральна стабільність вимірюється джитером – середньою нормою другої різниці сигналу:

$$J = \mathrm{mean}_t \, ||\theta(t+1) - 2\theta(t) + \theta(t-1)||, \tag{2.9}$$

де $\theta(t)$ – кут суглоба у кадрі t; одиниці – градуси на кадр у квадраті. Друга різниця відокремлює шум від справжнього руху: повільний рух має малу другу різницю, високочастотне тремтіння – велику.

Доповненням до джитера слугує діапазон рухливості суглоба (дисперсія кута за час кліпу): він показує, чи джерело взагалі рухає суглоб, і застосовується насамперед до пальців.

Для аудіосигналу додатково будується середньоквадратична огинаюча (RMS-огинаюча, root mean square envelope) – часова крива локальної енергії звуку, отримана як середньоквадратичне значення амплітуди на коротких вікнах. На відміну від сирої звукової хвилі, така огинаюча не містить швидких коливань несучого сигналу і відображає зміну інтенсивності мовлення у часі.

Узгодженість артикуляції з мовленням вимірюється коефіцієнтом кореляції Пірсона між кривою jawOpen(t) і RMS-огинаючою аудіодоріжки. Метрика обчислюється для різних політик рота (лише відео, ваги за

замовчуванням, лише аудіо) і для стрес-умов; зростання кореляції при підключенні аудіоканалу є кількісним свідченням його внеску.

Кадрова синхронність джерела і рендера вимірюється через енергію руху – середній модуль міжкадрової різниці яскравості зменшеного кадру по формулі (2.10).

$$E(t) = \frac{1}{HW}\Sigma_{x,y}\,|Y(x,y,t+1) - Y(x,y,t)|, \qquad (2.10)$$

де $Y(x,y,t)$ – яскравість пікселя зменшеного кадру t; H, W – розміри зменшеного кадру. Пряме порівняння пікселів джерела і рендера беззмістовне (персонаж не схожий на людину в кадрі), натомість моменти рухів мають збігатися, тому порівнюються саме криві енергії руху.

Критерієм синхронності слугує сканування взаємної кореляції двох кривих за цілими зсувами:

$$r(\tau) = \mathrm{corr}\left(E_{\text{джерела}}(t), E_{\text{рендера}}(t+\tau)\right),\ \tau = \{-12\ \ldots\ 12\}, \qquad (2.11)$$

де τ – зсув у кадрах. Якщо максимум $r(\tau)$ припадає на $\tau = 0$, систематичного дрейфу кадрів немає; значення $r(0)$ показує, наскільки щільно рендер повторює моменти рухів джерела.

Покриття і довіра описуються частками: відсоток кадрів з успішною детекцією кистей, відсоток кадрів з довірою регіону вище порога, поведінка conf(t) і region_conf(t) при деградації якості відео. Керованість перевіряється відгуком на політики: сімейство кривих jawOpen при зміні ваги w[mouth] від 0 до 1 має монотонно переходити від відеоформи до аудіоформи. Окремо фіксується стабільність каналів у часі: різкі провали довіри, поодинокі викиди та ділянки втрати детекції мають бути видимими на графіках і не маскуватися усередненням. Для кожного каналу зіставляються сирі та очищені послідовності, щоб перевірити, чи фільтрація зменшує шум без втрати характерних рухів.

Системні показники доповнюють картину: час обробки кожного етапу конвеєра, ефект кешу при повторному запуску, пікове споживання оперативної та відеопам’яті, кількість записів і фактична частота кожного каналу, розмір файлу до і після стиснення. Показники реального часу (наскрізна затримка, кадрова частота онлайн-обробки) не вимірюються: система офлайнова, і такі показники не мали б змісту.

Висновки до розділу 2

У другому розділі викладено архітектурну і алгоритмічну основу системи. Канальна модель розділяє дані на шість каналів спостереження з п’ятьма ролями; шарова модель фіксує межу файлу між очищеними спостереженнями з похідною семантикою та злиттям з бейком на боці споживача. Звідси наскрізна теза роботи: файл фіксує очищені спостереження моделей, а рішення про їхнє поєднання ухвалює споживач.

Зафіксовано словники даних (SMPL-X для тіла, MANO для пальців, кватерніон голови, ARKit-52 для лицевих каналів) і введено регіональну модель обличчя – повне розбиття коефіцієнтів на сім анатомічних регіонів з вектором довір region_conf, що об’єднує покриття і надійність джерела. Описано формат .aimocap: контейнер zstd(tar) з маніфестом першим членом, самоописними бінарними каналами і рекомендованою політикою, яка ніколи не застосовується до збережених значень.

Формалізовано три producer-алгоритми (калібрування з правоорієнтованим базисом, поканальне очищення у кватерніонному домені, кодування з перевіркою інваріантів) і п’ять consumer-алгоритмів (декодування за маніфестом, синхронізація з типозалежною інтерполяцією і крайовим штрафом довіри, регіональне злиття за ефективними довірами, контракт вендорського адаптера цільового рига, адитивний емоційний шар з FACS-рецептами).

Власна оцінка системи будується на проксі-метриках стабільності, узгодженості, покриття і керованості без еталонних даних; цитовані метрики сторонніх моделей використовуються лише за оригінальними публікаціями. Третій розділ проходить цей самий маршрут на реалізованій системі і експериментально показує внесок кожного каналу.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА СИСТЕМИ

3.1 Огляд реалізованого програмного комплексу

Запропоновану архітектуру реалізовано у вигляді завершеного програмного комплексу з трьох компонентів, межі відповідальності яких повторюють схему «producer – файл – consumer» з підрозділу 2.1. Бібліотека aimocap містить формат, producer- і consumer-алгоритми та обгортки моделей; настільний редактор керує обробкою і діагностикою; тестове доповнення Blender є еталонним адаптером, що застосовує збережені дані до персонажа DAZ. Такий поділ дає змогу перевіряти кожен рівень окремо: бібліотеку – через тести формату й алгоритмів, редактор – через спостережуваність каналів, а Blender-адаптер – через відповідність рендера вхідному кліпу. Крім того, компоненти не дублюють відповідальність один одного: редактор не виконує ретаргетинг, а Blender-адаптер не запускає важкі нейромережеві моделі. Склад комплексу наведено у Таблиці 3.1.

Таблиця 3.1 – Компоненти реалізованого програмного комплексу

Компонент	*Каталог*	*Призначення*
Бібліотека aimocap	aimocap/	формат .aimocap; алгоритми A1–A6; конвеєр обробки; обгортки моделей; кеш
Настільний редактор	app/	завантаження відео, вибір діапазону, запуск конвеєра, viewport-и, таймлайн, діагностика, експорт і повторне відкриття файлів
Доповнення Blender	blender_addon/	еталонний адаптер: імпорт .aimocap, синхронізація до частоти сцени, злиття, застосування до персонажа DAZ за профілем рига, емоційний шар, доріжки нелінійної анімації

Типовий сценарій користувача: відкрити відеофайл, обрати діапазон кадрів на таймлайні, налаштувати канали і джерела, запустити обробку, переглянути кожен канал у viewport-ах і на доріжках, експортувати або повторно відкрити файл .aimocap, імпортувати його у Blender і застосувати до персонажа.

Технологічний стек: Python, PySide6/Qt, PyTorch (NLF, smplfitter), MediaPipe Tasks, HaMeR в ізольованому підпроцесі, NVIDIA Audio2Face-3D як контейнерний NIM-сервіс у Docker під WSL (Windows Subsystem for Linux), Python API Blender. Розгортання компонентів показано на рисунку 3.1.

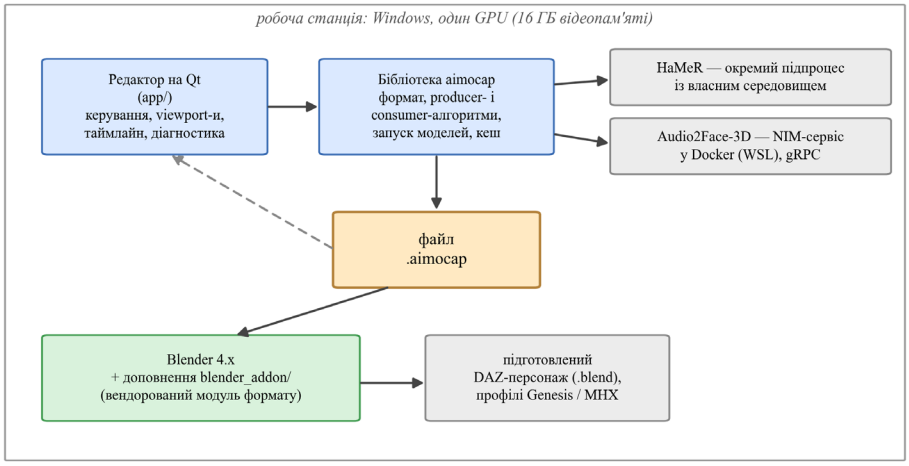

Рисунок 3.1 – Компоненти комплексу і середовища

Дві найважчі залежності ізольовано: HaMeR має несумісний з основним середовищем набір PyTorch-залежностей і виконується окремим підпроцесом з власним віртуальним середовищем, а Audio2Face-3D розгортається офіційним контейнером NVIDIA із взаємодією через gRPC (Remote Procedure Calls); бібліотека керує запуском і станом контейнера. Завдяки ізоляції основне середовище залишається легким, а збій важкої залежності не призводить до аварійного завершення редактора. Підрозділи 3.2–3.5 описують реалізацію за маршрутом даних, 3.6–3.7 – експериментальну перевірку, 3.8 – обмеження.

3.2 Реалізація формату .aimocap та її перевірка

Формат з підрозділу 2.3 реалізовано повністю; цей підрозділ наводить його фактичні характеристики на реальному тестовому файлі. Опорним матеріалом слугує самостійний десятисекундний кліп, вирізаний з відкритого тестового відеозапису [52] (інтервал 5–15 секунд джерела): 240 кадрів при 24 кадрах за секунду, роздільність 1280×720, синхронне аудіо. Фрагменти відеозапису використано виключно з некомерційною дослідницькою метою; кадри цього кліпу присутні на ілюстраціях розділу. Кліп містить безперервне мовлення з активною жестикуляцією, тобто навантажує одночасно всі чотири модальності; його часова шкала самодостатня – кадри нумеруються від нуля. Вміст відповідного файлу .aimocap наведено у Таблиці 3.2.

Таблиця 3.2 – Вміст тестового файлу .aimocap (кліп 10 с, 240 кадрів)

Канал	*Продюсер*	*Частота, Гц*	*Записів*	*Байтів на запис*
body.pose	NLF + smplfitter	24	240	336
body.hands	HaMeR	24	240	372
body.hands.debug	HaMeR (діагностика)	24	240	892
head.pose	MediaPipe	24	240	28
face.blend.video	MediaPipe	24	240	248
face.blend.audio	Audio2Face-3D	30	300	248
emotion.audio	Audio2Face-3D (емоція)	30	300	52

Таблиця демонструє дві властивості формату на реальних даних. Канали справді різночастотні: відеоканали успадковують 24 кадри за секунду джерела, аудіоканали працюють на власних 30 Гц виходу Audio2Face-3D. Розмір файлу залишається скромним: нестиснений tar-архів займає 1060 КБ, після zstd-стиснення – 592 КБ, тобто приблизно 59 КБ на секунду кліпу разом з діагностичним каналом.

Життєвий цикл файлу відповідає трьом ролям з підрозділу 2.3. Як кеш: повторне відкриття у редакторі відновлює діапазон, стан таймлайна і

налаштування з блоку editor маніфесту без повторної обробки. Як формат обміну: файл є входом будь-якого сумісного consumer-адаптера; у роботі таким адаптером є Blender/DAZ. Як одиниця експерименту: всі порівняння підрозділу 3.7 виконуються над зафіксованими файлами.

Реалізацію формату закріплено набором тестів, які перевіряють: повноту і неперетинність регіонального розбиття ARKit-52; відповідність розміру запису кожного каналу розміру його типу даних; точний цикл кодування-декодування всіх канонічних типів записів без втрат; коректне декодування каналу з невідомим іменем, але відомою роллю; безпечний пропуск каналів з невідомою роллю; відмову від файлів старої розкладки з явним повідомленням.

3.3 Конвеєр обробки відео

Producer-конвеєр бібліотеки перетворює обраний діапазон відео на файл .aimocap. Порядок етапів показано на рисунку 3.2:

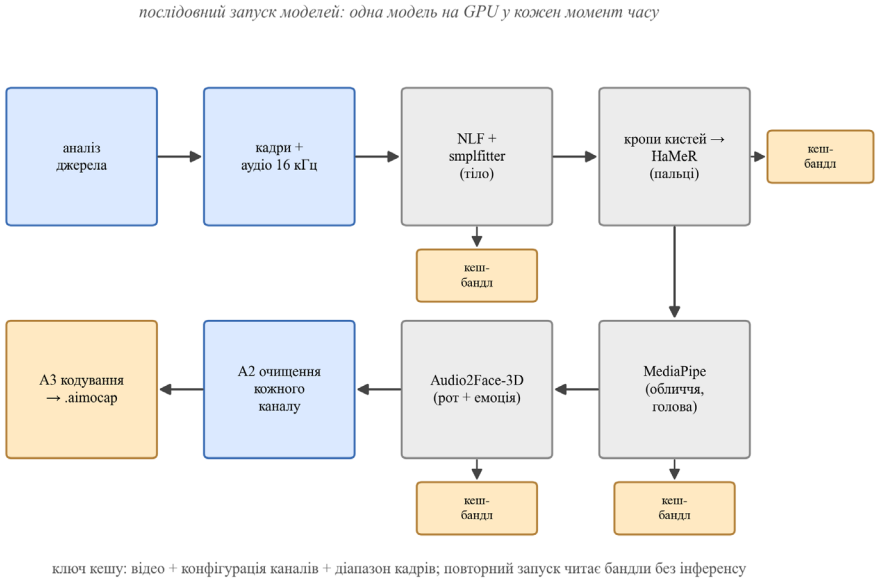

Рисунок 3.2 – Конвеєр обробки відео з кеш-пакетами етапів

Моделі запускаються послідовно: у кожен момент часу на GPU працює щонайбільше одна важка модель. Це свідоме рішення під обмеження побутового GPU з 16 ГБ відеопам’яті з постановки задачі: паралельний запуск NLF і HaMeR не вміщується у пам’ять, а вивантаження і завантаження ваг коштує дорожче за послідовне виконання.

Кожен важкий етап має власний кеш-пакет: результати тіла, обличчя з головою та аудіоканалів зберігаються окремими файлами, ключ кешу складається з відео, конфігурації каналів і діапазону кадрів. Повторний запуск з тими самими параметрами читає пакети і пропускає інференс повністю; зміна одного джерела (наприклад, перемикання пальців з NLF на HaMeR) перераховує лише відповідний етап. Редактор надає команду очищення кешу конкретного відео.

Ланцюг кистей реалізує декомпозицію з підрозділу 1.5. З двовимірних суглобів NLF будуються квадратні кропи кистей із запасом 2,15 від розміру долоні; кропи передаються у підпроцес HaMeR; той повертає MANO-пози пальців, ознаки успішної детекції та ключові точки. Пози потрапляють у канал body.hands, службові дані – у діагностичний канал body.hands.debug, який живить viewport-и кистей редактора. Якщо HaMeR недоступний або детекція не вдалася, канал заповнюється позами пальців з виходу NLF; фактичний продюсер каналу при цьому фіксується у маніфесті.

Аудіоканали походять від сервісу Audio2Face-3D: бібліотека керує життєвим циклом контейнера, передає аудіодоріжку за gRPC і розбирає вихідні послідовності у канали face.blend.audio та emotion.audio. Часові мітки потокового сервісу мають сталий зсув відносно аудіодоріжки; конвеєр калібрує його автоматично крос-кореляцією кривої jawOpen зі згладженою RMS-огинаючою (на тестовому матеріалі – близько 0,87 с) і фіксує застосований зсув у трасуванні. За недоступності сервісу передбачено віземний резерв на основі фонемного розпізнавача; в експериментальній перевірці він не використовується.

Останнім кроком перед кодуванням кожен канал проходить очищення A2 з канально-специфічними вікнами. Зведення моделей, каналів і резервів наведено у Таблиці 3.3.

Таблиця 3.3 – Моделі конвеєра, їхні канали і поведінка при збої

Модальність	*Основна модель*	*Канали*	*Поведінка при збої*
тіло	NLF + smplfitter	body.pose	етап обов'язковий; без тіла обробка зупиняється
пальці	HaMeR (підпроцес)	body.hands, body.hands.debug	пози пальців з виходу NLF; продюсер фіксується у маніфесті
обличчя, голова	MediaPipe Face Landmarker	face.blend.video, head.pose	кадри без детекції отримують нульову довіру
рот з аудіо	Audio2Face-3D (NIM-сервіс)	face.blend.audio	вбудований віземний резерв; в експериментах не використовується
емоція	Audio2Face-3D (емоційні оцінки)	emotion.audio	резерв класу wav2vec2-SER; в експериментах не використовується

3.4 Інтерактивний редактор

Редактор призначений для керування конвеєром і діагностики каналів; перегляд готового результату є допоміжною функцією. Його вікно складається з чотирьох зон: viewport-и зліва і по центру, панелі керування та стану конвеєра праворуч, таймлайн і консоль знизу. Така структура відповідає логіці роботи з .aimocap: користувач одночасно бачить вхідне відео, проміжні спостереження моделей, стан обробки та часові доріжки довіри. Основна увага редактора спрямована не на фінальний рендер, а на пояснюваність конвеєра: кожен канал можна перевірити до того, як він буде застосований у Blender. Це важливо для пошуку помилок, оскільки невдала поза тіла, нестабільні пальці або некоректна

артикуляція рота мають різні джерела і потребують різної діагностики. Загальний вигляд з відкритим обробленим тестовим кліпом показано на рисунку 3.3; далі кожну зону розглянуто окремо.

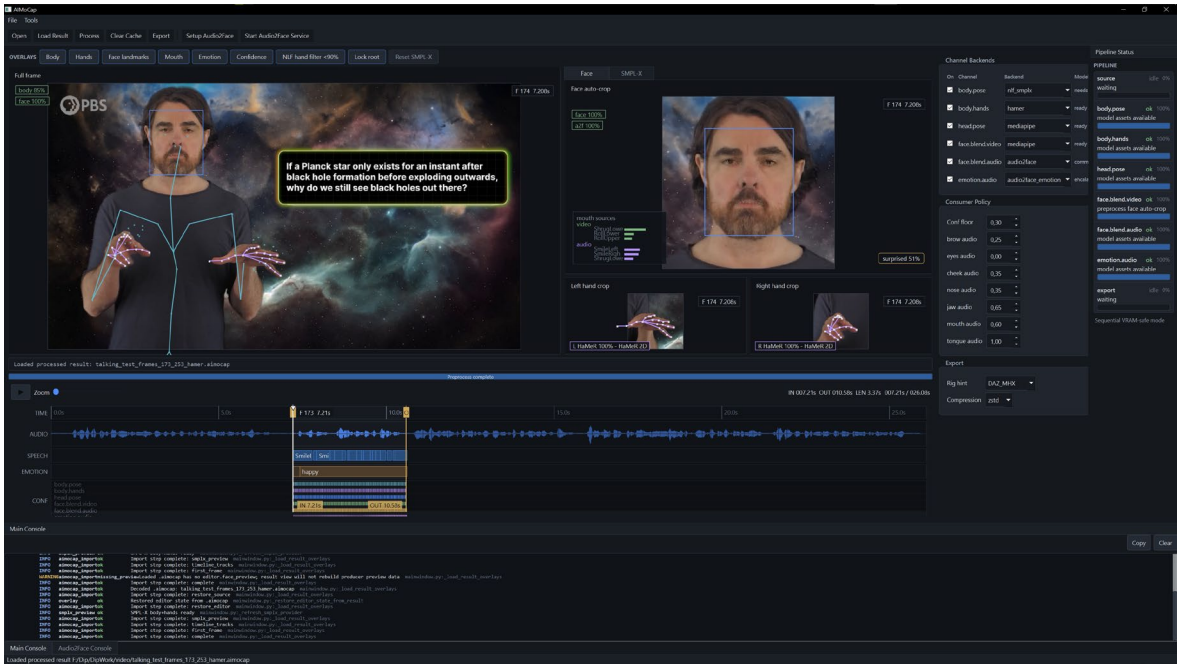

Рисунок 3.3 – Головне вікно редактора

Viewport-и показують кожен канал у його природному поданні. У базовій конфігурації їх чотири: повний кадр з перемикачами оверлеїв (скелет тіла, кисті, лицеві орієнтири, рот, емоція, довіри) і позначками покадрових довір каналів; автоматичний кроп обличчя з підписом активного класу емоції та стовпчиковими індикаторами найактивніших коефіцієнтів рота окремо для відео- та аудіоджерела; два кропи кистей з ключовими точками HaMeR, які читаються з діагностичного каналу body.hands.debug разом з підписом фактичного джерела. Завдяки цьому користувач бачить не лише фінальне положення персонажа, а й проміжні докази того, який саме канал сформував відповідний рух. Цю четвірку показано на рисунку 3.4.

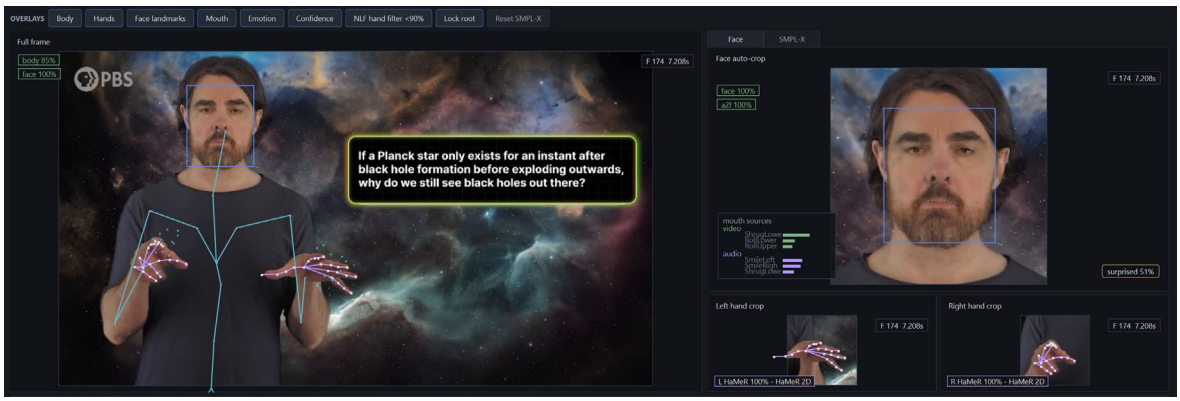

Рисунок 3.4 – Чотири viewport-и редактора

Окрема вкладка перемикає праву частину viewport-ів у режим SMPL-X: замість кропу обличчя і кистей показується тривимірна реконструкція тіла з каналів body.pose і body.hands через smplfitter. Кисті у реконструкції відображаються при довірі вище порога hand_conf_floor = 0,90, корінь фіксується для зручності огляду, а позначка над реконструкцією показує сумарну довіру і джерело пальців. Пару «повний кадр + SMPL-X» показано на рисунку 3.5.

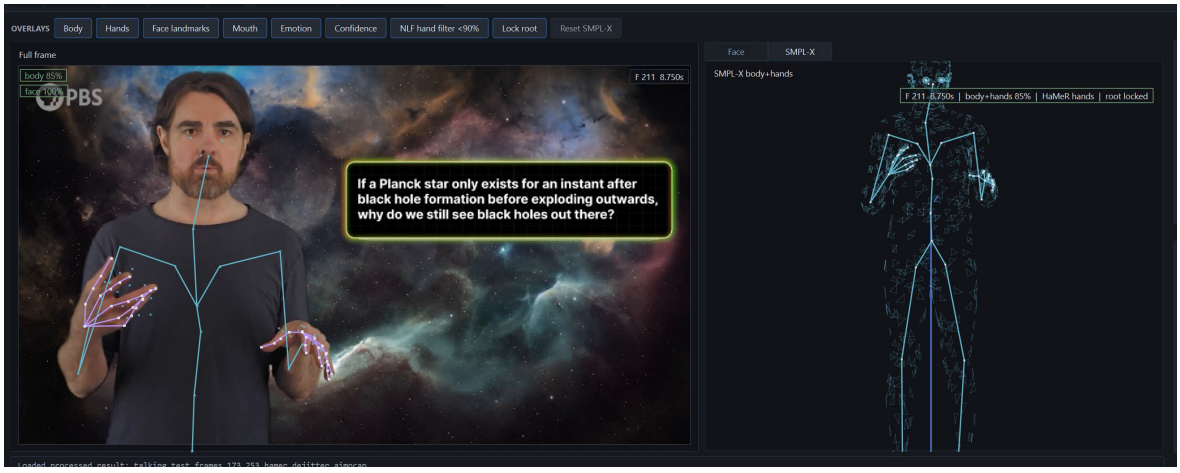

Рисунок 3.5 – Пара viewport-ів у режимі SMPL-X

Таймлайн зводить часову структуру каналів в одному місці: лінійка кадрів з позицією відтворення, осцилограма аудіо, доріжка активності мовлення, доріжка емоцій з підписами класів і шість кольорових доріжок довіри – по одній

на канал спостереження. Межі діапазону обробки задаються маркерами IN/OUT прямо на лінійці (рисунок 3.6).

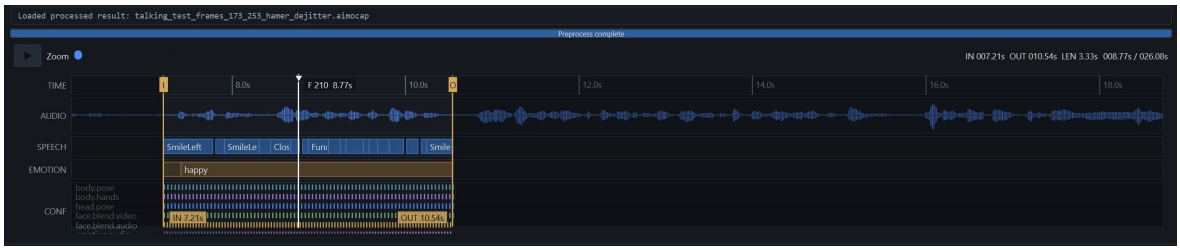

Рисунок 3.6 – Таймлайн редактора

Панелі керування праворуч зібрані у три блоки. Блок бекендів каналів дозволяє вмикати кожен канал окремо і обирати його джерело (наприклад, пальці з NLF або HaMeR) зі статусом готовності моделі: готова, потребує середовища, потребує ваг. Це дає змогу ще до запуску обробки побачити, які частини конвеєра доступні на поточній машині, а які потребують додаткового налаштування. Блок політики споживача містить поріг conf_floor і ваги аудіоканалу за сімома регіонами – ті самі параметри, що в Таблиці 2.4. Зміна цих параметрів не змінює збережені спостереження у файлі, а лише визначає спосіб їхнього поєднання під час застосування. Блок експорту задає підказку цільового рига і алгоритм стиснення. Панель стану конвеєра показує послідовність етапів з прогресом кожного і режим послідовного виконання моделей. Вона також дає змогу швидко побачити, на якому саме каналі зупинилася або сповільнилася обробка. Ці панелі показано на рисунку 3.7.

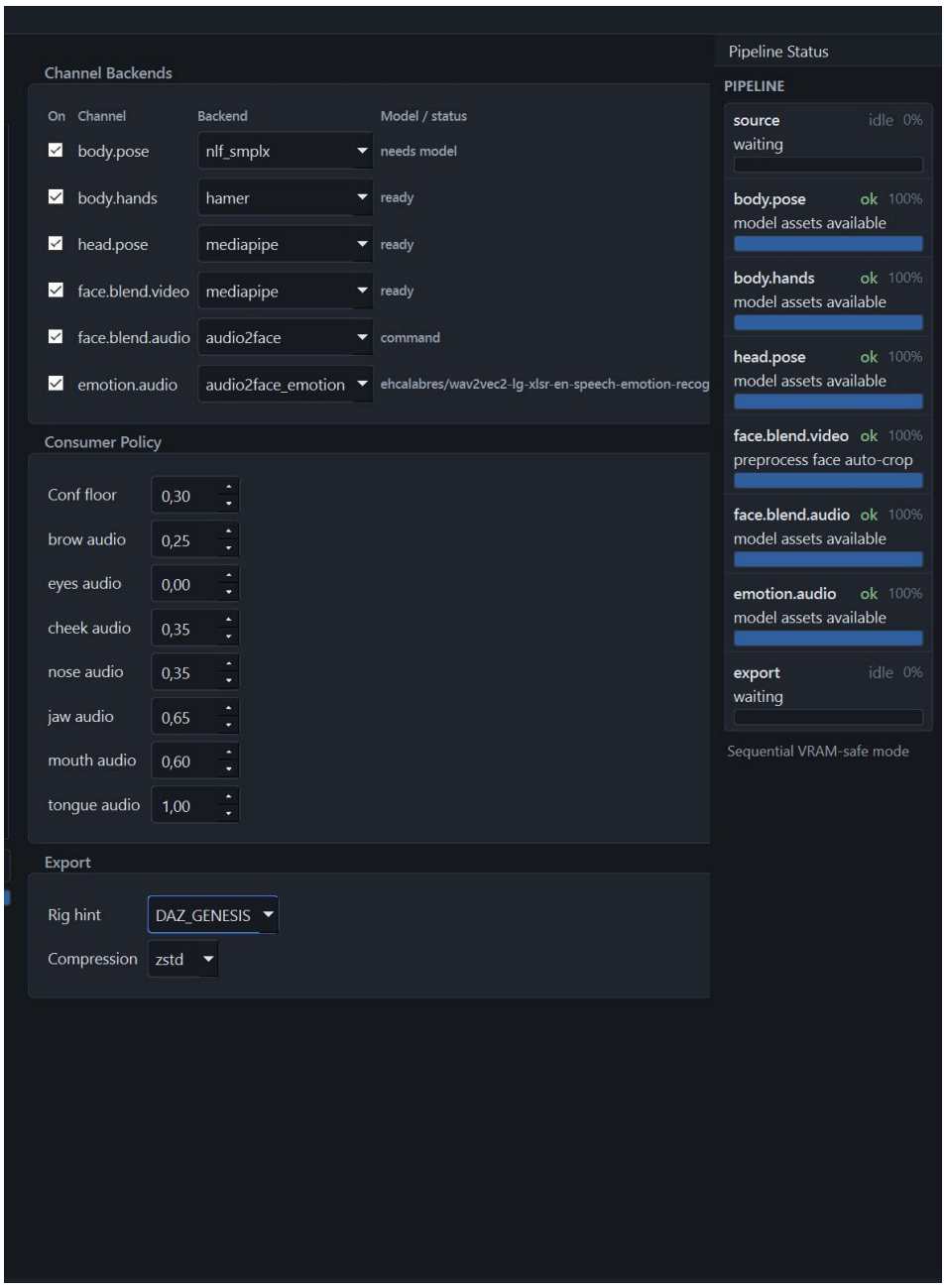

Рисунок 3.7 – Панелі керування редактора

Консольна зона документує обробку: головна консоль трасує кроки конвеєра та імпорту результату (декодування файлу, відновлення стану редактора, готовність реконструкції), окрема вкладка виводить журнал сервісу Audio2Face-3D. Така діагностика робить кожен канал спостережуваним до того, як він потрапить у файл: користувач бачить, яке джерело реально працювало і з якою довірою (рисунок 3.8).

Рисунок 3.8 – Консоль редактора

Життєвий цикл результату влаштовано строго через файл: після завершення обробки редактор читає лише декодований .aimocap, без доступу до сирих виходів моделей. Повторне відкриття файлу відновлює діапазон, поточний кадр і стан таймлайна з блоку editor маніфесту. Попередній перегляд злитої міміки використовує ті самі алгоритми A5 і A6, що і доповнення Blender, тому зображення у редакторі і результат бейку узгоджені за побудовою.

3.5 Еталонний адаптер Blender/DAZ

Тестове доповнення Blender відповідає на практичне питання роботи: чи придатний файл .aimocap для застосування у реальному 3D-середовищі. Воно позиціонується як споживач для перевірки, без заявки на окремий внесок з ретаргетингу.

Вхід – файл .aimocap і підготовлений персонаж DAZ Genesis з FACS-контролерами (профілі native і MHX). Послідовність застосування повторює consumer-ланцюг розділу 2 (рисунок 3.9): декодування вбудованою копією модуля формату, побудова часів бейку з діапазону файлу і частоти сцени, далі покадрово – вибірка A5, злиття A6 з вагами з інтерфейсу, посткорекція рота, передача тіла, пальців, голови, міміки та емоції в еталонний адаптер.

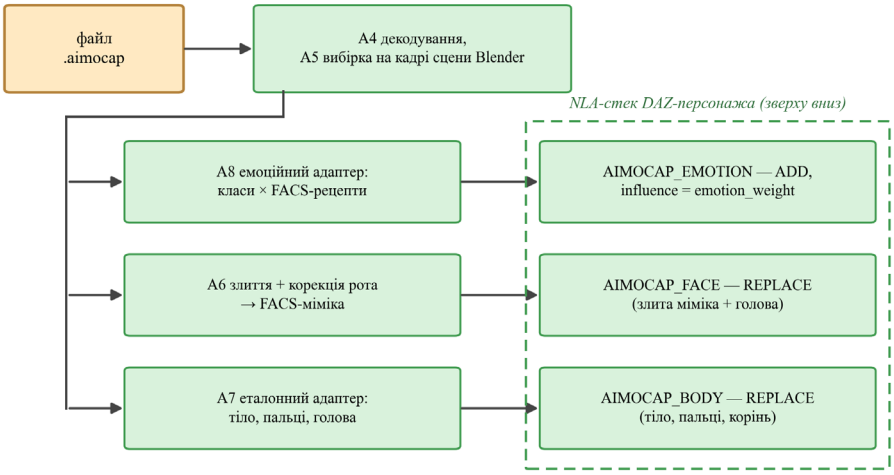

Рисунок 3.9 – Consumer-ланцюг доповнення Blender

Результат бейку розкладається у три дії нелінійної анімації: AIMOCAP_BODY і AIMOCAP_FACE у режимі REPLACE та AIMOCAP_EMOTION у режимі ADD з вагою emotion_weight. Дія AIMOCAP_BODY містить трансформації тіла, кореневого руху та пози кінцівок, тоді як AIMOCAP_FACE відповідає за базову міміку, отриману після злиття відео- й аудіоканалів обличчя. Емоційна доріжка винесена окремо, оскільки вона не повинна перезаписувати артикуляцію рота або основну міміку, а лише додавати керовану надбудову для брів, очей та інших областей, визначених рецептом емоції. Такий поділ дає змогу незалежно вимикати або редагувати тіло, базову міміку та емоційний шар без повторного імпорту файлу.

Нумерація кадрів узгоджена з кліпом: тестовий файл самодостатній з діапазоном від нуля, тож кадр k кліпу відповідає кадру k+1 сцени Blender, оскільки сцена нумерується з одиниці. Відповідність задається під час бейку, тому всі ключі анімації потрапляють у сцену з однаковим детермінованим зсувом на один кадр. Це спрощує повторну перевірку: кадр з відео, записаний у таблицях і підписах рисунків, має однозначну позицію у Blender-сцені. Завдяки цьому звіряння рендерів з відео не потребує жодних додаткових часових зсувів. Стек доріжок показано на рисунку 3.10.

Рисунок 3.10 – Доріжки нелінійної анімації персонажа після бейку

Параметри споживача зібрані у панелі доповнення: ваги злиття за регіонами, поріг conf_floor, вага емоційного шару, режим кореневого руху, профіль рига. Кожен з них відповідає параметру рекомендованої політики з Таблиці 2.4 і може відрізнятися від записаного у файлі. Доповнення також має автоматизований режим без графічного інтерфейсу, яким виконувалися повторювані бейки експериментальної перевірки.

Успішною перевіркою вважається виконання чотирьох критеріїв: персонаж не віддзеркалений і не перевернутий (працює калібрування A1); номери кадрів сцени збігаються з кадрами відео (працює синхронізація A5); тіло, міміка та емоція лежать на окремих редагованих доріжках (працює канальна модель); вимкнення емоційної доріжки не змінює базову міміку (працює адитивність A8). Збіг кадрів перевіряється не лише візуально, а й програмно – метрикою кадрової синхронності з підрозділу 2.10: для відео джерела і послідовності рендерів обчислюються криві енергії руху за формулою (2.10) і

сканується їхня взаємна кореляція за формулою (2.11). Пік сканування припадає точно на нульовий зсув зі значенням r = 0,904 – систематичного дрейфу кадрів немає. Збіг ілюструє рисунок 3.11: три стоп-пари «кадр відео – рендер того самого номера», обрані програмно на трьох найбільших піках енергії руху кліпу, тобто у найдинамічніші моменти, де розсинхронізація була б найпомітнішою.

Рисунок 3.11 – Кадрова відповідність відео і анімації

Саму порівнювану величину E(t) з формули (2.10) розгорнуто на рисунку 3.12: нормовані криві енергії руху відео і рендера на всьому кліпі збігаються пік у пік – кожен жест джерела має синхронний відгук на персонажі, а коефіцієнт кореляції кривих без зсуву r(0) за формулою (2.11) становить 0,904. Оскільки криві нормовано, це порівняння характеризує насамперед часову узгодженість змін, а не абсолютну рівність амплітуд руху. Тому значення r(0) використовується як показник синхронності динаміки між джерелом і результатом рендеру, а не як самостійна метрика геометричної точності пози. Динамічну версію цієї перевірки дає відеодоказ media/PVSync.mp4 у репозиторії роботи [53]: обидві послідовності відтворюються поруч строго за номером кадру зі спільним лічильником.

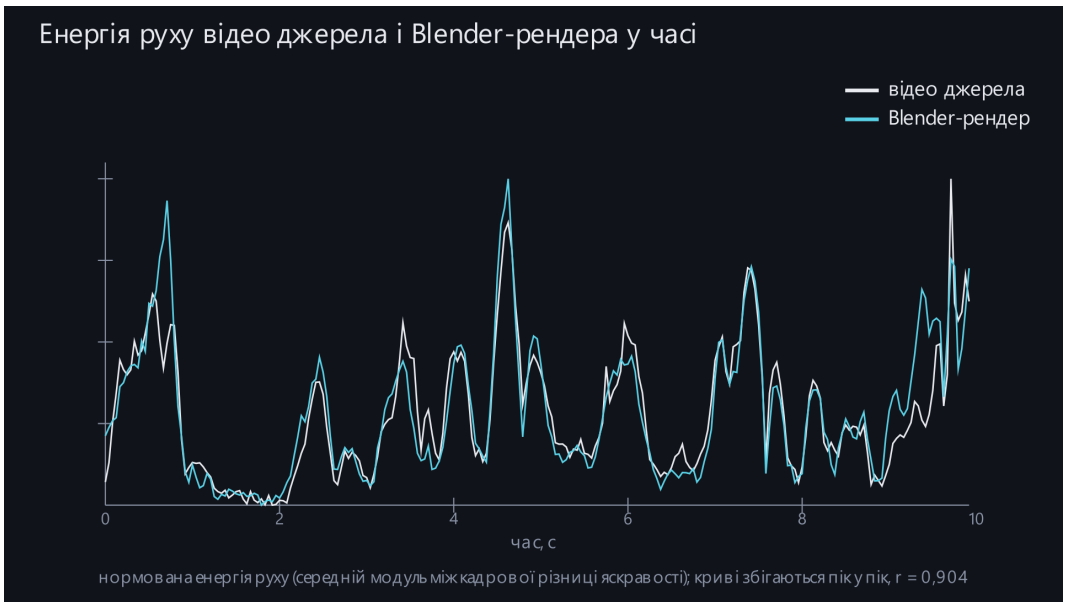

Рисунок 3.12 – Нормовані криві енергії руху відео джерела і Blender-рендера на всьому кліпі

3.6 Методика експериментальної перевірки

Експериментальна перевірка має показати внесок кожного каналу й алгоритму окремо. Використано три принципи: фіксований порівняльний фрагмент, порівняння політик без повторної обробки, стрес-модифікації входу.

Перший принцип: усі порівняння виконуються на одному десятисекундному кліпі з підрозділу 3.2, інакше відрізнявся б не лише алгоритм, а й рух, фраза та освітлення. Кожен вхід експерименту – окремий самодостатній відеофайл, а кожен експеримент – окремий файл .aimocap; це виключає будь-які зсуви часових шкал між порівнюваними конфігураціями. Кожне порівняння – пара експериментів одного кліпу, в яких змінюється рівно один фактор: джерело пальців, стан очищення, політика або вага емоційного шару.

Другий принцип спирається на головну властивість формату: рішення про поєднання каналів у файлі не зафіксоване. Порівняння політик споживача (ваги

регіонального злиття, вага емоції, режим кореня) виконуються над одним і тим самим файлом .aimocap зміною параметрів вибірки, без жодного повторного запуску моделей. Порівняння джерел (пальці NLF проти HaMeR, очищені канали проти сирих) використовують кеш-пакети конвеєра: перераховується лише підмінюваний канал. Конфігурації політик рота наведено у Таблиці 3.4.

Таблиця 3.4 – Політики рота для порівняльних експериментів

Політика	*w[jaw]*	*w[mouth]*	*Зміст*
P1 – рот лише з відео	0	0	аудіоканал не впливає на жоден регіон
P2 – ваги за замовчуванням	0,65	0,75	кероване поєднання двох джерел за Таблицею 2.5
P3 – рот лише з аудіо	1,0	1,0	щелепа і рот повністю від аудіоканалу

Третій принцип – стрес-модифікації входу. З порівняльного кліпу готуються дві контрольовані копії: з чорним прямокутником поверх обличчя у середній третині кліпу (повна оклюзія, відеоканал обличчя гарантовано втрачає спостереження) та з підміненою аудіодоріжкою – у кліп вставляється запис іншої фрази іншого мовця, причому зріз озвучення обирається за максимумом мовленнєвої енергії, а не позиційно. Кожна копія проходить той самий конвеєр у конфігурації обличчя й аудіо; порівняння вихідного і зіпсованого експерименту показує, як деградують довіри каналів і чи компенсує аудіоканал втрату відеоінформації.

Вимірюваними величинами слугують проксі-метрики з підрозділу 2.10: джитер за другою різницею і діапазон рухливості для пальців, кореляція jawOpen з RMS-огинаючою для артикуляції, частки кадрів з довірою вище порога для покриття, сімейства кривих при зміні ваг для керованості. Якісним доповненням є Blender-рендери персонажа для кожної конфігурації, виконані автоматизованим режимом доповнення Blender.

Жодна з метрик не порівнюється з еталонними даними, оскільки їх для тестових кліпів не існує; формулювання результатів обмежуються стабільністю, узгодженістю, покриттям і керованістю.

3.7 Результати експериментальної перевірки

Результати подано за порядком увімкнення каналів: тіло, обличчя з головою, пальці, очищення, аудіо-рот, регіональні політики, емоційний шар. Кожен крок відповідає на одне питання і спирається на власну пару порівняння. Оглядову карту експерименту дає рисунок 3.13. Під станами, які мають порівнюваний попередник з тією самою камерою, наведено підсилені вчетверо карти різниці (за методом рисунка 3.25): вони підсвічують саме те, що додав крок – кисті у S4, рот у S6 і S7, брови й очі у S8. Деталі кожного кроку розкривають окремі рисунки нижче, а повну поетапну збірку конвеєра в русі демонструє відеодоказ media/PVBuildup.mp4 у репозиторії роботи [53].

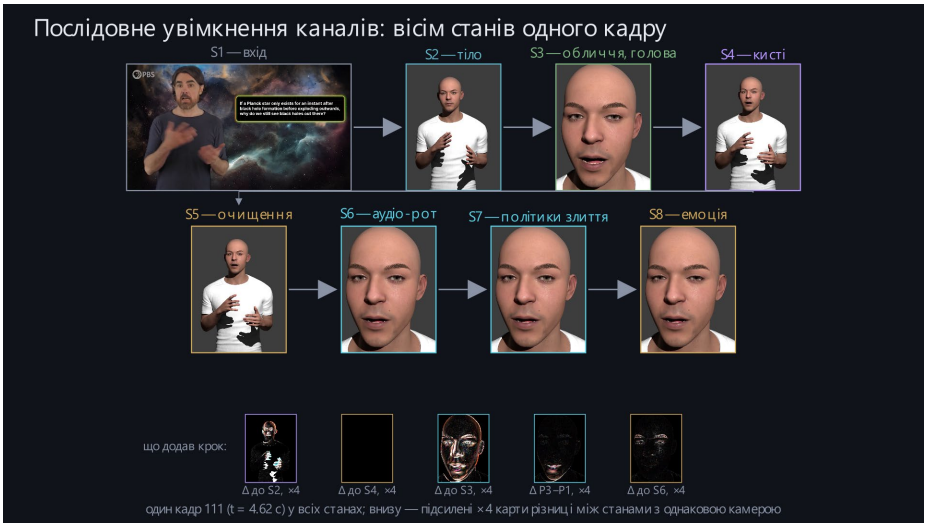

Рисунок 3.13 – Вісім станів системи на одному кадрі

Внесок каналу тіла. Після першого етапу конвеєра з’являється базова просторова поза: на вхідному кадрі видно скелетний оверлей NLF, у проміжному поданні формується SMPL-X-реконструкція, а у Blender персонаж уже повторює загальне положення корпуса, плечей і рук. Це підтверджує, що канал тіла задає каркас руху, на який пізніше накладаються інші модальності. Водночас цей стан ще не є повною анімацією персонажа: обличчя залишається майже статичним, рот закритим, а пальці майже нерухомими, оскільки повнотіла модель фізично не бачить достатньо пікселів кисті для стабільної реконструкції дрібних ланок. Саме ця різниця між загальною позою руки і деталізацією кисті обґрунтовує потребу в окремому каналі пальців. Стан системи після увімкнення тіла показано на рисунку 3.14.

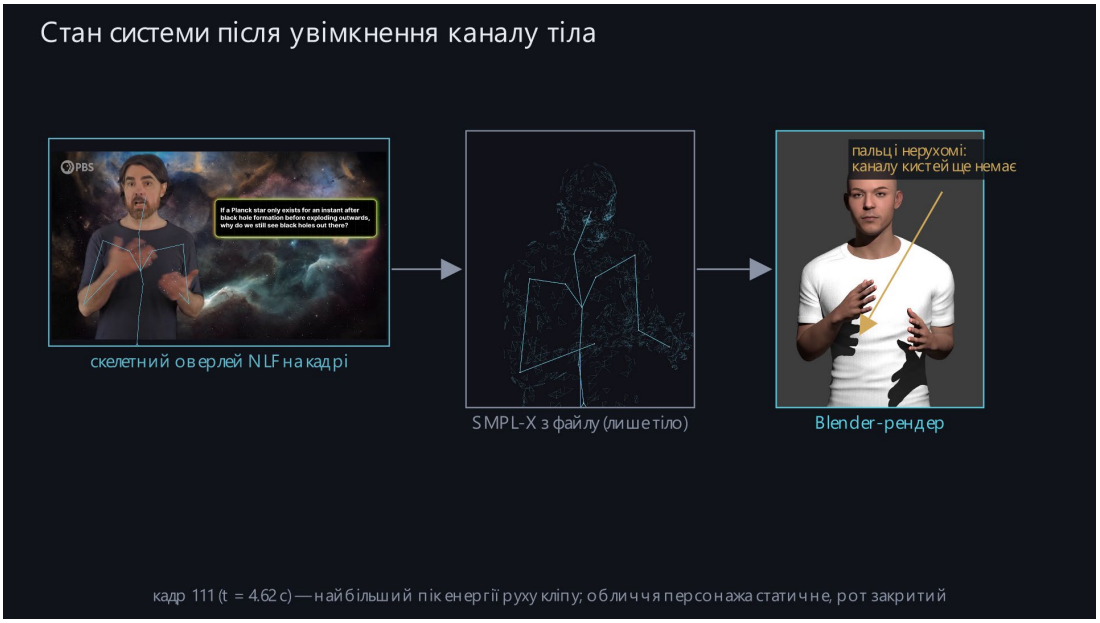

Рисунок 3.14 – Стан системи після увімкнення каналу тіла

Внесок каналів обличчя і голови. MediaPipe додає лицеві коефіцієнти і поворот голови: у редакторі стає активним viewport обличчя і доріжка довіри, на персонажі з’являється міміка і природний рух голови через окремий ланцюг. Рот на цьому етапі керується лише відео. Стан показано на рисунку 3.15.

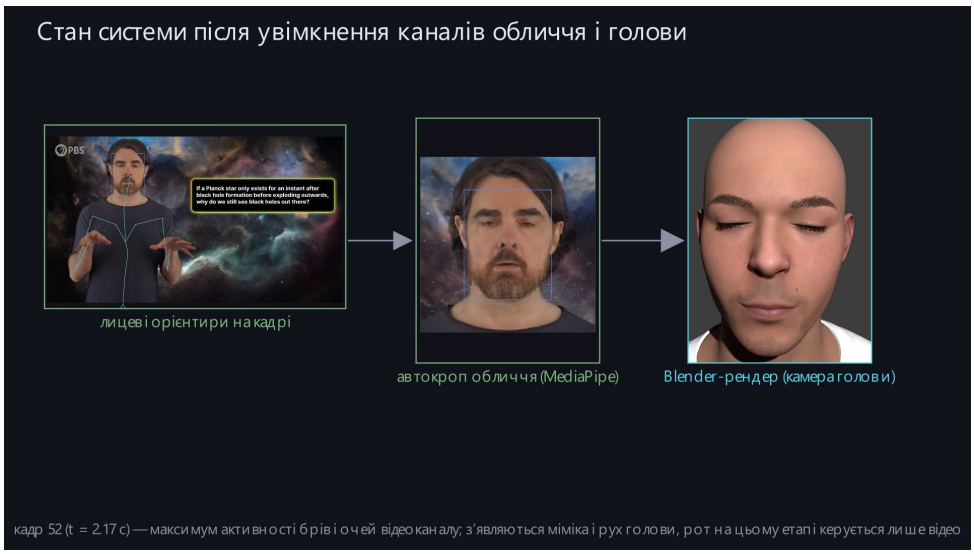

Рисунок 3.15 – Стан системи після увімкнення каналів обличчя і голови

Джерела пальців. Пара експериментів одного кліпу з каналом body.hands від NLF і від HaMeR дає головне кількісне порівняння роботи; на час порівняння фільтр довіри кистей вимкнено, щоб жодне відсікання не маскувало фактичну поведінку джерел. Причина різниці закладена в самій архітектурі моделей, і компонування рисунка 3.16 її відтворює.

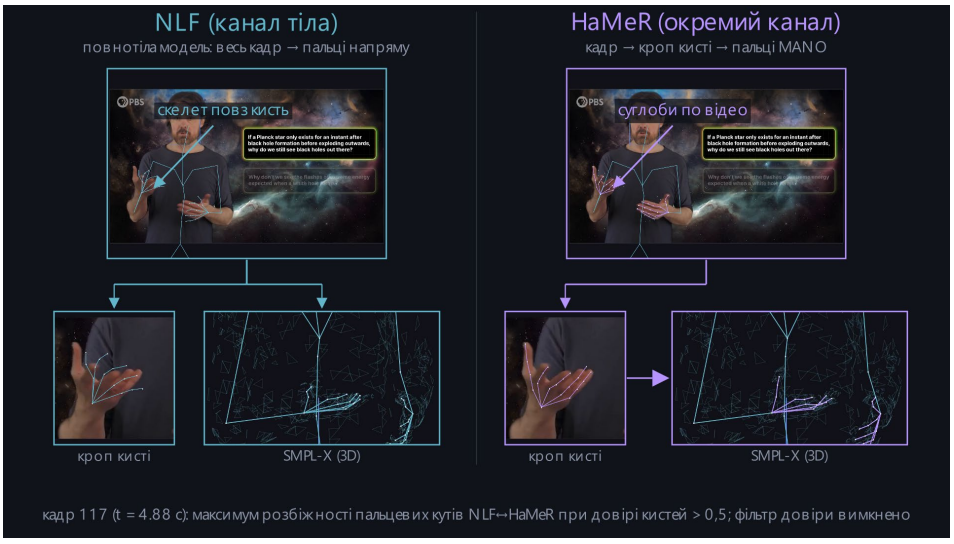

Рисунок 3.16 – Порівняння джерел пальців на одному кадрі кліпу

NLF – повнотіла модель: пальці оцінюються безпосередньо з повного кадру, де кисть займає близько одного відсотка площі, тому скелетна проєкція систематично не влучає у пальці на відео, а кроп кисті лише збільшує цей самий результат. HaMeR – окремий канал з власним кроком: з кадру вирізається кроп кисті, по ньому спеціалізована модель реконструює пальці MANO, і вже вони потрапляють у реконструкцію SMPL-X; на тому самому кадрі суглоби лягають точно по відео. Порівняльний кадр обрано програмно – як максимум середньої розбіжності пальцевих кутів двох джерел при довірі кистей понад 0,5.

Кількісно перевага виражається джитером за другою різницею за формулою (2.9): 10,16 градуса на кадр у квадраті у NLF проти 0,69 у HaMeR, тобто зниження приблизно у п’ятнадцять разів. Порівняння середніх значень показано на рисунку 3.17.

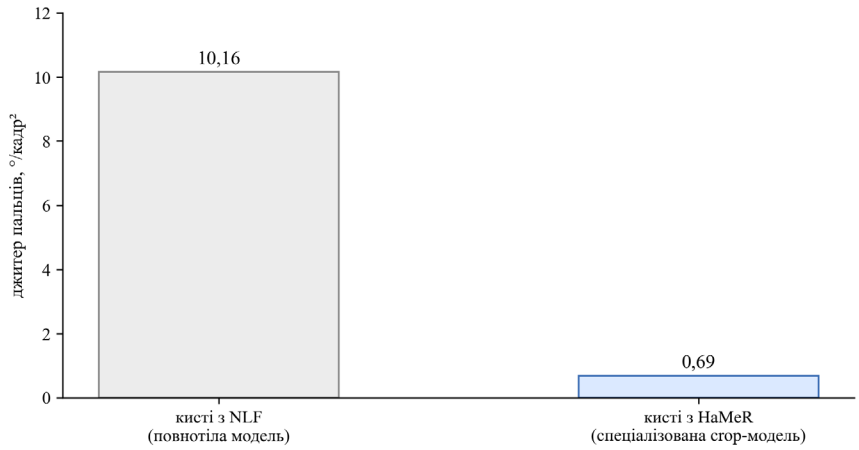

Рисунок 3.17 – Джитер пальців для двох джерел каналу body.hands

Стабільність зберігається не лише в середньому, а й у часі: покадрові криві джитера на всьому кліпі (рисунок 3.18) показують, що крива NLF постійно коливається у діапазоні десятків градусів, тоді як крива HaMeR утримується біля нуля без сплесків. Динаміку порівняння у русі демонструє відеодоказ media/PVHandsCompare.mp4 у репозиторії роботи [53]. Важливе розмежування: це порівняння якості спостережень у каналі; якість застосування до рига

обмежена згаданою механікою пальцевих ланцюгів і оцінюється якісно за рендерами.

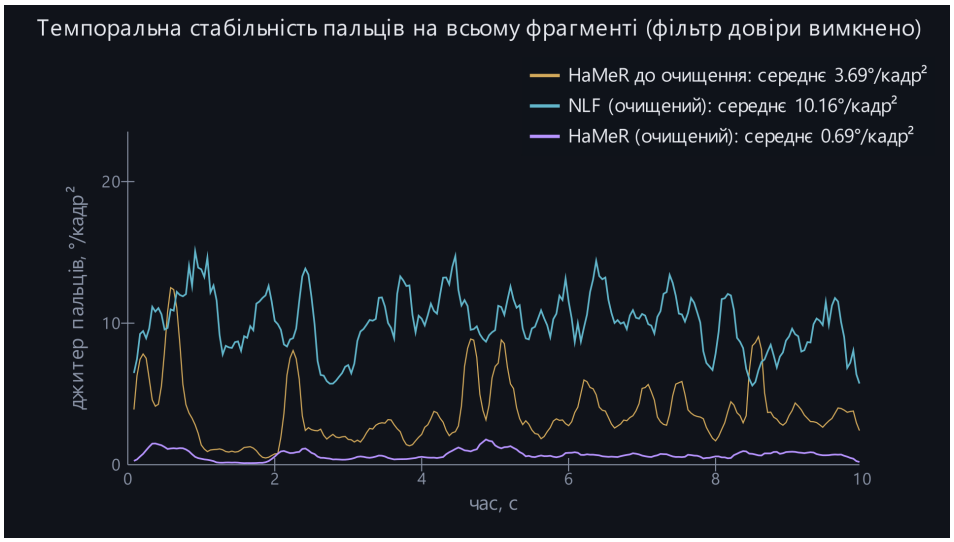

Рисунок 3.18 – Темпоральна стабільність пальців: покадровий джитер

Внесок очищення. Порівняння «сирі канали проти очищених» виконується на одному джерелі: сирі записи тіла і кистей беруться з кешів конвеєра до застосування алгоритму A2, очищені – з фінального файлу, тож різниця між ними зумовлена виключно очищенням, а не зміною моделі. Сукупний джитер каналів тіла і кистей (51 суглоб) знижується з 2,59 до 0,52 градуса на кадр у квадраті, тобто приблизно уп’ятеро, при збереженні форми руху: на рисунку 3.19 реконструкції сирого і очищеного каналу на піковому кадрі джитера збігаються позою, а криві внизу повторюють одна одну за обрисами, відрізняючись лише високочастотним тремтінням. Піковий кадр обрано програмно – як максимум згладженого сирого джитера. Очищення має і ціну: симетричне вікно здатне притлумити найшвидші рухи тривалістю один-два кадри.

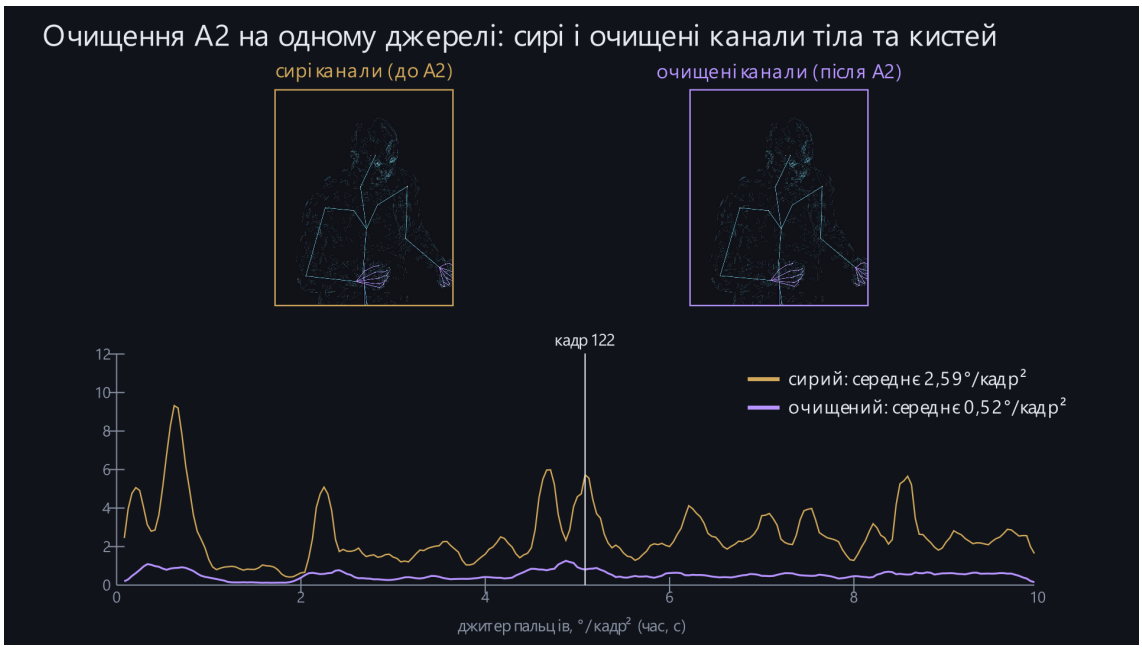

Рисунок 3.19 – Сирі і очищені канали тіла та кистей

Внесок аудіоканалу рота. Кореляція кривої jawOpen з RMS-огинаючою аудіо на порівняльному кліпі зростає з 0,068 у політиці P1, де рот формується лише з відеоканалу, до 0,228 у конфігурації з Audio2Face-3D та вагами за замовчуванням. Це означає, що артикуляція, отримана тільки з відео, на тестовому матеріалі слабко повторює ритм мовлення, тоді як аудіоканал відтворює його безпосередньо з акустичного сигналу. Різниця стає ще помітнішою у стресовому фрагменті з оклюзією обличчя: відеозалежна політика дає від'ємну кореляцію −0,073, тобто рух рота фактично втрачає зв'язок із мовленням, а конфігурація з Audio2Face-3D зберігає додатну кореляцію 0,297. Отже, аудіоканал не просто підсилює звичайний випадок, а виконує роль резервного джерела артикуляції тоді, коли відеоканал деградує. Значення для звичайних і стресових умов зведено на рисунку 3.20.

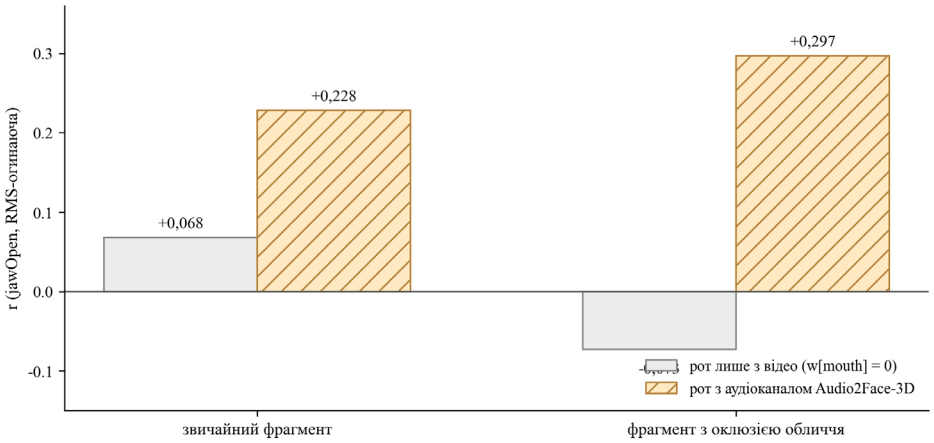

Рисунок 3.20 – Кореляція jawOpen з RMS у звичайних і стресових умовах

Поведінку політик P1–P3 на однакових кадрах ілюструє рисунок 3.21: файл один і той самий, змінюється лише вектор ваг споживача, і рот персонажа переходить від суто відеоформи через кероване поєднання до суто аудіоформи; у русі це порівняння показує відеодоказ media/PVMouth.mp4 [53].

Рисунок 3.21 – Порівняння політик рота P1–P3 на одному кадрі

Остаточним тестом є стрес-експеримент з оклюзією обличчя. У середній третині кліпу обличчя закрите чорним прямокутником: довіра відеоканалу для

регіону рота падає з одиниці до 0,022 у середньому по вікну оклюзії (75 з 81 кадру мають нульову довіру), тоді як довіра аудіоканалу залишається одиничною – мікрофон оклюзії не бачить. Наслідок для артикуляції: у політиці P1 кореляція jawOpen з мовленням падає до −0,073 (відеоканал видає шум або застиглу позу), тоді як з аудіоканалом вона не лише зберігається, а й зростає до 0,297. Цю пару станів показано на рисунку 3.22: на тому самому оклюзованому кадрі (програмний вибір – максимум добутку різниці артикуляцій і мовленнєвої енергії всередині вікна оклюзії) рот персонажа у конфігурації без аудіоканалу закритий і нерухомий (jawOpen = 0,00), а з аудіоканалом – відкритий у такт фразі (jawOpen = 0,29). Повну динаміку стрес-експерименту демонструє відеодоказ media/PVOcclusion.mp4 [53].

Рисунок 3.22 – Оклюзований кадр: рот персонажа без аудіоканалу і з ним

Механізм цієї стійкості у часі розкриває рисунок 3.23. Верхня панель – довіра регіону рота двох каналів: крива відеоканалу обвалюється до нуля на межі вікна оклюзії і відновлюється після нього, крива аудіоканалу – пряма лінія на одиниці. Нижня панель – крива jawOpen політики P1 проти злитої: у вікні оклюзії перша вироджується, друга продовжує слідувати RMS-огинаючій. Саме це і є

регіональне злиття у дії: вага джерела автоматично перетікає до каналу з вищою довірою, без жодної зміни файлу чи повторної обробки.

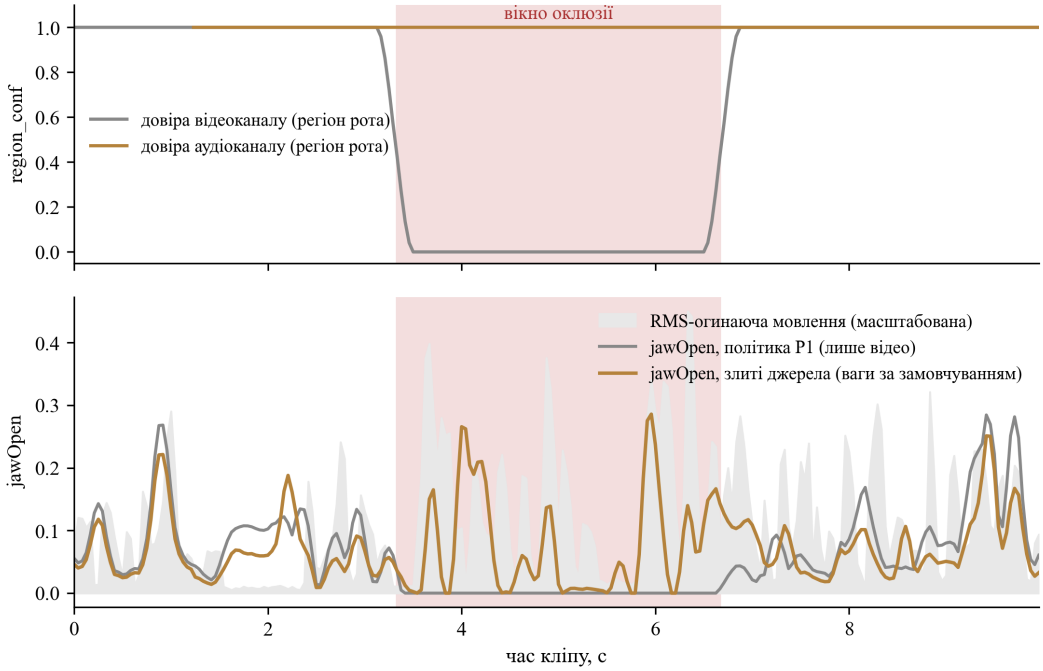

Рисунок 3.23 – Динаміка регіональної довіри і артикуляції при оклюзії обличчя

Експеримент з підміненою аудіодоріжкою доводить незалежність джерела (рисунок 3.24). Конструкція експерименту: відео не змінюється взагалі, замінюється лише аудіодоріжка на фрагмент іншої фрази іншого мовця; RMS-огинаючі двох доріжок некорельовані (r = −0,003). Контроль підтверджує чистоту умов – криві jawOpen політики P1 (рот лише з відео) у двох експериментах збігаються з r = 0,996, тобто відеочастина конвеєра детермінована. Натомість злиті криві рота розходяться (r = 0,736 між експериментами): аудіочастина артикуляції перейшла на підставлений запис, з яким злита крива корелює сильніше (0,252), ніж з рідним мовленням (0,196). Аудіоканал є самостійним спостереженням, а не похідною від відеоканалу.

Рисунок 3.24 – Незалежність джерела: однакове відео, різні аудіодоріжки

Емоційний шар. Рендери одного кадру з вагою впливу доріжки emotion_weight 0; 0,1 і 1,0 над одним файлом дають однакову базову міміку і різну емоційну модуляцію (рисунок 3.25); кадр для порівняння обирається програмно – як момент максимуму ненейтральної ймовірності у каналі emotion.audio (на тестовому кліпі це клас «здивування»). Оскільки емоційні рецепти охоплюють лише верхню частину обличчя, різниця між вагами навмисно делікатна, тому під рендерами наведено карти різниці: для ваги w береться попіксельний модуль різниці RGB-кадру з вагою w і кадру з вагою 0, помножений на чотири для видимості. Карти показують дві властивості шару одночасно: він діє виключно на брови й очі, а його внесок масштабується вагою – середня попіксельна різниця з кадром нульової ваги зростає з 1,0 рівня яскравості при w = 0,1 до 2,7 при w = 1,0. Вимкнення доріжки AIMOCAP_EMOTION повертає кадр у точності до стану без емоції, що підтверджує адитивність шару; артикуляція рота при цьому не змінюється, оскільки рецепти рота не містять; зміну ваги в русі показує відеодоказ media/PVEmotion.mp4 [53].

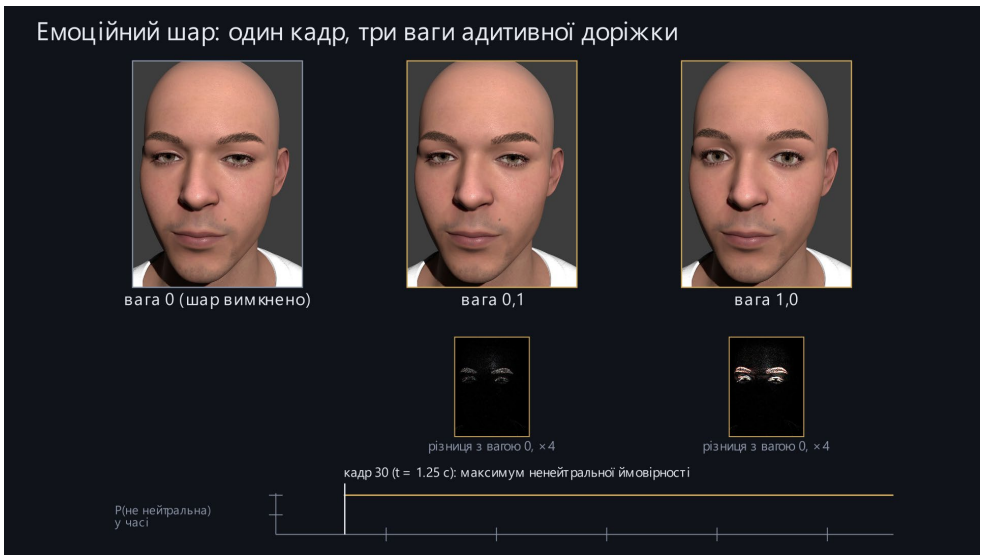

Рисунок 3.25 – Емоційний шар з вагою 0; 0,1 і 1,0

Підсумок абляції зведено у Таблиці 3.5. Вона узагальнює не всі проміжні зображення, а саме внесок кожного етапу в кінцеву поведінку системи.

Таблиця 3.5 – Підсумок послідовного увімкнення каналів

Крок	*Що додає*	*Що лишається обмеженням*	*Чим керується*
тіло (NLF + smplfitter)	поза, корінь, реконструкція SMPL-X	пальці майже нерухомі, обличчя статичне	вибір моделі тіла
обличчя і голова (MediaPipe)	міміка, поворот голови	рот вразливий до оклюзій і низької роздільності	довіри каналу, conf_floor
пальці (HaMeR)	артикуляція пальців, джитер 10,16 → 0,69 °/кадр²	зап'ястя лишається за каналом тіла	вибір джерела пальців
очищення (A2)	усунення тремтіння; джитер тіла і кистей 2,59 → 0,52 °/кадр² на одному джерелі	можливе притлумлення найшвидших рухів	вікна фільтра
аудіо-рот (Audio2Face-3D)	артикуляція, стійка до оклюзії; r 0,068 → 0,228	верх обличчя аудіо не відновлює	ваги w[jaw], w[mouth]
регіональні політики	кероване поєднання джерел без повторної обробки	–	вектор ваг w[r]
емоційний шар (A8)	адитивна емоційна модуляція	ймовірнісна оцінка емоцій без еталонної перевірки	emotion_weight, вимкнення доріжки

Для кожного каналу в таблиці вказано, що саме змінюється після його ввімкнення і який контрольний стан використано для порівняння. Такий формат дозволяє відокремити візуальний внесок тіла, кистей, обличчя, аудіорота та емоційного шару без змішування кількох причин в одному висновку. Окремо показано, що частина етапів впливає на геометрію руху, а частина – на керованість, стабільність або можливість редагування результату. Тому таблиця виконує роль компактної карти експерименту: вона пов’язує поетапні рисунки з числовими та якісними висновками підрозділу.

Системні вимірювання. Фактичні характеристики формату наведено у підрозділі 3.2: 592 КБ стисненого файлу на 10 секунд кліпу з сімома каналами. Часи обробки порівняльного кліпу зведено у Таблиці 3.6: повний експеримент усіх чотирьох модальностей займає 172 секунди, з яких понад 80 % припадає на канал тіла з пальцями (NLF і HaMeR – найважчі моделі конвеєра). Повторний експеримент того самого кліпу з тією самою конфігурацією завершується за 1,8 секунди – кеші проміжних результатів дають прискорення майже у сто разів, що й робить порівняльні експерименти з підміною одного каналу практично безкоштовними. Стресові копії обробляються у конфігурації обличчя й аудіо за приблизно 20 секунд кожна, оскільки для них не повторюється повний канал тіла з пальцями.

Таблиця 3.6 – Часи обробки порівняльного кліпу (10 с, 240 кадрів)

Етап	*Час, с*
декодування джерела	4,4
тіло і пальці (NLF + HaMeR)	140,4
обличчя і голова (MediaPipe)	21,3
рот з аудіо (Audio2Face-3D)	4,0
емоція (з оцінок Audio2Face-3D)	0,1
кодування файлу .aimocap	0,04
повний експеримент (разом)	171,8
повторний експеримент з кешами	1,8
стресова копія (обличчя й аудіо)	≈ 20

Порівняння з готовими рішеннями завершує перевірку. Перевага системи формулюється через властивості організації даних, кожну з яких підтверджено

конкретним експериментом цього підрозділу; заявка про вищу точність не робиться. Зведення наведено у Таблиці 3.7.

Таблиця 3.7 – Порівняння властивостей із готовими рішеннями

Властивість	*FreeMoCap*	*BlendArMocap*	*Audio2Face-3D*	*Запропонована система*
покриття чотирьох модальностей	–	–	–	+
відкритий проміжний формат каналів	–	–	–	+
кероване поєднання відео- і аудіоджерела обличчя	–	–	–	+ (ваги за регіонами)
незалежні редаговані шари на ризі	–	–	частково (MetaHuman)	+ (три NLA-доріжки)
діагностика окремих каналів	частково	–	–	+ (довіри, viewport-и)
заміна окремої моделі без зміни решти	–	–	–	+ (ролі каналів)
відтворюваність експерименту з файлу	–	–	–	+ (файл як одиниця експерименту)

3.8 Обмеження реалізованої системи

Окреслення меж застосовності є частиною перевірки. Обмеження зведено у Таблиці 3.8; для кожного вказано, як воно проявляється і яким механізмом системи діагностується або пом’якшується. Такий формат подання дозволяє розглядати кожне обмеження не ізольовано, а у зв’язку з конкретним способом його виявлення чи компенсації. Також завдяки цьому стає видно, які обмеження вже мають передбачений механізм реагування, а які потребують додаткового опрацювання.

Таблиця 3.8 – Обмеження системи та механізми їхньої діагностики

Обмеження	*Прояв*	*Діагностика або пом'якшення*
офлайн-обробка	система працює над записаним кліпом; реальний час не підтримується	обмеження зафіксоване у постановці задачі
одна людина у кадрі	конвеєр обробляє одного виконавця	діапазон обирається так, щоб у кадрі була одна особа
відсутність еталонних даних	кількісні оцінки є проксі-метриками	формулювання обмежені стабільністю, узгодженістю, покриттям, керованістю
еталонний адаптер перевірено лише на DAZ	інші програми або риги потребують власного адаптера і перевірки	формат і A4–A6 від них не залежать; A7 є контрактом адаптера
емоція лише з аудіо	відеоканал емоцій відсутній	роль emotion допускає другий канал без зміни формату
орієнтація зап'ястя з crop-моделі не використовується	зап'ястя успадковує якість каналу тіла	декомпозиція зафіксована у форматі каналів
залежність аудіоканалів від стека NVIDIA	сервіс потребує GPU NVIDIA, Docker і WSL	статус сервісу діагностується редактором; передбачено віземний резерв
некомерційні ліцензії ваг NLF і SMPL-X	публічне розповсюдження потребує переліцензування або заміни моделей	обмеження зафіксоване у постановці задачі

Окремо зазначимо, чого система не стверджує. Вона не претендує на вищу точність за сучасні моделі на академічних бенчмарках – власних вимірювань MPJPE чи PVE у роботі немає. Аудіоканал не замінює відеоканал обличчя: він підсилює регіони рота за вагами і компенсує деградацію відео. Розпізнавання емоцій залишається м'якою ймовірнісною оцінкою, придатною для адитивного шару, який можна вимкнути.

Висновки до розділу 3

У третьому розділі описано реалізований програмний комплекс і наведено результати його експериментальної перевірки. Комплекс складається з бібліотеки aimocap, настільного редактора на Qt і тестового вендорського

доповнення Blender/DAZ; важкі залежності ізольовано в окремий підпроцес (HaMeR) і контейнерний сервіс (Audio2Face-3D).

Формат .aimocap працює на реальних даних: десятисекундний тестовий кліп зберігається у 592 КБ із сімома різночастотними каналами, а інваріанти формату закріплені тестами кодування-декодування, регіонального розбиття і толерантності до невідомих каналів.

Producer-конвеєр послідовно запускає моделі під обмеження побутового GPU, кешує важкі етапи пакетами і автоматично калібрує часовий зсув потокового сервісу Audio2Face-3D. Редактор робить кожен канал спостережуваним через viewport-и, доріжки довіри і панель стану конвеєра; доповнення Blender розкладає результат на три незалежні доріжки нелінійної анімації.

Експериментальна перевірка побудована на фіксованому порівняльному кліпі, порівнянні політик без повторної обробки та стрес-модифікаціях входу. Показано внесок кожного каналу: спеціалізоване джерело пальців знижує джитер з 10,16 до 0,69 градуса на кадр у квадраті; аудіоканал рота підвищує кореляцію артикуляції з мовленням з 0,068 до 0,228 і утримує її на рівні 0,297 при повній оклюзії обличчя, де відеоканал падає до −0,073; емоційний шар додається і вимикається без сліду в базовій міміці.

Порівняння з готовими рішеннями сформульоване через властивості організації даних – покриття модальностей, відкритий проміжний формат, кероване злиття, незалежні шари, діагностованість, замінність моделей і відтворюваність – кожна з яких підтверджена окремим експериментом.

ВИСНОВКИ

Мету дипломної роботи – підвищення ефективності та доступності процесу створення анімації 3D-моделі людини з монокулярного відео і синхронного аудіо – досягнуто. Ефективність підвищено завдяки переходу від розрізненого запуску моделей і ручного перенесення результатів до канального процесу, у якому спостереження тіла, кистей, обличчя, аудіокерованого рота та емоцій зберігаються окремо, синхронізуються за часом і поєднуються на боці споживача за керованою політикою. Доступність підвищено тим, що обробка виконується на побутовій робочій станції з одним GPU, без маркерів, костюма, багатокамерної сцени або студійного павільйону.

У першому розділі проаналізовано предметну область за п’ятьма напрямами: поза тіла, поза кистей, лицеві рухи, аудіокерована анімація обличчя та розпізнавання емоцій. Для практичного конвеєра обґрунтовано використання NLF зі smplfitter для тіла, HaMeR для пальців із резервом кистей NLF, MediaPipe Face Landmarker для обличчя і голови, NVIDIA Audio2Face-3D для аудіокерованого рота й емоційного каналу. Огляд готових рішень показав, що наявні інструменти не забезпечують одночасно покриття потрібних модальностей, відкритий проміжний формат каналів, кероване злиття джерел і незалежне редагування шарів руху.

У другому розділі формалізовано канальну модель з шістьма каналами і п’ятьма ролями, шарову модель даних з межею файлу, самоописний формат .aimocap з рекомендованою політикою, три producer- і п’ять consumer-алгоритмів. Ключовим механізмом підвищення ефективності є регіональне злиття лицевих каналів за вектором довір семи анатомічних регіонів: воно дозволяє не перераховувати весь результат після зміни ваг, а змінювати спосіб використання вже збережених спостережень.

У третьому розділі експериментально підтверджено вплив запропонованої організації процесу на якість і повторюваність обробки. На фіксованому

десятисекундному кліпі спеціалізоване джерело пальців знижує джитер каналу кистей з 10,16 до 0,69 градуса на кадр у квадраті; очищення на одному джерелі знижує джитер тіла і кистей з 2,59 до 0,52; аудіоканал підвищує кореляцію артикуляції з мовленням з 0,068 до 0,228, а при повній оклюзії обличчя, де відеоканал падає до −0,073 і його довіра обвалюється до 0,022, утримує 0,297. Повторний експеримент того самого кліпу з кешами скорочується зі 171,8 с до 1,8 с, а десятисекундний файл із сімома каналами займає 592 КБ.

Практичне значення отриманих результатів полягає у зменшенні ручної роботи під час підготовки анімації: користувач отримує файл як одиницю експерименту, окремі канали спостережень, доріжки довіри, перегляд кожної модальності та можливість змінювати політику злиття без повторного запуску важких моделей. Еталонний адаптер Blender/DAZ показав, що результат переноситься у сцену як три незалежні доріжки нелінійної анімації: тіло, міміка та емоційний шар. Повний вихідний код і демонстраційні відеоматеріали розміщено у відкритому репозиторії.

Подальше підвищення ефективності можливе за рахунок відеоканалу емоцій з пізнім злиттям джерел, заміни HaMeR на WiLoR у спільному словнику MANO, профілів інших ригів, використання збережених VAD-скалярів, 3DMM-режиму високої якості, а також переліцензування компонентів з некомерційними вагами для публічного розповсюдження.

ПЕРЕЛІК ПОСИЛАНЬ

1. Vicon Motion Systems. Vicon Camera Systems: Valkyrie, Vanguard, Vero, Viper. URL: https://www.vicon.com/hardware/cameras/ (дата звернення: 01.06.2026).
2. Loper M., Mahmood N., Romero J. та ін. SMPL: A Skinned Multi-Person Linear Model. ACM Transactions on Graphics. 2015. Т. 34, № 6. № статті 248. DOI: 10.1145/2816795.2818013.
3. Cao Z., Hidalgo G., Simon T. та ін. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021. Т. 43, № 1. С. 172–186. DOI: 10.1109/TPAMI.2019.2929257.
4. Kanazawa A., Black M. J., Jacobs D. W. та ін. End-to-end Recovery of Human Shape and Pose. Матеріали IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 2018. С. 7122–7131. URL: https://arxiv.org/abs/1712.06584 (дата звернення: 01.06.2026).
5. Apple Inc. ARFaceAnchor.BlendShapeLocation. Apple Developer Documentation. URL: https://developer.apple.com/documentation/arkit/arfaceanchor/blendshapelocation (дата звернення: 01.06.2026).
6. Lugaresi C., Tang J., Nash H. та ін. MediaPipe: A Framework for Building Perception Pipelines: препринт. arXiv:1906.08172. 2019. URL: https://arxiv.org/abs/1906.08172 (дата звернення: 01.06.2026).
7. NVIDIA Corp. Audio2Face-3D: колекція моделей та інструментів. GitHub. URL: https://github.com/NVIDIA/Audio2Face-3D (дата звернення: 01.06.2026).
8. FreeMoCap Foundation. FreeMoCap: Free Motion Capture System. GitHub. URL: https://github.com/freemocap/freemocap (дата звернення: 01.06.2026).
9. Roetenberg D., Luinge H., Slycke P. Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors: техн. звіт. Enschede: Xsens Technologies, 2013.

10. Ekman P., Friesen W. V. Facial Action Coding System: A Technique for the Measurement of Facial Movement. Palo Alto, CA: Consulting Psychologists Press, 1978.
11. Kocabas M., Athanasiou N., Black M. J. VIBE: Video Inference for Human Body Pose and Shape Estimation. Матеріали CVPR. 2020. URL: https://arxiv.org/abs/1912.05656 (дата звернення: 01.06.2026).
12. Kocabas M., Huang C.-H. P., Hilliges O. та ін. PARE: Part Attention Regressor for 3D Human Body Estimation. Матеріали ICCV. 2021. URL: https://arxiv.org/abs/2104.08527 (дата звернення: 01.06.2026).
13. Fan Y., Lin Z., Saito J. та ін. FaceFormer: Speech-Driven 3D Facial Animation with Transformers. Матеріали CVPR. 2022. URL: https://arxiv.org/abs/2112.05329 (дата звернення: 01.06.2026).
14. Goel S., Pavlakos G., Rajasegaran J. та ін. Humans in 4D: Reconstructing and Tracking Humans with Transformers. Матеріали ICCV. 2023. URL: https://arxiv.org/abs/2305.20091 (дата звернення: 01.06.2026).
15. Shin S., Kim J., Halilaj E. та ін. WHAM: Reconstructing World-grounded Humans with Accurate 3D Motion. Матеріали CVPR. 2024. URL: https://arxiv.org/abs/2312.07531 (дата звернення: 01.06.2026).
16. Baradel F., Armando M., Galaaoui S. та ін. Multi-HMR: Multi-Person Whole-Body Human Mesh Recovery in a Single Shot. Матеріали ECCV. 2024. URL: https://arxiv.org/abs/2402.14654 (дата звернення: 01.06.2026).
17. Sárándi I., Pons-Moll G. Neural Localizer Fields for Continuous 3D Human Pose and Shape Estimation. Advances in Neural Information Processing Systems (NeurIPS). 2024. URL: https://arxiv.org/abs/2407.07532 (дата звернення: 01.06.2026).
18. Cai Z., Yin W., Zeng A. та ін. SMPLer-X: Scaling Up Expressive Human Pose and Shape Estimation. NeurIPS Datasets and Benchmarks. 2023. URL: https://arxiv.org/abs/2309.17448 (дата звернення: 01.06.2026).

19. Pavllo D., Feichtenhofer C., Grangier D. та ін. 3D Human Pose Estimation in Video with Temporal Convolutions and Semi-Supervised Training. Матеріали CVPR. 2019. URL: https://arxiv.org/abs/1811.11742 (дата звернення: 01.06.2026).
20. Zhu W., Ma X., Liu Z. та ін. MotionBERT: A Unified Perspective on Learning Human Motion Representations. Матеріали ICCV. 2023. URL: https://arxiv.org/abs/2210.06551 (дата звернення: 01.06.2026).
21. Li T., Bolkart T., Black M. J. та ін. Learning a model of facial shape and expression from 4D scans (FLAME). ACM Transactions on Graphics. 2017. Т. 36, № 6. № статті 194. DOI: 10.1145/3130800.3130813.
22. Daněček R., Black M. J., Bolkart T. EMOCA: Emotion Driven Monocular Face Capture and Animation. Матеріали CVPR. 2022. URL: https://arxiv.org/abs/2204.11312 (дата звернення: 01.06.2026).
23. Zielonka W., Bolkart T., Thies J. Towards Metrical Reconstruction of Human Faces (MICA). Матеріали ECCV. 2022. URL: https://arxiv.org/abs/2204.06607 (дата звернення: 01.06.2026).
24. Filntisis P. P., Retsinas G., Paraperas-Papantoniou F. та ін. Visual Speech-Aware Perceptual 3D Facial Expression Reconstruction from Videos (SPECTRE): препринт. arXiv:2207.11094. 2022. URL: https://arxiv.org/abs/2207.11094 (дата звернення: 01.06.2026).
25. Cudeiro D., Bolkart T., Laidlaw C. та ін. Capture, Learning, and Synthesis of 3D Speaking Styles (VOCA). Матеріали CVPR. 2019. URL: https://arxiv.org/abs/1905.03079 (дата звернення: 01.06.2026).
26. Richard A., Zollhöfer M., Wen Y. та ін. MeshTalk: 3D Face Animation from Speech using Cross-Modality Disentanglement. Матеріали ICCV. 2021. URL: https://arxiv.org/abs/2104.08223 (дата звернення: 01.06.2026).
27. Xing J., Xia M., Zhang Y. та ін. CodeTalker: Speech-Driven 3D Facial Animation with Discrete Motion Prior. Матеріали CVPR. 2023. URL: https://arxiv.org/abs/2301.02379 (дата звернення: 01.06.2026).

28. Peng Z., Wu H., Song Z. та ін. EmoTalk: Speech-Driven Emotional Disentanglement for 3D Face Animation. Матеріали ICCV. 2023. URL: https://arxiv.org/abs/2303.11089 (дата звернення: 01.06.2026).
29. Zhang W., Cun X., Wang X. та ін. SadTalker: Learning Realistic 3D Motion Coefficients for Stylized Audio-Driven Single Image Talking Face Animation. Матеріали CVPR. 2023. URL: https://arxiv.org/abs/2211.12194 (дата звернення: 01.06.2026).
30. Pepino L., Riera P., Ferrer L. Emotion Recognition from Speech Using wav2vec 2.0 Embeddings. Матеріали Interspeech. 2021. С. 3400–3404. DOI: 10.21437/Interspeech.2021-703.
31. Savchenko A. V. HSEmotion: high-speed emotion recognition library. Software Impacts. 2022. Т. 14. № статті 100433. DOI: 10.1016/j.simpa.2022.100433.
32. Atrey P. K., Hossain M. A., El Saddik A. та ін. Multimodal fusion for multimedia analysis: a survey. Multimedia Systems. 2010. Т. 16, № 6. С. 345–379. DOI: 10.1007/s00530-010-0182-0.
33. Bodyanskiy Ye., Zaychenko Yu., Hamidov G. та ін. Multilayer GMDH-neuro-fuzzy network based on extended neo-fuzzy neurons and its application in online facial expression recognition. System Research and Information Technologies. 2020. № 3. С. 66–78. DOI: 10.20535/SRIT.2308-8893.2020.3.05.
34. Casiez G., Roussel N., Vogel D. 1€ Filter: A Simple Speed-Based Low-Pass Filter for Noisy Input in Interactive Systems. Матеріали ACM CHI. 2012. С. 2527–2530. DOI: 10.1145/2207676.2208639.
35. Savitzky A., Golay M. J. E. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry. 1964. Т. 36, № 8. С. 1627–1639. DOI: 10.1021/ac60214a047.
36. Pavlakos G., Choutas V., Ghorbani N. та ін. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image (SMPL-X). Матеріали IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 2019. С. 10975–10985. URL: https://arxiv.org/abs/1904.05866 (дата звернення: 01.06.2026).

37. Google AI Edge. Face landmark detection guide: MediaPipe Solutions. URL: https://developers.google.com/edge/mediapipe/solutions/vision/face_landmarker (дата звернення: 01.06.2026).
38. Google AI Edge. Pose landmark detection guide: MediaPipe Solutions. URL: https://developers.google.com/edge/mediapipe/solutions/vision/pose_landmarker (дата звернення: 01.06.2026).
39. cgtinker. BlendArMocap: realtime motion tracking in Blender using MediaPipe and Rigify. GitHub. URL: https://github.com/cgtinker/BlendArMocap (дата звернення: 01.06.2026).
40. Digital-Standard Co., Ltd. ThreeDPoseTracker. GitHub. URL: https://github.com/digital-standard/ThreeDPoseTracker (дата звернення: 01.06.2026).
41. Qammaz A., Argyros A. A. MocapNET: Ensemble of SNN Encoders for 3D Human Pose Estimation in RGB Images. GitHub. URL: https://github.com/FORTH-ModelBasedTracker/MocapNET (дата звернення: 01.06.2026).
42. Chung C., Fedorov I., Huang M. та ін. Audio2Face-3D: Audio-driven Realistic Facial Animation For Digital Avatars: препринт. arXiv:2508.16401. 2025. URL: https://arxiv.org/abs/2508.16401 (дата звернення: 01.06.2026).
43. Sárándi I. SMPLFitter: The Fast Way From Vertices to Parametric 3D Humans. GitHub. URL: https://github.com/isarandi/smplfitter (дата звернення: 01.06.2026).
44. Romero J., Tzionas D., Black M. J. Embodied Hands: Modeling and Capturing Hands and Bodies Together. ACM Transactions on Graphics. 2017. Т. 36, № 6. № статті 245. DOI: 10.1145/3130800.3130883.
45. Pavlakos G., Shan D., Radosavovic I. та ін. Reconstructing Hands in 3D with Transformers (HaMeR). Матеріали IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR). 2024. URL: https://arxiv.org/abs/2312.05251 (дата звернення: 01.06.2026).
46. Potamias R. A., Zhang J., Deng J. та ін. WiLoR: End-to-end 3D Hand Localization and Reconstruction in-the-wild. Матеріали IEEE/CVF Conf. on Computer Vision

and Pattern Recognition (CVPR). 2025. URL: https://arxiv.org/abs/2409.12259 (дата звернення: 01.06.2026).

47. Baevski A., Zhou H., Mohamed A. та ін. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. Advances in Neural Information Processing Systems (NeurIPS). 2020. URL: https://arxiv.org/abs/2006.11477 (дата звернення: 01.06.2026).
48. Russell J. A. A circumplex model of affect. Journal of Personality and Social Psychology. 1980. Т. 39, № 6. С. 1161–1178. DOI: 10.1037/h0077714.
49. Shoemake K. Animating rotation with quaternion curves. Матеріали SIGGRAPH. 1985. С. 245–254. DOI: 10.1145/325334.325242.
50. Collet Y., Kucherawy M. Zstandard Compression and the application/zstd Media Type: RFC 8878. IETF, 2021. URL: https://www.rfc-editor.org/rfc/rfc8878 (дата звернення: 01.06.2026).
51. Feng Y., Feng H., Black M. J. та ін. Learning an Animatable Detailed 3D Face Model from In-the-Wild Images (DECA). ACM Transactions on Graphics. 2021. Т. 40, № 4. № статті 88. DOI: 10.1145/3450626.3459936.
52. PBS Space Time. We Thought Black Holes Ended in Singularities. They Might End In a Frozen Big Bang. YouTube, 2026. URL: https://www.youtube.com/watch?v=Wu8xNx4njoM (дата звернення: 01.06.2026).
53. AI MoCap: багатоканальна система захоплення руху з відео – вихідний код і демонстраційні відеоматеріали (тека media). Репозиторій GitHub. URL: https://github.com/CR0W33/aimocap2026 (дата звернення: 01.06.2026).

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Повний вихідний код програмного комплексу опубліковано у відкритому репозиторії [53]. У цьому додатку наведено вибрані компактні файли саме бібліотеки aimocap, які є найважливішими для формату, обробки, очищення, синхронізації та злиття каналів.

Нижче наведено ключові компактні файли бібліотеки aimocap, які безпосередньо реалізують роботу з моделями, джерелом даних, кешуванням, очищенням, синхронізацією та злиттям каналів. Великі інтеграційні модулі конвеєра, менеджера Audio2Face-3D та користувацьких інтерфейсів у додатку не дублюються; вони доступні у репозиторії роботи.

Файл aimocap/models.py відповідає за пошук і підготовку локальних файлів моделей SMPL-X та MediaPipe. Він ізолює залежність конвеєра від конкретного розташування моделей на диску.

Файл: aimocap/models.py:

```
from __future__ import annotations

from dataclasses import dataclass
from pathlib import Path
import os
import shutil


@dataclass(frozen=True, slots=True)
class ModelPaths:
    root: Path
    smplx_neutral: Path | None
    face_landmarker: Path | None


def find_smplx_neutral(models_root: str | Path = "models") -> Path | None:
    root = Path(models_root)
    candidates = [
        root / "body_models" / "smplx" / "SMPLX_NEUTRAL.npz",
        root / "body_models" / "SMPLX_NEUTRAL.npz",
        root / "smplx" / "SMPLX_NEUTRAL.npz",
        root / "SMPLX_NEUTRAL.npz",
    ]
    data_root = os.environ.get("DATA_ROOT")
    if data_root:
        candidates.insert(0, Path(data_root) / "body_models" / "smplx" /
"SMPLX_NEUTRAL.npz")
        candidates.insert(1, Path(data_root) / "body_models" / "SMPLX_NEUTRAL.npz")
    for candidate in candidates:
```

```
        if candidate.exists():
            return candidate.resolve()
    return None


def require_smplx_neutral(models_root: str | Path = "models") -> Path:
    found = find_smplx_neutral(models_root)
    if found is None:
        raise FileNotFoundError(
            "SMPLX_NEUTRAL.npz was not found. Place it in "
            "models/body_models/smplx/SMPLX_NEUTRAL.npz or
models/body_models/SMPLX_NEUTRAL.npz."
        )
    return found


def prepare_smplfitter_layout(models_root: str | Path = "models") -> Path:
    neutral = require_smplx_neutral(models_root)
    target_dir = Path(models_root) / "body_models" / "smplx"
    target_dir.mkdir(parents=True, exist_ok=True)
    target = target_dir / "SMPLX_NEUTRAL.npz"
    if not target.exists():
        try:
            os.link(neutral, target)
        except OSError:
            shutil.copy2(neutral, target)
    os.environ.setdefault("DATA_ROOT", str(Path(models_root).resolve()))
    os.environ.setdefault("SMPLFITTER_BODY_MODELS", str((Path(models_root) /
"body_models").resolve()))
    return target.resolve()


def discover_models(models_root: str | Path = "models") -> ModelPaths:
    root = Path(models_root)
    face_candidates = [
        root / "face_landmarker.task",
        root / "face_landmarker_v2_with_blendshapes.task",
    ]
    face = next((path.resolve() for path in face_candidates if path.exists()), None)
    return ModelPaths(root=root, smplx_neutral=find_smplx_neutral(root),
face_landmarker=face)
```

Файл aimocap/source.py містить роботу з вхідним відеоджерелом: визначення параметрів контейнера, декодування відеокадрів і аудіодоріжки з урахуванням частоти кадрів та повороту.

Файл: aimocap/source.py:

```
from __future__ import annotations

from dataclasses import dataclass
from pathlib import Path
from typing import Iterator

import numpy as np


class SourceError(Exception):
```

```
    pass


@dataclass(slots=True)
class SourceInfo:
    path: str
    width: int
    height: int
    fps: float
    duration_s: float
    has_audio: bool
    audio_sample_rate: int | None
    variable_frame_rate: bool
    rotation_deg: int


def probe(path: str | Path) -> SourceInfo:
    try:
        import av
    except ImportError as exc:
        raise SourceError("PyAV is required. Run `uv sync --extra app`.") from exc
    try:
        with av.open(str(path)) as container:
            if not container.streams.video:
                raise SourceError("source has no video stream")
            stream = container.streams.video[0]
            fps_raw = float(stream.average_rate or stream.guessed_rate or 30.0)
            duration_s = 0.0
            if stream.duration is not None:
                duration_s = float(stream.duration * stream.time_base)
            elif container.duration:
                duration_s = float(container.duration / 1_000_000)
            audio = container.streams.audio[0] if container.streams.audio else None
            return SourceInfo(
                path=str(path),
                width=int(stream.codec_context.width),
                height=int(stream.codec_context.height),
                fps=float(np.clip(round(fps_raw), 24.0, 60.0)),
                duration_s=duration_s,
                has_audio=audio is not None,
                audio_sample_rate=int(audio.sample_rate) if audio is not None else
None,
                variable_frame_rate=bool(
                    stream.average_rate and stream.guessed_rate and stream.average_rate
!= stream.guessed_rate
                ),
                rotation_deg=_rotation_deg(stream),
            )
    except SourceError:
        raise
    except Exception as exc:
        raise SourceError(str(exc)) from exc


def decode_video(path: str | Path) -> Iterator[tuple[float, np.ndarray]]:
    import av

    with av.open(str(path)) as container:
        stream = container.streams.video[0]
        rotation = _rotation_deg(stream)
        first_pts: float | None = None
        for frame in container.decode(stream):
```

```
            if frame.pts is None:
                continue
            pts_s = float(frame.pts * stream.time_base)
            if first_pts is None:
                first_pts = pts_s
            img = frame.to_ndarray(format="rgb24")
            if rotation:
                img = _rotate(img, rotation)
            yield pts_s - first_pts, img


def decode_audio(path: str | Path, target_sr: int = 16000) -> tuple[np.ndarray, int,
float]:
    import av

    chunks: list[np.ndarray] = []
    t0_s: float | None = None
    with av.open(str(path)) as container:
        if not container.streams.audio:
            return np.zeros(0, dtype=np.float32), target_sr, 0.0
        stream = container.streams.audio[0]
        resampler = av.AudioResampler(format="flt", layout="mono", rate=target_sr)
        for frame in container.decode(stream):
            if t0_s is None and frame.pts is not None:
                t0_s = float(frame.pts * stream.time_base)
            frames = resampler.resample(frame)
            if not isinstance(frames, list):
                frames = [frames]
            for out in frames:
                arr = out.to_ndarray()
                if arr.ndim == 2:
                    arr = arr.mean(axis=0)
                chunks.append(np.asarray(arr, dtype=np.float32).reshape(-1))
        tail = resampler.resample(None)
        if tail:
            for out in tail:
                arr = out.to_ndarray()
                if arr.ndim == 2:
                    arr = arr.mean(axis=0)
                chunks.append(np.asarray(arr, dtype=np.float32).reshape(-1))
    if not chunks:
        return np.zeros(0, dtype=np.float32), target_sr, 0.0
    audio = np.concatenate(chunks).astype(np.float32)
    audio = np.clip(audio, -1.0, 1.0)
    return audio, target_sr, float(t0_s or 0.0)


def read_video_frames(path: str | Path) -> list[tuple[float, np.ndarray]]:
    return list(decode_video(path))


def _rotation_deg(stream) -> int:
    rotate = 0
    metadata = getattr(stream, "metadata", {}) or {}
    if "rotate" in metadata:
        try:
            rotate = int(metadata["rotate"]) % 360
        except ValueError:
            rotate = 0
    return rotate if rotate in {0, 90, 180, 270} else 0
```

```
def _rotate(img: np.ndarray, rotation: int) -> np.ndarray:
    if rotation == 90:
        return np.rot90(img, k=3).copy()
    if rotation == 180:
        return np.rot90(img, k=2).copy()
    if rotation == 270:
        return np.rot90(img, k=1).copy()
    return img
```

Файл aimocap/cache.py реалізує кеш проміжних результатів за підписом джерела, конфігурації потоку, параметрів очищення і часового діапазону. Це дає змогу повторно запускати експерименти без повної переобробки відео.

Файл: aimocap/cache.py:

```
from __future__ import annotations

from pathlib import Path
import hashlib
import json
import os
from typing import Any

import numpy as np

from aimocap.config import Config


def cache_root(config: Config) -> Path:
    return Path(str(config.cache.get("dir", ".aimocap_cache")))


def source_signature(video_path: str | Path) -> dict[str, Any]:
    path = Path(video_path)
    stat = path.stat()
    return {
        "path": str(path.resolve()),
        "size": int(stat.st_size),
        "mtime_ns": int(stat.st_mtime_ns),
    }


def stream_cache_key(
    video_path: str | Path,
    stream_name: str,
    stream_config: dict,
    clean_config: dict,
    t_start: float,
    t_end: float,
) -> str:
    public_stream_config = {key: value for key, value in stream_config.items() if not
str(key).startswith("_")}
    payload = {
        "source": source_signature(video_path),
        "stream": stream_name,
        "stream_config": public_stream_config,
        "clean": clean_config,
        "range": [round(float(t_start), 6), round(float(t_end), 6)],
```

```
        "cache_version": 3,
    }
    data = json.dumps(payload, sort_keys=True, separators=(",", ":"),
default=str).encode("utf-8")
    return hashlib.sha256(data).hexdigest()[:24]


def load_stream_cache(root: Path, key: str) -> np.ndarray | None:
    path = root / f"{key}.npy"
    if not path.exists():
        return None
    return np.load(path, allow_pickle=False)


def load_stream_cache_meta(root: Path, key: str) -> dict[str, Any]:
    path = root / f"{key}.json"
    if not path.exists():
        return {}
    with path.open("r", encoding="utf-8") as handle:
        data = json.load(handle)
    return data if isinstance(data, dict) else {}


def save_stream_cache(root: Path, key: str, records: np.ndarray) -> Path:
    root.mkdir(parents=True, exist_ok=True)
    path = root / f"{key}.npy"
    tmp = path.with_suffix(".tmp.npy")
    np.save(tmp, records, allow_pickle=False)
    os.replace(tmp, path)
    return path


def save_stream_cache_meta(root: Path, key: str, meta: dict[str, Any]) -> Path:
    root.mkdir(parents=True, exist_ok=True)
    path = root / f"{key}.json"
    tmp = path.with_suffix(".tmp.json")
    with tmp.open("w", encoding="utf-8") as handle:
        json.dump(meta, handle, indent=2, sort_keys=True)
    os.replace(tmp, path)
    return path


def save_cache_source_index(root: Path, key: str, video_path: str | Path, *, kind: str)
-> Path:
    root.mkdir(parents=True, exist_ok=True)
    path = root / f"{key}.source.json"
    tmp = path.with_suffix(".tmp.json")
    payload = {
        "source": source_signature(video_path),
        "key": str(key),
        "kind": str(kind),
    }
    with tmp.open("w", encoding="utf-8") as handle:
        json.dump(payload, handle, indent=2, sort_keys=True)
    os.replace(tmp, path)
    return path


def clear_cache_for_source(root: str | Path, video_path: str | Path) -> list[Path]:
    base = Path(root)
    if not base.exists():
        return []
```

```
    wanted = source_signature(video_path)
    removed: list[Path] = []
    for manifest in list(base.rglob("*.source.json")):
        try:
            with manifest.open("r", encoding="utf-8") as handle:
                payload = json.load(handle)
        except Exception:
            continue
        if payload.get("source") != wanted:
            continue
        key = str(payload.get("key") or manifest.name[: -len(".source.json")])
        for candidate in _cache_files_for_key(manifest.parent, key):
            if candidate.exists():
                candidate.unlink()
                removed.append(candidate)
        if manifest.exists():
            manifest.unlink()
            removed.append(manifest)
    return removed


def remove_cache_entry(root: str | Path, key: str) -> list[Path]:
    removed: list[Path] = []
    for candidate in _cache_files_for_key(Path(root), str(key)):
        if candidate.exists():
            candidate.unlink()
            removed.append(candidate)
    return removed


def _cache_files_for_key(root: Path, key: str) -> list[Path]:
    return [
        root / f"{key}.npy",
        root / f"{key}.npz",
        root / f"{key}.audio2face.npz",
        root / f"{key}.debug.npy",
        root / f"{key}.json",
        root / f"{key}.source.json",
    ]
```

Файл aimocap/clean/filters.py містить базові процедури очищення часових рядів: заповнення коротких розривів за довірою та згладжування фільтром Савицького–Голея.

Файл: aimocap/clean/filters.py:

```
from __future__ import annotations

import numpy as np
from scipy.signal import savgol_filter


def fill_small_gaps(times: np.ndarray, values: np.ndarray, conf: np.ndarray, gap_max_s:
float) -> np.ndarray:
    out = np.asarray(values, dtype=np.float32).copy()
    ts = np.asarray(times, dtype=np.float64)
    good = np.asarray(conf, dtype=np.float32) > 0.0
    bad_indices = np.flatnonzero(~good)
```

```
    if bad_indices.size == 0:
        return out
    for idx in bad_indices:
        prev = idx - 1
        while prev >= 0 and not good[prev]:
            prev -= 1
        nxt = idx + 1
        while nxt < len(good) and not good[nxt]:
            nxt += 1
        if prev >= 0 and nxt < len(good) and ts[nxt] - ts[prev] <= gap_max_s:
            alpha = (ts[idx] - ts[prev]) / max(ts[nxt] - ts[prev], 1e-12)
            out[idx] = (1.0 - alpha) * out[prev] + alpha * out[nxt]
    return out


def savgol_clean(values: np.ndarray, window: int = 7, poly: int = 2) -> np.ndarray:
    arr = np.asarray(values, dtype=np.float32)
    if arr.shape[0] < 3:
        return arr.copy()
    win = int(window)
    if win % 2 == 0:
        win += 1
    win = min(win, arr.shape[0] if arr.shape[0] % 2 else arr.shape[0] - 1)
    if win <= poly:
        return arr.copy()
    return savgol_filter(arr, window_length=win, polyorder=int(poly), axis=0,
mode="interp").astype(
        np.float32
    )
```

Файл aimocap/clean/apply.py застосовує очищення до структурованих каналів AIMOCAP. Для кватерніонів та подань вісь-кут він виконує згладжування через нормалізовані обертання, щоб не ламати геометрію руху.

Файл: aimocap/clean/apply.py:

```
from __future__ import annotations

import numpy as np

from aimocap.geometry import axis_angle_to_quat, quat_normalize, quat_to_axis_angle

from .filters import fill_small_gaps, savgol_clean


def clean_channel(records: np.ndarray, cfg: dict) -> np.ndarray:
    names = records.dtype.names or ()
    if len(records) < 3 or ("conf" not in names and "region_conf" not in names):
        return records
    out = records.copy()
    gap_max_s = float(cfg.get("gap_max_s", 0.5))
    window = int(cfg.get("savgol_window", 7))
    poly = int(cfg.get("savgol_poly", 2))
    rotation_window = int(cfg.get("rotation_savgol_window", max(window, 9)))
    hands_window = int(cfg.get("hands_savgol_window", max(rotation_window, 11)))
    if "conf" in names:
        conf = np.asarray(out["conf"], dtype=np.float32)
    else:
```

```
        conf = np.max(np.asarray(out["region_conf"], dtype=np.float32), axis=1)
    times = np.asarray(out["t"], dtype=np.float64)
    for name in out.dtype.names or ():
        if name in {"t", "conf"}:
            continue
        values = out[name]
        if not np.issubdtype(values.dtype, np.floating):
            continue
        if name in {"root_q", "head_q"}:
            out[name] = _clean_quaternions(values, window=rotation_window, poly=poly)
            continue
        if name in {"body_pose", "left", "right", "hands", "jaw", "eyes"} and
values.shape[-1] % 3 == 0:
            aa_window = hands_window if name in {"left", "right", "hands"} else
rotation_window
            out[name] = _clean_axis_angle_set(values, window=aa_window, poly=poly)
            continue
        flat = values.reshape((len(out), -1))
        filled = fill_small_gaps(times, flat, conf, gap_max_s=gap_max_s)
        cleaned = savgol_clean(filled, window=window, poly=poly)
        out[name] = cleaned.reshape(values.shape)
    if "conf" in names:
        out["conf"] = np.clip(savgol_clean(conf[:, None], window=window, poly=min(poly,
1)).reshape(-1), 0.0, 1.0)
    if "region_conf" in names:
        out["region_conf"] = np.clip(
            savgol_clean(np.asarray(out["region_conf"], dtype=np.float32),
window=window, poly=min(poly, 1)),
            0.0,
            1.0,
        )
    return out


def _clean_quaternions(values: np.ndarray, window: int, poly: int) -> np.ndarray:
    q = quat_normalize(np.asarray(values, dtype=np.float32))
    q = _unroll_quaternion_signs(q)
    smoothed = savgol_clean(q.reshape((len(q), -1)), window=window,
poly=poly).reshape(q.shape)
    return quat_normalize(smoothed).astype(np.float32)


def _clean_axis_angle_set(values: np.ndarray, window: int, poly: int) -> np.ndarray:
    original_shape = values.shape
    aa = np.asarray(values, dtype=np.float32).reshape((len(values), -1, 3))
    q = axis_angle_to_quat(aa)
    q = _unroll_quaternion_signs(q)
    smoothed = savgol_clean(q.reshape((len(q), -1)), window=window,
poly=poly).reshape(q.shape)
    smoothed = quat_normalize(smoothed)
    return quat_to_axis_angle(smoothed).reshape(original_shape).astype(np.float32)


def _unroll_quaternion_signs(q: np.ndarray) -> np.ndarray:
    out = np.asarray(q, dtype=np.float32).copy()
    flat = out.reshape((out.shape[0], -1, 4))
    for idx in range(1, flat.shape[0]):
        dots = np.sum(flat[idx - 1] * flat[idx], axis=-1)
        flat[idx, dots < 0.0] *= -1.0
    return out
```

Файл aimocap/fusion/sync.py реалізує часову синхронізацію каналів: вибір найближчого запису, лінійну інтерполяцію числових полів і сферичну інтерполяцію обертань.

Файл: aimocap/fusion/sync.py:

```
from __future__ import annotations

from typing import Any

import numpy as np

from aimocap.geometry import axis_angle_to_quat, quat_to_axis_angle, slerp


QUAT_FIELDS = {"root_q", "head_q"}
AXIS_ANGLE_FIELDS = {"body_pose", "left", "right", "jaw", "eyes", "hands"}


def nearest_index(times: np.ndarray, t: float, max_gap_s: float | None = None) -> int |
None:
    arr = np.asarray(times, dtype=np.float64)
    if arr.size == 0:
        return None
    idx = int(np.searchsorted(arr, t))
    candidates = []
    if idx < arr.size:
        candidates.append(idx)
    if idx > 0:
        candidates.append(idx - 1)
    best = min(candidates, key=lambda i: abs(float(arr[i]) - float(t)))
    if max_gap_s is not None and abs(float(arr[best]) - float(t)) > max_gap_s:
        return None
    return best


def sample_nearest(records: np.ndarray, t: float, max_gap_s: float | None = None):
    idx = nearest_index(records["t"], t, max_gap_s=max_gap_s)
    if idx is None:
        return None
    return records[idx]


def sample_linear(times: np.ndarray, values: np.ndarray, t: float) -> np.ndarray:
    ts = np.asarray(times, dtype=np.float64)
    vals = np.asarray(values, dtype=np.float64)
    if ts.size == 0:
        raise ValueError("cannot sample empty series")
    if ts.size == 1 or t <= ts[0]:
        return vals[0].copy()
    if t >= ts[-1]:
        return vals[-1].copy()
    right = int(np.searchsorted(ts, t))
    left = right - 1
    alpha = (float(t) - float(ts[left])) / max(float(ts[right] - ts[left]), 1e-12)
    return ((1.0 - alpha) * vals[left] + alpha * vals[right]).astype(values.dtype,
copy=False)
```

```
def sync_sample(records: np.ndarray | None, t: float):
    """A5: sample one structured channel at render time t."""
    if records is None or len(records) == 0:
        return None
    if len(records) == 1:
        out = records[0].copy()
        out["t"] = float(t)
        return out
    ts = np.asarray(records["t"], dtype=np.float64)
    right = int(np.searchsorted(ts, float(t), side="right"))
    if right == 0:
        return _edge_record(records[0], float(t))
    if right >= len(records):
        return _edge_record(records[-1], float(t))
    left = right - 1
    alpha = (float(t) - float(ts[left])) / max(float(ts[right] - ts[left]), 1e-12)
    return _interpolate_record(records[left], records[right], alpha, float(t))


def sync_channels(channels: dict[str, np.ndarray], t: float) -> dict[str, Any]:
    return {name: sample for name, records in channels.items() if (sample :=
sync_sample(records, t)) is not None}


def _edge_record(record: np.void, t: float) -> np.void:
    out = record.copy()
    out["t"] = float(t)
    names = getattr(out.dtype, "names", ()) or ()
    if "conf" in names:
        out["conf"] = float(out["conf"]) * 0.5
    if "region_conf" in names:
        out["region_conf"] = np.asarray(out["region_conf"], dtype=np.float32) * 0.5
    return out


def _interpolate_record(left, right, alpha: float, t: float):
    out = np.zeros((), dtype=left.dtype)
    out["t"] = t
    for name in left.dtype.names or ():
        if name == "t":
            continue
        if name in QUAT_FIELDS:
            out[name] = slerp(left[name], right[name], alpha)
        elif name in AXIS_ANGLE_FIELDS:
            out[name] = _slerp_axis_angle_set(left[name], right[name],
alpha).reshape(out[name].shape)
        elif np.issubdtype(out[name].dtype, np.floating):
            out[name] = (1.0 - alpha) * left[name] + alpha * right[name]
        else:
            out[name] = left[name]
    return out


def _slerp_axis_angle_set(left, right, alpha: float) -> np.ndarray:
    left_aa = np.asarray(left, dtype=np.float64).reshape((-1, 3))
    right_aa = np.asarray(right, dtype=np.float64).reshape((-1, 3))
    q0 = axis_angle_to_quat(left_aa)
    q1 = axis_angle_to_quat(right_aa)
    blended = np.stack([slerp(a, b, alpha) for a, b in zip(q0, q1)], axis=0)
    return quat_to_axis_angle(blended)
```

Файл aimocap/fusion/fuse.py містить регіональне злиття лицевих каналів. Відео- та аудіоджерела поєднуються окремо для ділянок обличчя з урахуванням довіри та політики ваг.

Файл: aimocap/fusion/fuse.py:

```
from __future__ import annotations

from dataclasses import dataclass
from typing import Any

import numpy as np

from aimocap.constants import DEFAULT_FACE_AUDIO_WEIGHT, REGION_INDEX,
REGION_PARTITION, REGIONS_ORDER
from aimocap.format import AimocapFile
from aimocap.fusion.sync import sync_sample


NEUTRAL_52 = np.zeros(52, dtype=np.float32)


@dataclass(slots=True)
class FaceFusionResult:
    t: float
    blendshapes: np.ndarray
    region_conf: np.ndarray
    video: object | None
    audio: list[object]


def policy_from_manifest(manifest: dict[str, Any], override: dict[str, Any] | None =
None) -> dict[str, Any]:
    out = dict(manifest.get("policy", {}))
    if "face_fusion_audio_weight" not in out:
        out["face_fusion_audio_weight"] = dict(DEFAULT_FACE_AUDIO_WEIGHT)
    else:
        merged = dict(DEFAULT_FACE_AUDIO_WEIGHT)
        merged.update(out["face_fusion_audio_weight"])
        out["face_fusion_audio_weight"] = merged
    out.setdefault("conf_floor", 0.30)
    out.setdefault("emotion_weight", 0.5)
    if override:
        for key, value in override.items():
            if key == "face_fusion_audio_weight" and isinstance(value, dict):
                merged = dict(out["face_fusion_audio_weight"])
                merged.update(value)
                out[key] = merged
            else:
                out[key] = value
    return out


def fuse_face_layers(
    video_face: object | None,
    audio_faces: list[object] | tuple[object, ...],
    policy: dict[str, Any] | None = None,
    *,
    t: float = 0.0,
```

```
) -> FaceFusionResult:
    """A6: per-region consumer fusion over video + any number of audio face
channels."""
    policy = policy_from_manifest({"policy": policy or {}})
    floor = float(policy.get("conf_floor", 0.30))
    weights = policy.get("face_fusion_audio_weight", DEFAULT_FACE_AUDIO_WEIGHT)
    base = NEUTRAL_52.copy()
    region_conf = np.zeros(len(REGIONS_ORDER), dtype=np.float32)
    if video_face is not None:
        base = np.asarray(video_face["blendshapes"], dtype=np.float32).copy()
        region_conf = np.asarray(video_face["region_conf"], dtype=np.float32).copy()
    for region in REGIONS_ORDER:
        ridx = REGION_INDEX[region]
        indices = np.asarray(REGION_PARTITION[region], dtype=np.int64)
        fc = max(0.0, float(region_conf[ridx]) - floor)
        for audio in audio_faces:
            audio_conf = np.asarray(audio["region_conf"], dtype=np.float32)
            ac = max(0.0, float(audio_conf[ridx]) - floor) *
float(np.clip(weights.get(region, 0.0), 0.0, 1.0))
            denom = fc + ac
            if denom <= 1e-8:
                continue
            alpha = ac / denom
            audio_blend = np.asarray(audio["blendshapes"], dtype=np.float32)
            base[indices] = (1.0 - alpha) * base[indices] + alpha *
audio_blend[indices]
            region_conf[ridx] = max(region_conf[ridx], float(audio_conf[ridx]))
            fc = max(0.0, float(region_conf[ridx]) - floor)
    return FaceFusionResult(
        t=float(t),
        blendshapes=np.clip(base, 0.0, 1.0).astype(np.float32),
        region_conf=np.clip(region_conf, 0.0, 1.0).astype(np.float32),
        video=video_face,
        audio=list(audio_faces),
    )


def sample_fused_face(amc: AimocapFile, t: float, policy_override: dict[str, Any] |
None = None) -> FaceFusionResult:
    policy = policy_from_manifest(amc.manifest, policy_override)
    video = None
    audio: list[object] = []
    for item in amc.manifest.get("channels", []):
        if item.get("role") != "face":
            continue
        records = amc.channels.get(str(item.get("name")))
        if records is None:
            continue
        sampled = sync_sample(records, t)
        if sampled is None:
            continue
        if item.get("modality") == "video" and video is None:
            video = sampled
        elif item.get("modality") == "audio":
            audio.append(sampled)
    return fuse_face_layers(video, audio, policy, t=t)


# Compatibility name for old imports; now delegates to region fusion with mouth-only
coverage.
def fuse_face_and_mouth(
    face_blendshapes: np.ndarray,
```

```
    face_conf: float,
    mouth: np.ndarray | None,
    mouth_conf: float,
    audio_weight: float = 0.6,
    conf_floor: float = 0.3,
    per_dim: bool = False,
) -> np.ndarray:
    del per_dim
    video = np.zeros((), dtype=[("t", "<f8"), ("blendshapes", "<f4", (52,)),
("region_conf", "<f4", (7,))])
    video["blendshapes"] = np.asarray(face_blendshapes, dtype=np.float32)
    video["region_conf"] = float(face_conf)
    audio = np.zeros((), dtype=video.dtype)
    if mouth is None:
        return np.asarray(face_blendshapes, dtype=np.float32).copy()
    audio["region_conf"] = 0.0
    audio["region_conf"][REGION_INDEX["jaw"]] = float(mouth_conf)
    audio["region_conf"][REGION_INDEX["mouth"]] = float(mouth_conf)
    from aimocap.constants import MOUTH_15_TO_ARKIT

    audio["blendshapes"][MOUTH_15_TO_ARKIT] = np.asarray(mouth, dtype=np.float32)
    policy = {"conf_floor": conf_floor, "face_fusion_audio_weight": {"jaw":
audio_weight, "mouth": audio_weight}}
    return fuse_face_layers(video, [audio], policy).blendshapes
```