МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут електронних та інформаційних технологій
Кафедра інформаційних технологій і програмної інженерії

Допущено до захисту
В.о. кафедрою ІТіПІ
к.т.н. І.В.Білоус
« ___ » _____ 2020 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

РОЗРОБКА ВІДЕОПЛЕЄРУ З МОЖЛИВІСТЮ ІНТЕРПОЛЯЦІЇ ЧАСТОТИ КАДРІВ ВІДЕО

по спеціальності 121 – «Інженерія програмного забезпечення»

Виконавець:
здобувач ВО гр. ПІ-161
Заворотний Андрій Олександрович _____
(прізвище, ім'я, по батькові,) (підпис)

Керівник:
викладач _____
(посада) (науковий ступінь, вчене звання)
Нехай Валентин Валентинович _____
(прізвище, ім'я, по батькові,) (підпис)

Чернігів 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут електронних та інформаційних технологій
Кафедра інформаційних технологій і програмної інженерії

ЗАТВЕРДЖУЮ:
В.о. кафедрою ІТіПІ
к.т.н. І.В.Білоус
« ___ » _____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА (ПРОЕКТ)
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Заворотного Андрія Олександровича

Тема роботи: розробка відеоплеєру з можливістю інтерполяції частоти кадрів відео

Тему затверджено наказом ректора
від "16" березня 2020р.
№166-С

1. *Вхідні дані до роботи*: інтерфейс взаємодії з відеоплеєром, модель згорткової штучної нейронної мережі.

2. *Зміст розрахунково-пояснювальної записки:* розробка плагіну відеоплеєру mpv для інтерполяції частоти кадрів за допомогою моделі згорткової штучної нейронної мережі DAIN.

3. *Перелік графічного матеріалу:* графічний інтерфейс користувача, IDEF0, DFD, IDEF3, UML.

4. Календарний план

№	*Назва етапів роботи*	*Термін виконання*	*Примітки*
1	Узгодження теми дипломного проекту з керівником	24.09.2019	
2	Аналіз предметної області та методів розв'язання проблеми	25.09.2019	
3	Аналіз існуючих рішень	01.10.2019	
4	Вибір технологій та інструментальних рішень розробки	06.03.2020	
5	Написання вступу та першого розділу документації	04.05.2020	
6	Моделювання та проектування плагіну	06.05.2020	
7	Написання другого розділу документації	10.05.2020	
8	Розробка та реалізація плагіну	15.05.2020	
9	Завершення оформлення документації. Написання третього розділу та висновків	05.06.2020	
10	Подання ДР рецензенту. Отримання рецензії	18.06.2020	
11	Надання пакету документів по ДР до захисту в ЕК	19.06.2020	
12	Захист ДР в ЕК	22.06.2020	

Завдання підготував:

керівник_____ Нехай Валентин Валентинович

(підпис) (*прізвище, ім'я, по батькові*)

«__»_____202_р.

Завдання одержав:

здобувач вищої освіти _____ Заворотний Андрій Олександрович

(підпис) *(прізвище, ім'я, по батькові)*

«__» _____202_р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

здобувача вищої освіти гр. ПІ–161 Заворотного А.О.

Тема роботи: Розробка відеоплеєру з можливістю

інтерполяції частоти кадрів відео

Запропоновані технічні та експлуатаційні результати роботи

Необхідно розробити відеоплеєр, що застосовує метод інтерполяції частоти кадрів за допомогою сучасних методів обробки зображення та глибокого навчання у вигляді плагіну, виконати системний аналіз інформаційних процесів для плагіну, що розробляється. Плагін повинен мати змогу обробляти довільні відео-файли.

Плагін розробляється для перегляду відео-файлів з підвищеною кількістю кадрів за секунду.

У ході виконання кваліфікаційної роботи потрібно розробити і описати:

– архітектуру плагіну для відеплеєру;

– структуру модулю інтерполяції;

– інтерфейс користувача.

Функціональне призначення:

1) інтерполяція кадрів за допомогою моделі згорткової штучної нейронної мережі;

2) перегляд відео-файлу без затримок на обробку завдяки пререндерингу;

3) синхронність відео та аудіо потоків.

Вимоги по дизайну плагіну:

1) інтуїтивний інтерфейс користувача;

2) мінімалістичне управління.

Для реалізації буде використана мова програмування Python, платформонезалежний відеоплеєр mpv з відкритим вихідним кодом, фреймворк глибокого навчання PyTorch.

Обсяг текстової та графічної інформації

Робота обсягом 60-80 стор. формату А4 і 10–12 електронних слайдів.

Орієнтовна трудомісткість робіт – 260 людино–годин.

Планові терміни по етапах

Попередній захист з ілюстративного матеріалу в повному обсязі – 08.06.2020.

Плановий термін захисту роботи

Робота планується до захисту на засіданні ЕК 19-Б з 22.06.2020.

Керівник роботи

Керівник роботи і консультант по розділах	В.В. Нехай
Керівник роботи	В.В. Нехай
Дата видачі завдання	24.09.2019 р

РЕФЕРАТ

Кваліфікаційної роботи бакалавра Заворотного Андрія Олександровича на тему: "Розробка відеоплеєру з можливістю інтерполяції частоти кадрів відео" містить 67 с., 27 рис., 1 табл., 56 джерел.

Актуальність роботи. Зір людини відмічає різницю між відео зі стандартною частотою кадрів та частотою більш ніж у 4 рази більшою за стандартну, коли знято об’єкт, що швидко рухається. Стандарти частоти кадрів за секунду відображають технічні обмеження відеозапису майже столітньої давнини, що вже не є актуальними.

Метою роботи є плагін відеоплеєру mpv для інтерполяції частоти кадрів для довільного відео-файлу за допомогою моделі згорткової штучної нейронної мережі DAIN.

Завдання дослідження:

– провести аналіз предметної області та рішень із схожим функціоналом;

– описати структуру плагіну;

– сформулювати вимоги до розроблюваного плагіну;

– змоделювати та спроектувати плагін;

– розробити інтерфейс;

– розробити архітектуру плагіну;

– реалізувати програмний проект, який би задовольняв поставлені вимоги.

Об’єктом дослідження є підвищення частоти кадрів для довільного відео у відеоплеєрі.

Предметом дослідження є аспекти реалізації плагіну інтерполяції частоти кадрів та інтеграції моделі згорткової штучної нейронної мережі у плагін.

У роботі було використано ***методи дослідження:*** аналітичний, програмний, порівняльний метод аналізу та синтезу.

Наукова новизна одержаних результатів дослідження полягає у розробленні платформонезалежного і простого у використані програмного продукту, що вирішує завдання інтерполяції частоти кадрів відео, використовуючи модель згорткової штучної нейронної мережі.

Практичне значення отриманих результатів. Очікується отримати плагін, що реалізує можливість підвищення частоти кадрів для довільного відео методами глибокого навчання.

Ключові слова: PYTHON, PYTORCH, CUDA, MPV, DAIN, DEEP LEARNING.

ABSTRACT

Bachelor's qualification work of Zavorotnyi Andrii, entitled "Development of a video player with the ability to interpolate the frame rate", 67p., 27 images, 1 tables, 56 sources.

Relevance. A person notices a difference between a video with a standard frame rate and a video with frame rate more than 4 times greater than a standard, when shooting a fast-moving subject. The frame rate standards reflect the technical limitations of a long-ago video capturing that is no longer relevant.

The aim of the work is to develop a plugin for the mpv video player to interpolate the frame rate for a custom video file using a convolutional artificial neural network DAIN model**.**

Objectives of the study:

– analyze the subject area and solutions with similar functionality;

– describe the structure of the plugin;

– formulate requirements for the developed plugin;

– model and design a plug-in;

– develop an interface;

– develop a plugin architecture;

– implement a software project that would meet the requirements.

The object of study is to increase the frame rate for any video in the video player.

The subject of research is the implementation aspects of the frame rate interpolation plugin and integration of the convolutional artificial neural network model into the plugin.

The research methods were used in the work: analytical, programing, comparative method of analysis and synthesis.

The scientific novelty of the obtained research results is the development of a cross platform and easy-to-use software that solves the problem of video

frame frequency interpolation using a convolutional artificial neural network model.

The practical significance of the results. It is expected to get a plugin that realizes the ability to increase the frame rate for custom video with deep learning methods.

Keywords: PYTHON, PYTORCH, CUDA, MPV, DAIN, DEEP LEARNING.

ЗМІСТ

ВСТУП

Зір людини відмічає різницю між відео зі стандартною частотою кадрів та частотою більш ніж у 4 рази більшою за стандартну, коли знято об’єкт, що швидко рухається.

Стандарти частоти кадрів за секунду відображають технічні обмеження відеозапису майже столітньої давнини, що вже не є актуальними. Сьогодні навіть найдешевші камери можуть знімати відео з вищою частотою кадрів

Але вже відзняте відео перезняти з високою частотою кадрів у більшості випадків неможливо. Тому існує проблема інтерполяції частоти кадрів.

Сучасні рішення цієї проблеми, що представлені готовими до використання користувачем програмними рішеннями, не є ідеальними. Ці рішення використовують суто алгебраїчні методи інтерполяції, що ведуть до появи значних спотворень.

У той же час сучасні рішення з використанням моделей штучних нейронних мереж показують значно кращий результат, але не представлені програмними рішеннями, що готові до використання пересічним користувачем.

В процесі виконання роботи буде реалізавано плагін для відеоплеєру, що використовує модель згорткової штучної нейронної мережі для інтерполяції частоти кадрів для довільного відео-файлу.

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

API (application programming interface) – набір готових класів, процедур, функцій, структур та констант, наданих прикладенням (бібліотекою, сервісом) або операційною системою для використання в зовнішніх програмних продуктах.

UI (user interface) – інтерфейс користувача, що забезпечує передачу інформації між користувачем людиною і програмно-апаратними компонентами комп'ютерної системи.

FPS (frames per second) – частота кадрів, або кадрова частота – це частота, з якою пристрій формування зображення відображає послідовні зображення, що називаються кадрами. Частота кадрів зазвичай задається в кадрах за секунду (англ. FPS).

ШНМ (штучні нейронні мережі) – це обчислювальні системи, натхнені біологічними нейронними мережами, що складають мозок тварин. Такі системи навчаються задач, розглядаючи приклади, загалом без спеціального програмування під задачу.

PID (process identifier) – ідентифікатор процесу багатозадачної операційної системи, номер його в порядку запуску, від нуля до максимального значення, можливого в даній системі.

GUI (graphical user interface) — тип інтерфейсу, який дає змогу користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ

1.1 Аналіз предметної області та методів розв’язання проблеми

1.1.1 Стандарт частоти кадрів

Перші фільми, зняті в кінці XIX століття братами Люм'єр, були зняті з частотою 16 кадрів в секунду. Цю цифру вибрали тому, що витрата стандартної плівки 35мм при такій частоті становила рівно 1 фут в секунду. Таким чином спрощувались розрахунки необхідної кількості плівки для запису. Цього було достатньо, щоб домогтися ілюзії руху, а студії не витрачали непомірні гроші з кожним обертанням ручки камери. [2]

З початку 1900 років до 1920-х в індустрії не було стандарту частоти кадрів, правил не існувало. Студії були зацікавлені, щоб частота кадрів залишалася якомога нижчою, бо чим більше частота, тим більше використовується плівка, отже, потрібно більше витрачати коштів. [1]

Потреба в збільшенні частоти кадрів виникла з переходом від німого кіно до звукового. Доріжка в ті часи писалася на плівку поруч з картинкою у вигляді смужок, кожна з яких відповідала певній частоті. Мала довжина плівки, що прокручується за секунду (всього 30 см), не дозволяла записати звук досить чітко, тому довжину потрібно було збільшувати. [2]

Протягом 1920-х років винахідники і режисери продовжували розвивати способи запису. Де Форест поліпшив якість Phonofilm, а Western Electronics і Warner Brothers розширювали технологічні кордони з форматом аудіодисків Vitaphone. Саме Vitaphone і привів до стандартизації частоти кадрів. Vitaphone використовував дисковий і плівковий запис, за допомогою синхронних моторів. Це робило розсинхронізацію відео потоків та аудіо потоків набагато менш вірогідною. Інженери Vitaphone використовували шістнадцяти дюймовий формат дисків зі швидкістю відтворення 33.(3) обертів на хвилину, це давало 11 хвилин запису, що приблизно дорівнювало

тому, скільки 300-метрова котушка плівки зможе відтворити при зчитуванні 27 метрів в хвилину: 24 кадри в секунду. [2]

Стандартизація швидкості запису була першим відчутним результатом появи звуку. У той час як відображення руху має досить широкі межі без якогось помітного дискомфорту для непідготовленого ока, але для аудіо різна швидкість відтворення змінює висоту звуку. У вересні 1927 року Комітет номенклатури і стандартів SMPTE провів дослідження, щоб знайти швидкості, які використовують сучасні аудіосистеми. Дві з них, які найчастіше використовувалися в індустрії (Vitaphone і Movietone), використовували 24 кадри за секунду. Система RCA, що знаходиться в розробці на той момент, використовувала 22 кадри, а Forest Phonofilms (практично припинила діяльність на той момент) - 20 кадрів. Беручи до уваги дані факти, а також тренди в показі, задані за останні 10 років, і рішення винахідників найуспішніших аудіосистем, Комітет зробив 24 кадри за секунду стандартом. [6]

Таким чином, 24 кадри за секунду є частотою кадрів за замовчуванням тільки тому, що Vitaphone перший досяг фінішу в 1927 році з випуском «Співака джазу». Якби цього не сталося, ми могли б дивитися фільми з 22 кадрами за секунду. «Співак джазу» був не першим фільмом зі звуком, але першою картиною з синхронізованими діалогами. [3]

За новим стандартом секундна витрата плівки становила 1,5 фути, хвилинна - 90 футів або 30 ярдів. Ці цифри теж виявилися зручними для розрахунків при плануванні бюджету запису. Також 24 кадри за секунду легко ділити, а монтажери могли знайти точний час для склейок, ґрунтуючись на кількості кадрів. 12 кадрів становили пів секунди, 6 кадрів - чверть, і так далі. Частоту намагалися збільшити і більше, до 30, 48 і навіть 60 кадрів за секунду, але виникали проблеми. [2]

Остаточно затвердили частоту близько 25 кадрів в секунду через електрифікацію Європи і появу телебачення. [8] При частоті змінного струму 50 Гц (змін напрямку в секунду) 24-25 кадрів зручно прив'язувати до

параметрів струму. При такому підході зміна кадру відбувається один раз на період синусоїди, представлена на рисунку 1.1. А в США, де замість звичних 220-230 вольт 50 Гц використовується 110-120 вольт 60 Гц, телевізійний стандарт NTSC працює з частотою 30 (29,97) кадрів в секунду. [4]

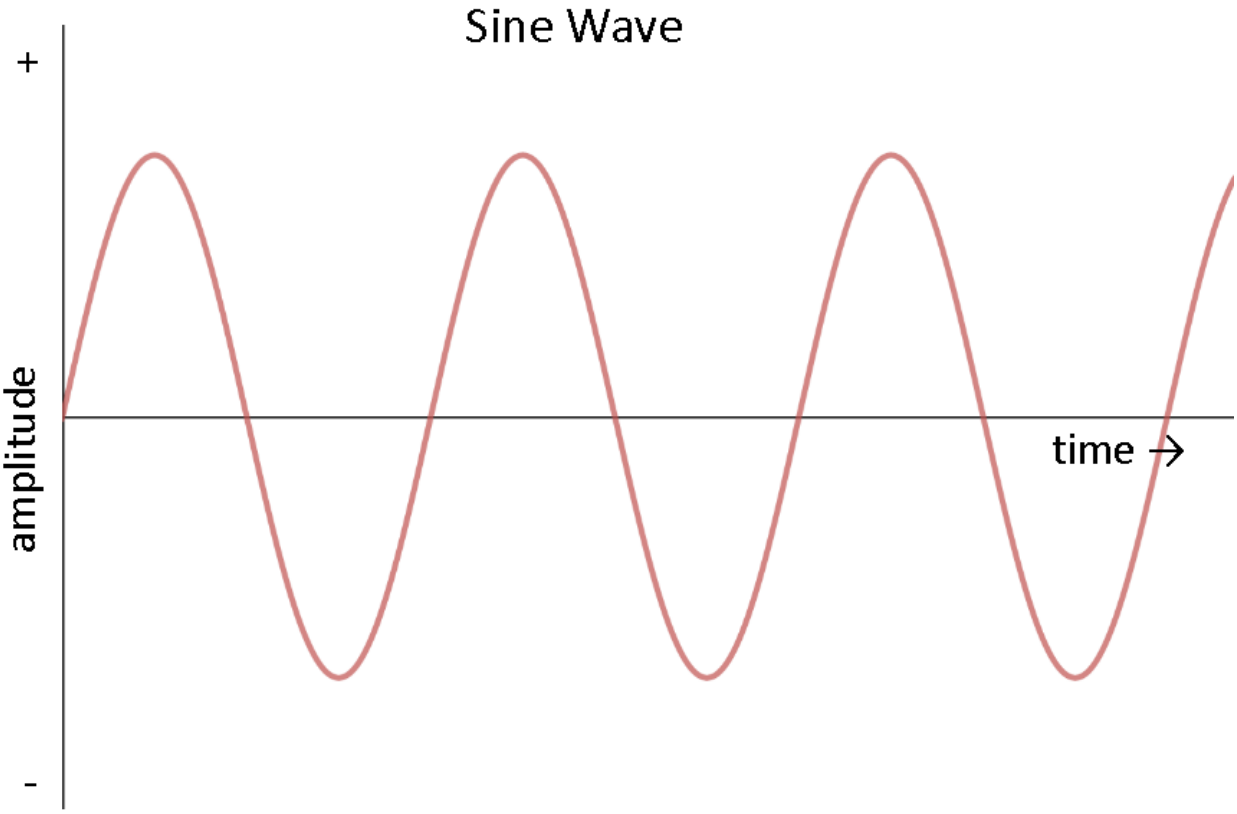

Рисунок 1.1 — Синусоїда змінної частоти струму.

1.1.2 Зір та сприйняття частоти кадрів

Людина не зчитує реальність як комп'ютер, а візуальне сприйняття цілком будується на спільній роботі очей і мозку. Тому, наприклад, люди по-різному бачать рух і світло, а периферійний зір краще справляється з деякими аспектами картинки, ніж основний - і навпаки. [5] Це пов'язано з відмінностями між основним і периферійним зором, які дісталися людям від далеких предків. Коли людина дивиться прямо на об'єкт, то розрізняє найдрібніші деталі, проте зір погано справляється з швидко рухомими предметами. Периферійний зір, навпаки, страждає нестачею деталей, але гарно сприймає рухи. [4]

Зір людини може сприймати від 10 до 12 змінюваних кадрів як окремі зображення. Відчуття зображення у русі виникає тоді, коли за секунду людина бачить більше кадрів. Вперше це було задокументовано психологом Максом Вертгеймером. [4]

Суперечки про те, скільки людське око може максимально сприймати кадрів в секунду ведуться давно, тому, що на це питання немає однозначної відповіді. Час, за який людина сприймає візуальну інформацію, сумується з часом, за який світло потрапляє в очі, часом передачі отриманої інформації в мозок та часом її обробки. [5]

Зорова система сприймає картинку цілісно, помічаючи тільки її зміни. Тому ніякої конкретної цифри, що вказує на межі можливостей ока, немає. Межі сприйняття сильно залежать від особливостей об'єкта, що спостерігається. Чим швидше він рухається, та чим різкіше ці рухи - тим вище гранична частота [3], приклад представлений на рисунку 1.2. Варто зазначити, що відео з великою частотою кадрів здаються більш реальними. [9]

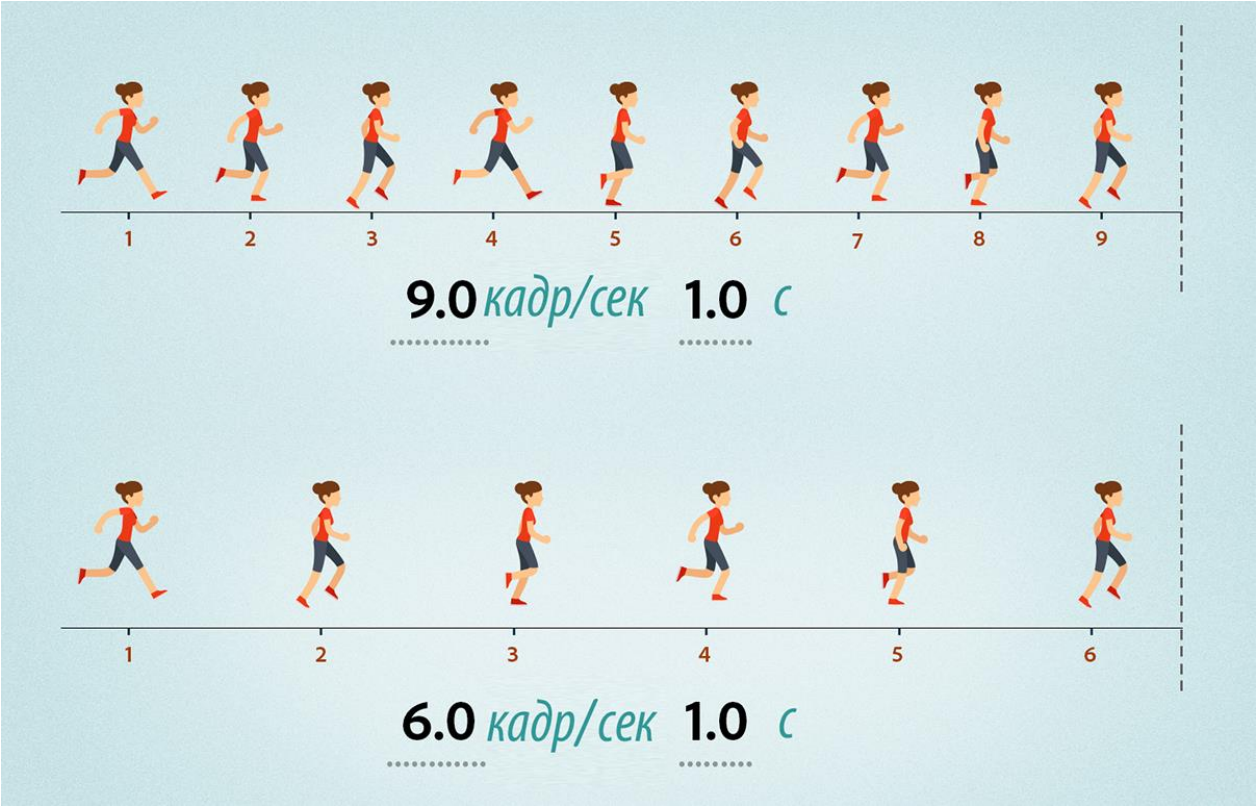

Рисунок 1.2 — Приклад кадрів людини, що біжить.

Також сприйняття руху багато в чому залежить і від того, в якому положенні людина знаходиться, сидить на місці і стежить за об'єктом, чи рухається. Нерухомо дивлячись відео, на якому інша людина повільно йде по прямій, око не помітить істотної різниці між 24 і 60 кадрів за секунду, так як рухи плавні. Якщо ця людина швидко біжить - різниця вже буде гарно помітна, відео з частотою 60 кадри за секунду буде набагато плавніше і приємніше, ніж з частотою 24 кадрів за секунду. А якщо ця людина не просто біжить, а біжить зигзаґом, попутно стрибаючи через перешкоди - то навіть різниця між 60 і 120 кадрами за секунду буде помітна, на користь більшої частоти. [4] Межа, після якої різниця вже непомітна залежить також від індивідуальних особливостей зору, і приблизно становить 80-150 кадрів в секунду. [7]

1.1.3 Проблема інтерполяції частоти кадрів

Оскільки перегляд відео більше не обмежується частотою показу кадрів пристроями виводу, то підвищення їх частоти з точки зору технічної можливості є питанням програмних засобів інтерполяції частоти кадрів.

Взагалі рух інтерполяції або кадр інтерполяції з компенсацією руху (MCFI) є однією з форм обробки відео, в якому проміжні кадри анімації генеруються між існуючими за допомогою інтерполяції, в спробі зробити анімацію більш плавною і компенсувати розмиття руху. [8] Головне завдання — це розрахунок проміжних кадрів і додавання їх до відео-потоку.

Для більшості методів інтерполяції описані візуальні аномалії, що виглядають як збій у зображенні, що з'являється на долю секунди. Ефект найбільш очевидний під час непередбачуваних або неправильно інтерпретованих алгоритмом рухах. Це явище називається цифровим артефактом. Оскільки технології інтерполяції з часом удосконалювалися, такі артефакти з'являються рідше. [10] Варто зазначити, що артефакти

трапляються частіше у відео-потоках з низької частотою кадрів. Приклад візуального артефакту представлений на рисунку 1.3.

Рисунок 1.3 — Приклад візуального артефакту.

Також має місце так званий "ефект мильної опери" (soap opera effect, SOE), посилаючись на характерний вигляд більшості телевізійних мильних опер або багатокамерних ситкомів до 2000-х років, які, як правило, знімали, використовуючи дешеві “60i” камери. Деякі люди скаржаться на те, що ефект мильної опери руйнує театральний вигляд кінематографічних творів, додаючи відчуття, ніби глядач або на знімальному майданчику, або дивиться спектакль позаду, за кадром. [11]

Через особливості задачі, такі як непередбачуваність об’єктів на відео та висока варіативність якості відео-файлів, загальноприйнятого алгоритму чи методу інтерполяції частоти кадрів немає. Суто алгебраїчні методи це у великій мірі пошук середнього між двома зображеннями. Артефакти неминучі, тому виникає потреба у методах усунення артефактів. [10]

З огляду на ефективність згорткових штучних нейронних мереж у сфері комп'ютерного зору та генеративно-змагальних штучних нейронних мереж у задачах створення нових зображень [12], можна говорити про новий основний напрямок розробки методів інтерполяції частоти кадрів.

1.2 Аналіз існуючих рішень

1.2.1 HDTV

Існує великий клас рішень для HDTV [7], що були розроблені виробниками телевізорів, до них належать (у вигляді «вирбоник – технологія»:

- Hitachi — Reel120;
- Insignia — DCM Plus, Insignia Motion;
- LG — TruMotion;
- AOC — Motion;
- Bose — VideoWave III;
- Mitsubishi — Smooth;
- Panasonic — Smooth Film;
- Philips — HD Digital Natural Motion, Perfect Motion Rate;
- Samsung — Auto Motion Plus, Clear Motion;
- Sharp — Fine Motion Enhanced, AquoMotion, AquoMotion Pro;
- Sony — MotionFlow;
- Toshiba — ClearScan;
- Vizio — SmoothMotion, Clear Action;
- Скипетр — МЕМС;
- Hisense — Motion Rate SMR 120;
- Westinghouse — MEMC;
- Kogan.com — MotionMax;
- Loewe — Digital Movie (DMM);

та інші.

Приклад застосування представлено на рисунку 1.4.

Рисунок 1.4 — Приклад застосування рішень HDTV.

Усі перелічені технології пов’язують спільні риси, а саме: рішення працюють лише на певних моделях виробника, архітектура та алгоритм роботи не афішуються, рішення не є суто програмними. Також більшість із них створені для того, щоб телевізори краще продавалися, характеристики можуть буди завищенні для рекламних цілей. З огляду на неможливіть аналізу і неможливість дізнатися алгоритм роботи, цей клас рішень буде внесено до порівняльної характеристики у вигляді однієї сутності.

1.2.2 Програмні рішення

Більш цікавим для аналізу є клас суто програмних рішень [7], а саме:

– WinDVD, що використовує Philips' TrimensionDNM;

– PowerDVD, що використовує TrueTheater Motion;

– Splash PRO, що використовує Mirillis Motion;

– DmitriRender, що використовує «GPU-oriented frame rate conversion algorithm with native DXVA support»;

– Bluesky Frame Rate Converter (BFRC), що використовує AMD Fluid Motion;

– SmoothVideo Project (SVP).

Найбільш популярним рішенням серед перелічений є SmoothVideo Project (SVP). Вдалий та невдалий приклад обчисленого проміжного кадру, застосовуючи Smooth Video Project, представлено на рисунку 1.5 та 1.6 відповідно.

Рисунок 1.5 — Вдалий приклад застосування рішень SVP.

Рисунок 1.6 — Невдалий приклад застосування рішень SVP.

Перелічені рішення піддаються аналізу і можуть бути використані для порівняння, що представлене в таблиці 1.1

Таблиця 1.1 - Порівняння функціоналу чат-ботів

	HDTV	WinDVD	PowerDVD	Splash PRO	DmitriRender	BFRC	SVP	Плагін, що буде розроблено
Перегляд на ПК		✔	✔	✔	✔	✔	✔	✔
Платформонезалежний							✔	✔
Не комерційний						✔		✔
Використана модель глибокого навчання					✔			✔
Обробка у реальному часі	✔		✔	✔	✔		✔	✔
Фільтрація артефактів	✔		✔	✔			✔	
Збереження результату		✔				✔	✔	✔

1.3 Висновки до розділу

Була описана проблема частоти кадрів, існуючі стандарти, біологічні обмеження сприйняття частоти кадрів.

Також були виокремлені основні проблеми інтерполяції частоти кадрів та сучасні методи їх вирішення.

Були проаналізовані та порівняні існуючі рішення.

Необхідно реалізувати можливість переглядати відео-файли з інтерполяцією частоти кадрів у плагіні для платформонезалежного відеоплеєру за допомогою сучасних методів глибокого навчання.

РОЗДІЛ 2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Аналіз технологій та інструментальних засобів розробки

2.1.1 Мова програмування

Обрана мова програмування повинна мати можливості та ефективну екосистему для роботи з відео та аудіо, для створення моделей згорткових штучних нейронних мереж та їх навчання. [14]

Такою мовою програмування є Python версії 3. Логотип Python представлено на рисунку 2.1.

Рисунок 2.1 – Логотип Python.

Python (найчастіше вживане прочитання — «Па́йтон», запозичено назву з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. [13]

Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні

бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована. Python має динамічну типізацію та збирач сміття та також має всебічну бібліотеку стандартів. [15]

2.1.2 Відеоплеєр

Немає необхідності створювати власний відеоплеєр, якщо існують вже створені повнофункціональні платформонезалежні відеоплеєри з відкритим вихідним кодом, що вільно розповсюджуються. Важливим фактором вибору є API на мові програмування Python версії 3.

Усім переліченим критеріям відповідає mpv відеоплеєр. [17] Логотип mpv представлено на рисунку 2.2.

Рисунок 2.2 – Логотип mpv.

Mpv (українською: емпіві) – це програвач, який був створений на базі MPlayer та mplayer2. Відеоплеєр містить вільне та відкрите програмне забезпечення яке включає в себе суміш GNU General Public License версії 2 плюс (GPLv2+), з елементами GNU Lesser General Public License версії 2.1 плюс (LGPLv2.1+) та деякі додаткові частини GNU General Public License версії 3 (GPLv3). [16]

Плеєр працює на кількох операційних системах, включаючи Unix-like версії Berkeley Software Distribution (BSD), Linux, та OS X, а також на Windows. Це багатоплатформна програма, яка працює на ARM, PowerPC, x86 / IA-32, x86-64 та MIPS. [16]

Окрім режиму медіа програвача, mpv зроблений для використання іншими програмами напряму, через бібліотеку інтерфейсу, яка має назву libmpv, використання якої є безпечним у багатопотоковому режимі. [18]

Відеоплеєр включає в себе режим стиснення даних, який можна використовувати для зберігання відтворюваних на даний момент файлів в різні формати. Це дозволяє mpv працювати як транскодер, підтримуючи багато різних відео форматів. [17]

Поведінка та функції mpv є користувацько-налаштовними. Використовуючи малі програми, які написані на скриптовій мові Lua, яка може бути використана для таких задач як кадрування відео, надання графічного інтерфейсу користувача (GUI) або автоматичого підлаштовування до швидкости оновлення дисплею. [18]

2.1.3 Бібліотека для створення інтерфейсу

Для платформонезалежного відеоплеєру слід обрати інструмент відображення інтерфейсу плагіну, що також буде платформонезалежним. Таким інструментом є PyQT. Логотип PyQT представлено на рисунку 2.3.

Існує декілька платформонезалежних інструментів відображення інтерфейсу та декілька реалізацій Qt для використання у Python, але саме PyQT підтримується бібліотекою пакетом python-mpv для відображення відео у довільному вікні, тому було обрано PyQT. [40]

Рисунок 2.3 – Логотип PyQT.

PyQT — оболонка на мові програмування Python для бібліотеки Qt. Бібліотека реалізована в Python-модулях, та охоплює біля 1,000 класів. [19] PyQt розробляється англійською компанією Riverbank Computing. Підтримуються операційні системи Microsoft Windows, Linux, OS X, iOS та Android.

PyQt5 є одним з найбільш часто використовуваних модулів для створення GUI додатків в Python, і це пов'язано з його простотою. [20]

Ще одна чудова особливість, яка надихає розробників користуватися PyQt5 - це PyQt5 Designer, завдяки якому можна створювати складні GUI додатки досить швидко. [20]

PyQt є вільним програмним забезпеченням і розповсюджується на умовах ліцензії GNU GPL v3 для некомерційного використання. Для комерційного використання існує окрема пропрієтарна ліцензія. Поточна версія PyQt5, відповідно підтримує Qt5. Попередня версія Qt4 більше офіційно не підтримується. [19]

2.1.4 Фреймворк машинного навчання

Існує доволі багато фреймворків машинного навчання для мови програмування Python версії 3, більше ніж для будь-якої іншої мови. Але необхідно, щоб обраний фреймворк підтримував моделі глибокого навчання, був гнучкий і потужний у використанні.

Таким фрейморком машинного навчання штучних нейронний мереж є PyTorch. [22] Логотип mpv представлено на рисунку 2.4.

Рисунок 2.4 – Логотип PyTorch.

PyTorch — відкрита бібліотека машинного навчання на основі бібліотеки Torch, що використовують для таких застосувань, як комп'ютерне бачення та обробка природної мови. Розробляє її переважно група дослідження штучного інтелекту компанії Facebook. Вона є вільним та відкритим програмним забезпеченням, що випускають під ліцензією Modified BSD. І хоча інтерфейс Python є більш відшліфованим, і головним зосередженням розробки, PyTorch також має зовнішній інтерфейс і для C++. [21] PyTorch забезпечує дві високорівневі функціональності:

– Тензорні обчислення (як NumPy) із сильним прискоренням через графічні процесори (ГП);
– Глибинні нейронні мережі, побудовані на системі автоматичного диференціювання.

Тензори можливо розглядати як багатовимірні масиви. Тензори в PyTorch є подібними до масивів NumPy, але ними також можливо оперувати на відеокартах Nvidia з підтримкою CUDA. Ще PyTorch підтримує різні типи тензорів.[21]

Також PyTorch використовує метод автоматичного диференціювання. Записувач зберігає дії, що були виконані, а потім програє їх назад, щоб обчислити градієнти. Цей метод є особливо потужним при будуванні нейронних мереж, щоб заощаджувати час на одній ітерації, обчислюючи диференціювання параметрів на прямому проході. [23]

2.1.5 Модель штучної нейронної мережі

Модель «**D**epth-**A**ware video frame **In**terpolation» **N**etwork (DAIN), яка дозволяє чітко виявити оклюзію, досліджуючи взаємне розташування об’єктів по осі Z. [25]

Інтерполяція відеокадрів спрямована на синтез неіснуючих кадрів між вхідними кадрами. Незважаючи на те, що глибокі нейронні мережі досягли значного прогресу, якість інтерполяції часто знижується через рух великих об'єктів або оклюзію. Тому була обрана саме модель DAIN, зокрема вона створює шар проекції потоку для синтезу проміжних потоків, які переважно відбирають ближчі об'єкти, ніж більш далекі. [24]

Модель ШНМ DAIN створювалася для рішення завдання інтерполяції частоти кадрів, але має певні обмеження.

Модель DAIN використовує згорткові штучні нейронні мережі для обробки кадрів, тому потребує можливості тензорних обчислень потужної відеокарти, яка підтримує технологію CUDA. [24]

Модель DAIN є компактною, ефективною та повністю диференційованою для оптимізації всіх компонентів. Згідно джерелу, кількісні та якісні результати свідчать про те, що запропонована модель вигідно відрізняється від інших сучасних методів інтерполяції частоти кадрів для широкого спектру наборів даних. [25]

Архітектура моделі DAIN представлена на рисунку 2.5.

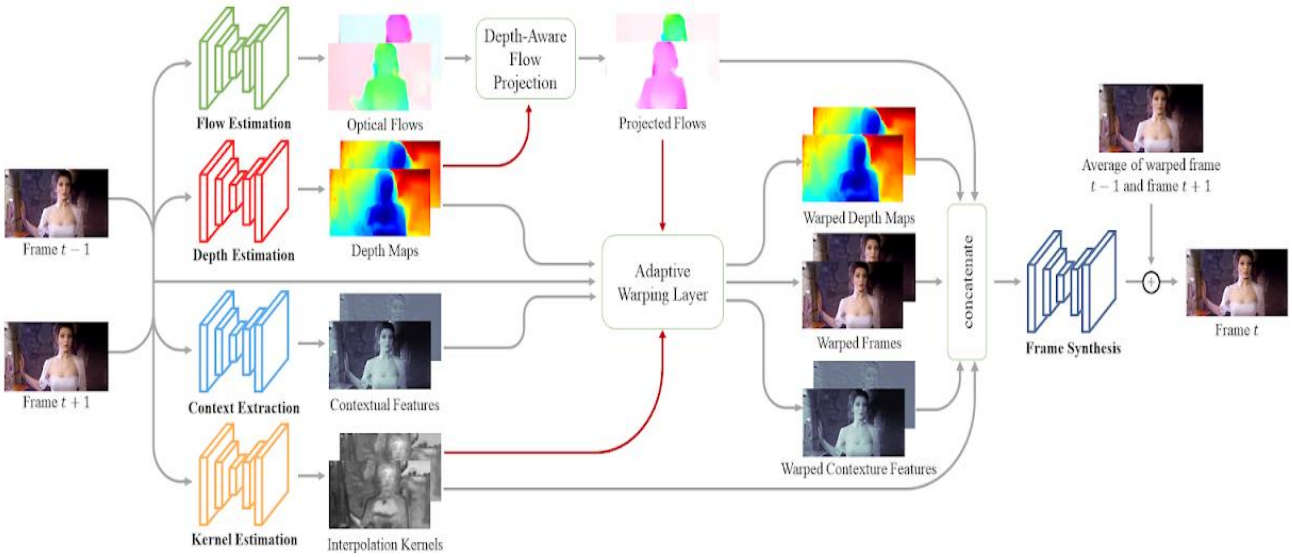

Рисунок 2.5 – Архітектура моделі DAIN.

2.2 Моделювання

Моделювання процесу створення плагіну відеоплеєру для інтерполяції частоти кадрів було виконано за допомогою діаграм системного аналізу інформаційних процесів.

2.2.1 Контекстна діаграма та діаграма декомпозиції

При побудові моделі складної системи інформаційна система може бути представлена в найзагальнішому вигляді на контекстній діаграмі. [26]

Контекстна діаграма моделює процес створення плагіну для відеоплеєру. Для цього необхідні: інструменти для створення плагінів, документація по ним, стандарти створення, технічне завдання для плагіну, опис структури, фреймворк для навчання моделей згорткових штучних нейронних мереж, API відеоплеєру та програміст.

Побудована контекстна діаграма зображена на рисунку 2.6.

Рисунок 2.6 – Контекстна діаграма створення плагіну.

Деталізація головної функції системи здійснюється за допомогою діаграм декомпозиції, які будуються за тим самим принципом, що і контекстна діаграма, але включає більшу кількість робіт. Кожна робота, у свою чергу, може бути декомпонована. [27]

Процес створення плагіну декомпонується на такі роботи: створення архітектури та налаштування середовища, створення модуля інтерполяції, інтеграція модуля інтерполяції з API відеоплеєра, тестування та зневадження.

Для роботи по розробці структури та налаштування середовища необхідно: опис структури плагіну, технічне завдання, документація, стандарти та програміст. Результатом роботи буде розроблена структура плагіну та налаштоване середовище.

Для роботи по створенню модуля інтерполяції частоти кадрів необхідно: технічне завдання, документація, стандарти, структура плагіну, фреймворк машинного навчання та програміст. Результатом роботи буде розроблений модуль інтерполяції частоти кадрів.

Для роботи по інтеграції модуля інтерполяції частоти кадрів з API відеоплеєра необхідно: технічне завдання, документація, стандарти, модуль інтерполяції, API відеоплеєра та програміст. Результатом роботи буде розроблений плагін з вадами.

Для роботи по тестуванню плагіну необхідно: технічне завдання, документація, стандарти, плагін з вадами та програміст. Результатом роботи буде розроблений готовий плагін.

Побудована діаграма декомпозиції на рисунку 2.7.

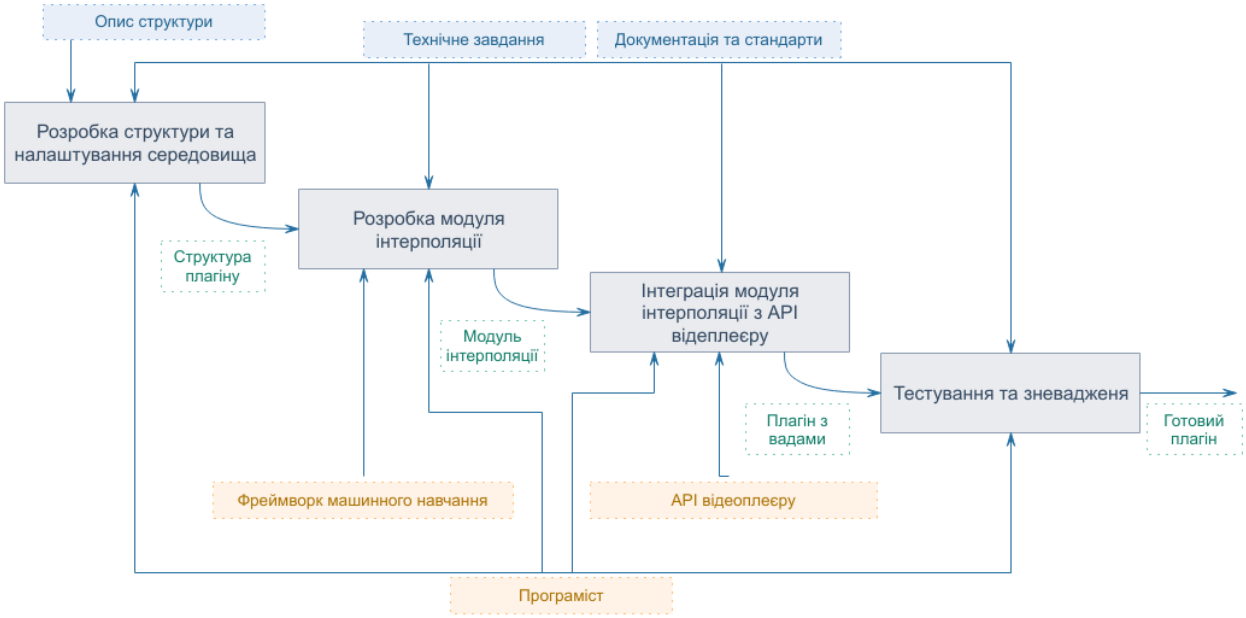

Рисунок 2.7 – Діаграма декомпозиції.

2.2.2 Діаграма дерева вузлів

Процес створення моделі робіт є ітераційним, отже, роботи можуть змінювати своє розташування в дереві вузлів багаторазово. Щоб не заплутатися й перевірити спосіб декомпозиції, треба після кожної зміни створювати діаграму дерева вузлів. [28]

Створення архітектури та налаштування середовища декомпозується на такі роботи: побудова архітекрури, налаштування середовища, вибір інструментів, налаштування системи контролю версій. [27]

Створення модуля інтерполяції декомпозується на такі роботи: вибір моделі штучної нейронної мережі для модуля інтерполяції, кодування модуля та його інтерфейсу, кодування алгориму склеювання кадрів, навчання моделі згорткової штучної нейронної мережі.

Інтеграція з API відеоплеєра декомпозується на такі роботи: ознайомлення з методами API, інтеграція модуля інтерполяції з відеоплеером.

Тестування і зневаджування декомпозується на такі роботи: тестування та валідація модуля інтерполяції, тестування та зневаджування плагіна цілком.

Побудована діаграма вузлів зображена на рисунку 2.8.

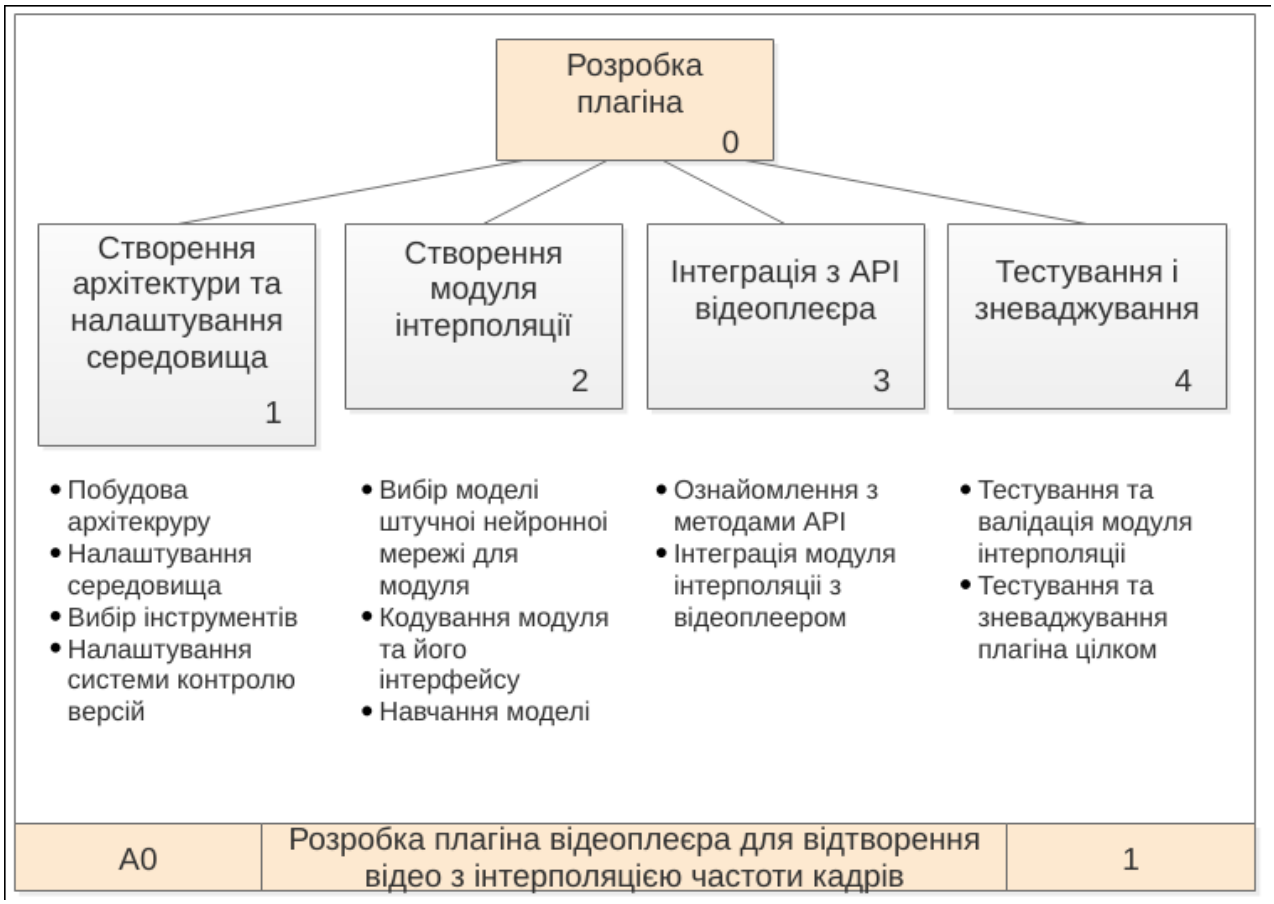

Рисунок 2.8– Діаграма вузлів.

2.2.3 Діаграма потоків даних

З допомогою діаграми потоків даних вимоги розбиваються на функціональні компоненти (процеси) і представляються у вигляді мережі, зв'язаної потоками даних. Головна мета таких засобів - продемонструвати, як кожен процес перетворить свої вхідні дані у вихідні, а також виявити відносини між цими процесами. [29]

Для створення архітектури та налаштування середовища вихідними даними є готова архітектура. Зовнішня сутність програміст надає потік кодування. Зовнішня сутність експерти на “Stack Overflow” надають потік думки експертів. Також процес потребує сховища даних з технічним завданням, описом структури, діаграмами, стандартами та документацією для використованих інструментів.

Для процесу створення модуля інтерполяції вхідними даними є готова архітектура, вихідними – модель інтерполяції. Зовнішня сутність програміст надає потік кодування. Зовнішня сутність експерти на “Stack Overflow” надають потік думки експертів. Також процес потребує сховища даних з технічним завданням, описом структури, діаграмами, стандартами та документацією для використованих інструментів.

Для процесу інтеграції з API відеоплеєра вхідними даними є модель інтерполяції, вихідними – плагін, що не пройшов зневадження та тестування. Зовнішня сутність програміст надає потік кодування. Зовніщня сутність експерти на “Stack Overflow” надають потік думки експертів. Також процес потребує сховища даних з технічним завданням, описом структури, діаграмами, стандартами та документацією для використованих інструментів.

Для тестування і зневаджування вхідними даними є плагін, що не пройшов зневадження та тестування, вихідними – готовий плагін. Зовнішня сутність програміст надає потік кодування.

Зовнішня сутність користувач використовує сховище даних, готовий програмний продукт (плагін), для користування. Також надає потік фідбеку до зовнішньої сутності програміста.

Побудована діаграма потоків даних згідно описаній послідовності дій традиційного системного аналізу, зображена на рисунку 2.10.

Рисунок 2.10 – Діаграма потоків даних.

2.2.4 Діаграма IDEF3 (потоків робіт)

Діаграми IDEF3 описують логіку роботи системи і взаємодію потоків у ній. На цій діаграмі система деталізується в більшій мірі, описується більш повно. У діаграму включають елементи логіки та елементи розділення на паралельні потоки. Це дозволяє моделювати послідовність виконання робіт у системі і описувати логічні зв' язки і послідовність між окремими процесами. [30]

У модельованій послідовності роботи по налаштуванню середовища, створення архітектури плагіну мають виконуватися послідовно.

Після цього роботи по створенню моделі інтерполяції частоти кадрів та роботи по тестуванню та зневадженню можуть виконуватись паралельно.

Після цього роботи по синхронізації відео та аудіо потоків, роботи по інтеграції модуля інтерполяції частоти кадрів з API відеоплеєра, роботи по

тестуванню та зневадженню також можуть виконуватись паралельно.

Побудована діаграма IDEF3 (потоків робіт) зображена на рисунку 2.11.

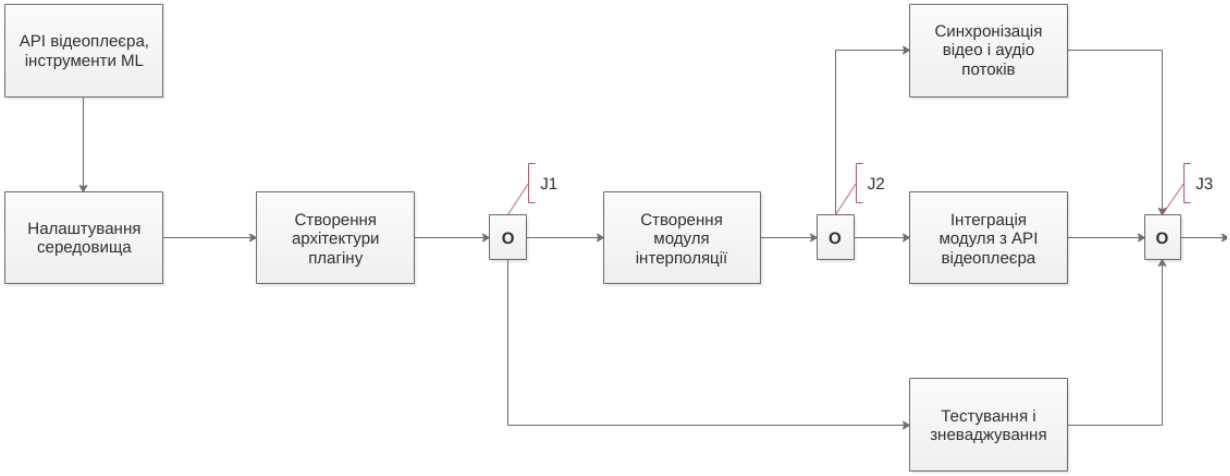

Рисунок 2.11 – Діаграма IDEF3 (потоків робіт).

2.3 Проектування

2.3.1 Діаграма варіантів використання

Основна мета створення будь-якої програмної системи - це створення програмного продукту, який допомагає користувачу виконувати свої повсякденні завдання. Для створення таких програм насамперед визначаються вимоги, яким повинна задовольняти система. [31]

Вимогами до модельованої системи є: можливість переглянути відео-файл у відеоплеєрі з підвищеною частотою кадрів, зберегти відео-файл з підвищеною частотою кадрів, відправити звіт про помилку та офірувати кошти на підтримку розробника.

Сценарії: переглянути відео-файл у відеоплеєрі з підвищеною частотою кадрів, зберегти відео-файл з підвищеною частотою кадрів, відправити звіт про помилку включають в себе сценарій увімкнення інтерполяції частоти кадрів.

Сценарій відправлення звіту про помилку включає в себе сценарій знайдення помилки.

Побудована діаграма варіантів використання зображена на рисунку 2.12.

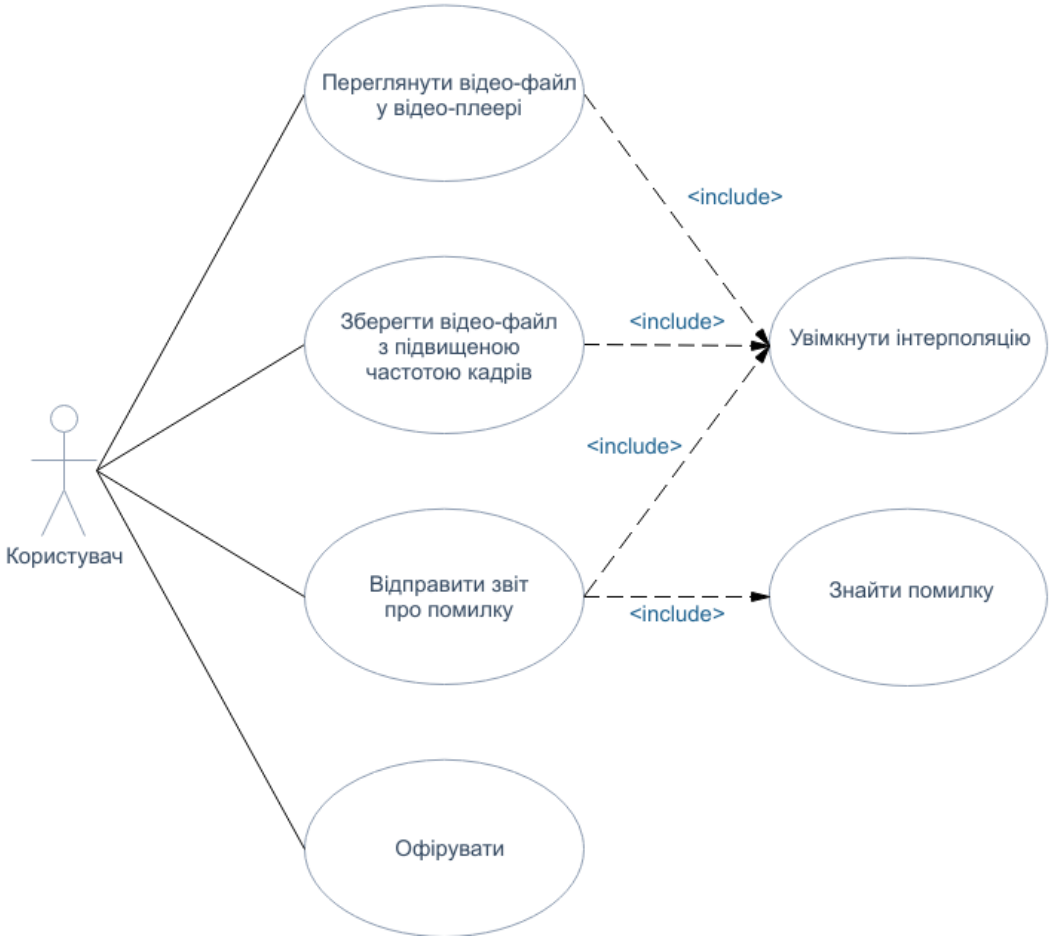

Рисунок 2.12 – Діаграма варіантів використання.

2.3.2 Діаграма класів

Діаграма класів — це тип діаграм, які частіше за все використовуються при моделюванні об’єктно-орієнтованих систем. Елементи діаграм класів пов’язані різними структурними зв’язками. [32]

Побудована діаграма класів зображена на рисунку 2.13.

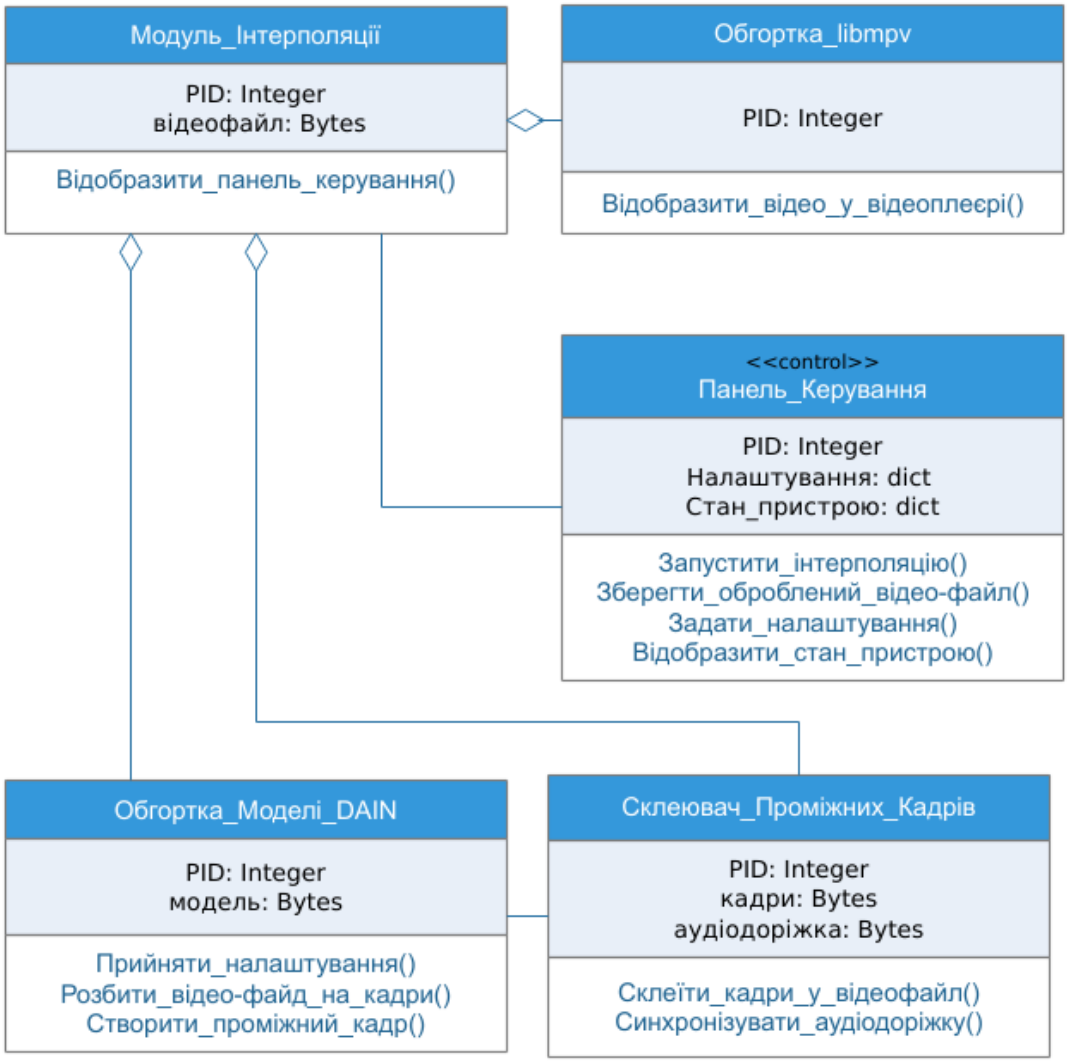

Рисунок 2.13 – Діаграма класів.

У модельованій системі є такі класи: “Модуль_Інтерполяції”, “Обгортка_libmpv”, “Панель_Керування”, “Обгортка_Моделі_DAIN” та “Склеювач_Проміжних_Кадрів”.

Клас “Панель_Керування” керує класом “Модуль_Інтерполяції”.

Також клас “Модуль_Інтерполяції” агрегує класи “Обгортка_libmpv”, “Обгортка_Моделі_DAIN” та “Склеювач_Проміжних_Кадрів”.

Клас “Модуль_Інтерполяції” має PID, містить відео-файл та метод відображення панелі керування.

Клас “Обгортка_libmpv” має PID, містить відео-файл та метод показу відео-файлу у відеоплеєрі.

Клас “Панель_Керування” має PID, містить налаштування, стан пристрою та методи, що дозволяють запустити інтерполяцію частоти кадрів,

зберегти оброблений відео-файл, задати налаштування плагіну, відобразити стан пристрою, а саме навантаження у відсотках та температуру процесора та відеокарти.

Клас “Обгортка_Моделі_DAIN” має PID, містить ваги зв’язків моделі згорткової штучної нейронної мережі DAIN та методи прийняття налаштувань, розбиття відео-файлу на кадри, створення нових проміжних кадрів.

Клас “Склеювач_Проміжних_Кадрів” має PID, містить кадри, аудіодоріжку та методи склеювання кадрів у відео-файл, метод синхронізації аудіодоріжки.

2.3.3 Діаграма пакетів

Діаграма пакетів — відображає залежності між пакетами, з яких і складається модель. [33]

Діаграми пакетів можуть використовувати пакети, які містять прецеденти для демонстрації функціональності програмного забезпечення системи. Діаграми можуть використовувати пакети, які показують різноманітні шари програмного комплексу для ілюстрації його архітектури, що складається з різних шарів. Залежності між цими пакетами можуть бути наділені позначками / стереотипами, щоби вказати механізм зв'язку між шарами. [33]

У модельованій системі є такі пакети: “libmpv”, “Dain-mpv плагін” та “DAIN модель”. У пакеті “Модуль_інтерполяції” міститься модуль “Модуль Інтерполяції”.

Пакети “libmpv” та “DAIN модель” є зовнішніми пакетами та взаємодіють з пакетом “Dain-mpv плагін” за допомогою обгорток у середині модуля “Модуль_інтерполяції”.

Побудована діаграма пакетів — на рисунку 2.14.

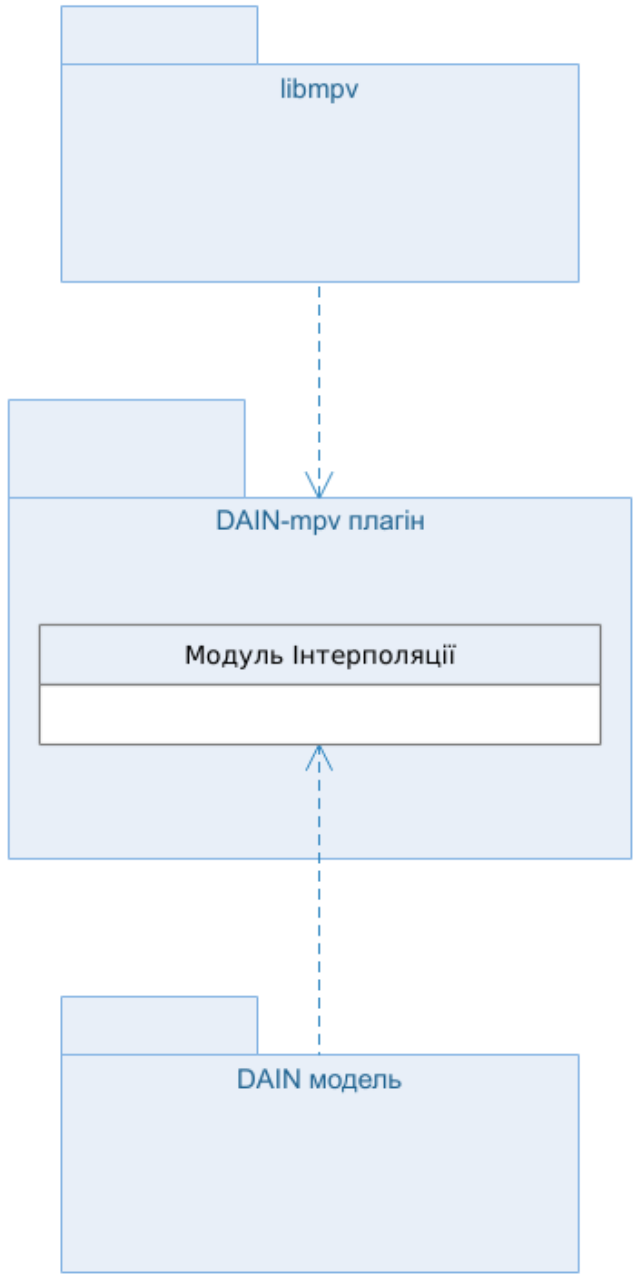

Рисунок 2.14 – Діаграма пакетів.

2.3.4 Діаграма станів

Діаграма станів описує процес зміни станів та моделює всі можливі зміни в стані конкретної системи. При цьому зміна стану системи може бути викликана зовнішніми впливами з боку інших систем або ззовні. Саме для опису реакції системи на подібні зовнішні впливи і використовуються діаграми станів. [34]

У модельованій системі є 5 станів: ініціалізація, пререндеринг відео, обробка і показ відео, завершення роботи плеєру. Зміна станів відбувається саме у такій послідовності, лінійно.

Стани змінюються через дії користувача, такі як відкриття файлу та закриття плеєру, та через зміни стану модуля інтерполяції, такі як завершення пререндерингу та завершення інтерполяції.

Стан ініціалізації викликається запуском відеоплеєра з плагіном. У цьому стані відбувається старт усіх процесів та відображення панелі керування плагіном. Зміна стану стається, коли усі процеси успішно запущено.

Стан пререндерингу відео викликається отриманням відео-файлу. У цьому стані відбувається інтерполяція частоти кадрів для відео-файлу. Зміна стану стається, коли оброблено достатньо кадрів, щоб показ відео відбувався без затримок.

Стан обробки і показу відео викликається, коли оброблено достатньо кадрів, щоб показ відео відбувався без затримок. У цьому стані відбувається інтерполяція частоти кадрів для відео-файлу та його показ. Зміна стану стається, коли не залишилось необроблених кадрів.

Стан показу відео викликається, коли не залишилось необроблених кадрів. У цьому стані відбувається показ відео-файлу. Зміна стану стається, коли відеоплеєр завершив показ відео-файлу.

Стан завершення роботи плеєру викликається, коли відеоплеєр завершив показ відео-файлу. У цьому стані відбувається завершення усіх процесів та очищення пам’яті. Зміна стану стається, коли відеоплеєр завершив роботу.

Побудована діаграма станів зображена на рисунку 2.15.

Рисунок 2.15 – Діаграма станів.

2.3.5 Діаграма активності та діаграма взаємодії

Діаграма діяльності — візуальне представлення графу, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Це графічне зображення робочих процесів поетапних дій та дій з підтримкою вибору, ітерації та одночасності. Діаграми діяльності призначені для моделювання як обчислювальних, так і організаційних процесів (тобто робочих процесів), а також потоків даних, що перетинаються із супутніми діями. Хоча діаграми діяльності в першу чергу показують загальний потік управління, вони також можуть включати елементи, що показують потік даних між видами діяльності через один або кілька сховищ даних. [35]

У модельованій системі існують наступні дії: запуск відеоплеєра, завантаження відеофайлу, ініціалізація процесу інтерполяції, ініціалізація процесу склеювання, обробка кадрів, склеювання відео, показ відео, завершення процесів.

Побудована діаграма активності зображена на рисунку 2.16.

Рисунок 2.16 – Діаграма активностей.

Завантаження відео-файлу, ініціалізація процесу інтерполяції та ініціалізація процесу склеювання кадрів відбуваються паралельно після дії запуску відеоплеєра. Також після завантаження відео-файлу та ініціалізації процесу інтерполяції кадрів відбувається дія обробки кадрів паралельно до дії ініціалізації процесу склеювання кадрів.

Після цього відбувається дія склеювання кадрів до моменту, коли оброблено достатньо кадрів для показу відео-файлу без затримок. Тепер дії склеювання кадрів та показу відео-файлу відбуваються паралельно до моменту завершення перегляду, або закінчення необроблених кадрів, у такому випадку далі відбувається дія показу відео до завершення перегляду.

Діаграми взаємодії — сукупність об’єктів, поведінка яких значуща для досягнення складових мети системи, та взаємовідношення тих ролей, які об’єкти відіграють у взаємодії.

Діаграма послідовностей показує взаємодії об'єктів, розташованих у часовій послідовності, зображує об'єкти та класи, що беруть участь у сценарії, та послідовність повідомлень, що обмінюються між об'єктами, необхідні для виконання функціональності сценарію.

Діаграми послідовності іноді називають діаграмами подій або сценаріями подій. [36]

У модельованій системі є такі взаємодіючи об’єкти: відеоплеєр, процес інтерполяції, процес склеювання. Ці об’єкти взаємодіють згідно сценарію обробки-інтерполяції кадрів для перегляду їх користувачем, а саме відеоплеєр віддає потокові кадри для інтерполяції, які у свою чергу віддаються на склеювання разом з створеними кадрами та повертаються назад до відеоплеєру у реальному часі, але показ відео-файлу починається лише тоді, коли процес інтерполяції не буде зупиняти перегляд для обробки.

Побудована діаграма діаграма взаємодії — на рисунку 2.17.

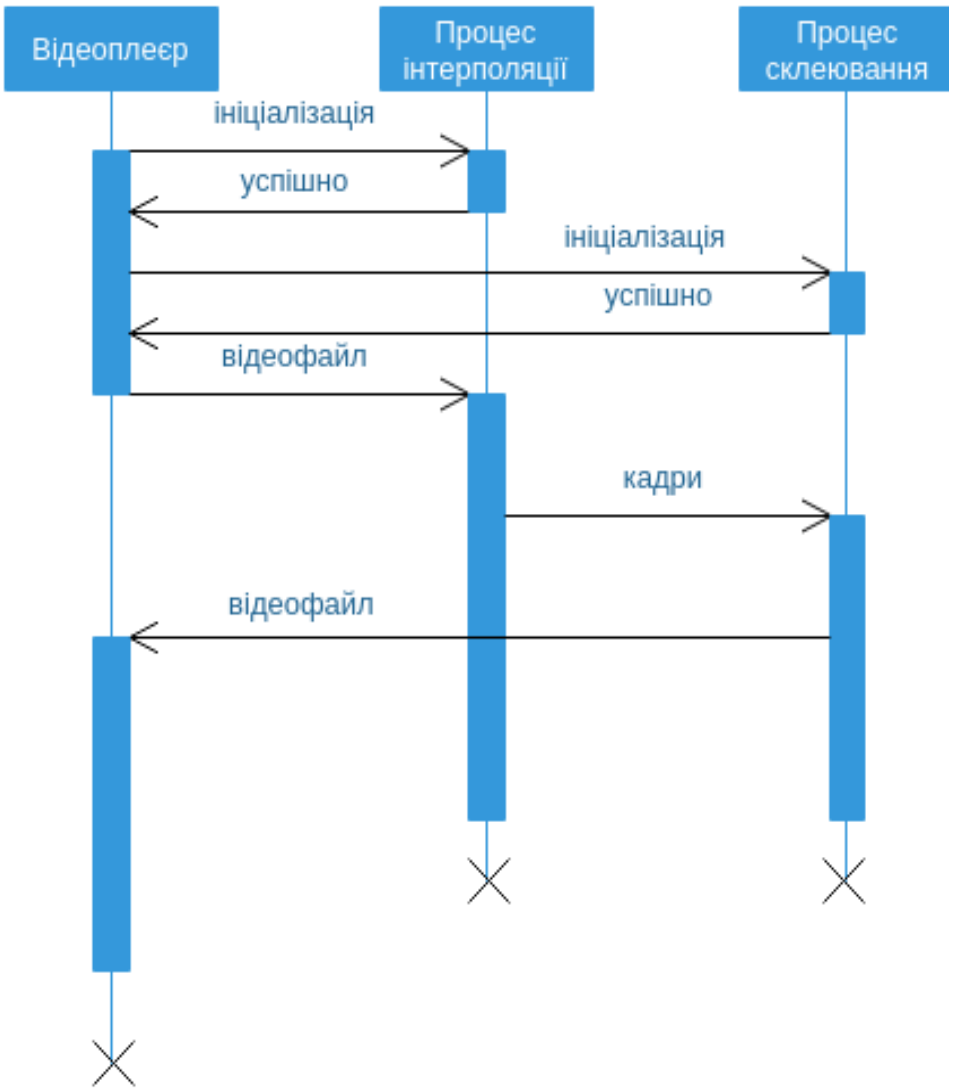

Рисунок 2.17 – Діаграма взаємодії.

2.4 Висновки по розділу

Були обрані та проаналізовані технології та інструментальні засоби розробки.

Був спроектований плагін для відеоплеєру, що застосовує метод інтерполяції частоти кадрів за допомогою сучасних методів обробки зображення та глибокого навчання до довільного відео-файлу.

Також виконаний системний аналіз інформаційних процесів для створюваного плагіну.

Були спроектовані діаграми класів плагіну та інші UML діаграми, на основі яких був розроблений цей плагін.

РОЗДІЛ 3 РОЗРОБКА ПЛАГІНУ

3.1 Розробка архітектури та налаштування середовища

3.1.1 Налаштування середовища

Для розробки плагіну відеоплеєру mpv для інтерполяції частоти кадрів за допомогою моделі згорткової штучної нейронної мережі DAIN буде використовуватись інтегроване середовище розробки для мови програмування Python, що має назву PyCharm, а саме PyCharm Community Edition (v2020.1.2). [50]

Для створення плагіну була обрана версія Python 3.7.8, як найновіша версія, що підтримується усіма необхідними інструментами. [19] [24] Для отримання та встановлення пакетів мови програмування Python використовується pip (v20.1.1), що являє собою систему керування пакетами, яка використовується для встановлення та управління програмними пакетами. [51]

Для створення плагіну для mpv відеоплеєру треба використовувати зовнішій (для мови прогрмаування Python) інструмент libmpv та пакет взаємодії з libmpv (v0.32.0), що має назву python-mpv (v0.4.6). [18] [40]

Для відображення панелі керування, необхідно мати зовнішний (для мови прогрмаування Python) інструмент Qt5 (v5.15) та наступні пакети PyQt: PyQt5 (v5.15.0), PyQt5-sip (v12.8.0), PyQt5-stubs (v5.14.2.2). [20] [49]

Для склеювання кадрів у відео-файл потрібно встановити opencv (v4.1.1) та інтерфейс для мови програмування Python, що має назву opencv-python (v3.4.2.17) для склеювання кадрів у відео-файл [55]

Для навчання та використання моделі згорткової штучної нейронної мережі DAIN необхідні наступні зовнішні інструменти: CUDA (>=v9.0), що являє собою програмно-апаратну архітектуру паралельних обчислень, яка дозволяє істотно збільшити обчислювальну продуктивність завдяки

використанню графічних процесорів фірми Nvidia [41]; cuDNN (>=v7.0), що являє собою бібліотеку примітивів для глибоких нейронних мереж, що використовує графічні процесори фірми Nvidia для обрахунків [42]; GCC (v4.9) [43] та NVCC (v9.0) [44] для компіляції пакетів, що входять до моделі DAIN. [24]

Також для використання PyTorch (v1.1.0) необхідні пакети екосистеми для машинного навчання мови програмування Python, а саме numpy (v1.18.4) для підтримки великих багатовимірних масивів і матриць [45], SciPy (v1.1.0) для оптимізації та інших загальних задач [46], Pillow (v7.1.2) для обробки зображень. [47] Також модель згорткової штучної нейронної мережі DAIN використовує пакет PWC-Net для операцій над зображеннями [24] [48], і базується на моделі MEMC-Net, що використовує адаптивний шар викривлення для розміщення об’єктів по осі Z. [24] [37]

Отже, розробка потребує встановлення зовнішніх (для мови прогрмаування Python) інструментів:

- PyCharm Community Edition (v2020.1.2);
- pip (v20.1.1);
- libmpv (v0.32.0);
- Qt5 (v5.15);
- CUDA (v9.0);
- cuDNN (v7.0);
- GCC (v4.9);
- NVCC (v9.0);
- OpenCV (v4.1.1)

Також пакетів мови прогрмаування Python, що були встановлені у віртуальне середовище:

- python-mpv (v0.4.6);
- PyQt5 (v5.15.0), PyQt5-sip (v12.8.0) та PyQt5-stubs (v5.14.2.2);
- PyTorch (v1.1.0);

– numpy (v1.18.4;
– SciPy (v1.1.0);
– Pillow (v7.1.2);
– opencv-python (v3.4.2.17).

та їх пакети-залежності.

3.1.2 Налаштування системи контролю версій

Система контролю версій необхідна для зберігання декількох версій плагіну відеоплеєру для інтерполяції частоти кадрів задля подальшого перегляду та відкатів до застарілих версій. Також передбачається розповсюдження плагіну як програмного продукту з відкритим вихідним кодом, а система контролю версій допоможе редагувати одні і тіж файли різними розробниками та вирішувати проблеми злиття незалежних версій.

Була обрана система контролю версій Git, як найбільш популярна та звична система котролю версій для більшості розробників. Git - це безкоштовна та з відкритим кодом розповсюджена система управління версіями, призначена для швидкого та ефективного управління всіма проектами від маленьких до дуже великих проектів. [52]

Був створений приватний репозиторій на github.com, порожня гілка master, до якої у процесі розробки додаються новий функціонал та виправлення.

Також був створений файл .gitignore, в якому вказана дерикторія з віртуальною середою розробки, що була створена у попередньому розділі.

Був створений файл requirements.txt, в якому вказані необхідні пакети мови програмування Python, що потребують встановлення для розробки.

Ініціалізований файл README.md з коротким описом проекту.

3.1.3 Побудова архітектури

Кодування відбувається згідно стандарту написання коду PEP 8 задля можливості розробки сторонніми розробниками. [56]

Архітектура плагіну відеоплеєру mpv для інтерполяції частоти кадрів за допомогою моделі згорткової штучної нейронної мережі DAIN розроблена згідно діаграми класів, що приведена у другому розділі, а саме:

Створений клас модуля інтерполяції частоти кадрів, що має назву InterpolationModule та знаходиться у файлі interpolation.py. Клас ініціалізується при старті плагіну та має викликати ініціалізування інших класів.

Створений клас обгортки для libmpv, що має назву LibmpbWrapper та знаходиться у файлі libmpv_wrapper.py. Клас ініціалізується у класі InterpolationModule та має впроваджувати взаємодію з пакетом python-mpv.

Створений клас обгортки для моделі згорткової штучної нейронної мережі DAIN, що має назву DAINModelWrapper та знаходиться у файлі DAIN_wrapper.py. Цей клас ініціалізується у класі InterpolationModule та має впроваджувати взаємодію з моделлю DAIN та виконувати генерацію проміжних кадрів за її допомогою.

Створений клас склеювача проміжних кадрів, що має назву FramesGluer та знаходиться у файлі frames_gluer.py. Цей клас ініціалізується у класі InterpolationModule та має перетворювати оригінальні та створені кадри у відео, додавати та синхронізувати оригінальну аудіодоріжку.

Створений клас панелі керування, що має назву ControlPanel та знаходиться у файлі control_panel.py. Клас ініціалізується у класі InterpolationModule та має впроваджувати налаштування та ініціювати процес інтерполяції частоти кадрів.

Також створені допоміжний клас для логерування, що має назву Logger та знаходиться у файлі logger.py. Цей клас ініціалізується при старті плаігну.

3.2 Розробка модуля інтерполяції частоти кадрів

3.2.1 Навчання моделі згорткової штучної нейронної мережі DAIN

Розробники моделі згорткової штучної нейронної мережі DAIN пропонують навчати модель на датасеті Vimeo90K, що був створений спеціально для навчання моделей ШНМ, які вирішують завдання інтерполяції частоти кадрів. Цей датасет містить спеціально відібрані і розкадровані відео-файли, які складно обробити методами комп’ютерного зору. [53]

Датасет Vimeo90K містить лише відео-файли реально знятих об’єктів та сцен [53]. Але плагін створюється для інтерполяції частоти кадрів для довільних відео-файлів, у тому числі анімованих об’єктів та сцен. Тому для інтерполяції частоти кадрів у відео-файлах, що були створені задопомогою комп'ютерної анімації модель потрібно навчати додатково та окремо зберігати ваги зв’язків моделі.

Датасет для тренування інтерполяції частоти кадрів для анімованих відео був зібраний, розкадрувавши відео-файли, які були анімовані з підвищеною частотою кадрів. Як тренувальні зразки були взяті кадри, з яких можна зібрати відео-файл з частотою кадрів у 24 кадри за секунду, а як валідуючі зразки взяті проміжні кадри між взятими для тренування кадрами. За допомогою алгебраїчних алгоритмів знаходження різниці між яскравістю та місцезнаходженням контурів [54], були відкинуті тренувальні і валідаційні кадри, що знаходились на межі між сценами у відео-файлі.

Налаштування ваг зв’язків для інтерполяції є змінним у межах натренерованих сетапів, їх зміна та додавання реалізується у панелі керування плагіном. Користувач повинен мати можливість надати власні ваги зв’язків.

3.2.2 Розробка обготки моделі ШНМ

Обгортка моделі згорткової штучної нейронної мережі DAIN має на меті представляти зручний інтерфейс для створення проміжного кадру для двох заданих кадрів з довільного відео.

На вхід клас обгортки приймає налаштування інтерполяції частоти кадрів та відео-файл, який буде розкадровано. Кадри в свою чергу будуть прийняті до черги на вхід моделі. Вхідні і вихідні кадри зберігаються у оперативній пам’яті, і будуть використані класом склеювачем кадрів для поєднання з аудіодоріжкою та показу у відеоплеєрі. Фрагмент коду для створення проміжних кадрів представлено у лістингу 3.1.

Лістинг 3.1 — Алгоритм створення проміжного кадру.

```
X0 =  torch.from_numpy( np.transpose(imread(arguments_strFirst) , (2,0,1)).astype("float32")/
255.0).type(dtype)
    X1 =  torch.from_numpy( np.transpose(imread(arguments_strSecond) , (2,0,1)).astype("float32")/
255.0).type(dtype)

    y_ = torch.FloatTensor()
    assert (X0.size(1) == X1.size(1))
    assert (X0.size(2) == X1.size(2))

    intWidth = X0.size(2)
    intHeight = X0.size(1)
    channel = X0.size(0)
        torch.set_grad_enabled(False)
    X0 = Variable(torch.unsqueeze(X0,0))
    X1 = Variable(torch.unsqueeze(X1,0))
    X0 = pader(X0)
    X1 = pader(X1)

    if use_cuda:
       X0 = X0.cuda()
       X1 = X1.cuda()
    proc_end = time.time()
    y_s,offset,filter = model(torch.stack((X0, X1),dim = 0))
    y_ = y_s[save_which]
```

Клас обгортки вигружається з пам’яті, залишаючи лише кадри, у той момент, коли не залишилось оригінальних кадрів, для яких треба згенерувати проміжний кадр.

3.2.3 Розробка алгориму склеювання кадрів

Алгоритм склеювання кадрів використовує бібліотеку opencv та інтерфейс opencv-python для процесу склеювання кадрів у реальному часі. Фрагмент коду цього алгоритму представлено у лістингу 3.2.

Лістинг 3.2 — Алгоритм склеювання кадрів.

```
for i in range(len(files)):
    img = images[i]
    height, width, layers = img.shape
    size = (width,height)
    frame_array.append(img)out = cv2.VideoWriter(video_out,
            cv2.VideoWriter_fourcc(*'DIVX'), fps, size)
```

Склеювання кадрів у відео і подальше синхронізація з аудио потоком відбувається і зберігається у опретивній пам’яті комп’ютера, у випадку, якщо не відбувається експорт відео-файлу.

Показ відео починається лише після оброки і склейки достатньої кількості кадрів для показу без затримок.

У результаті штатної роботи плагіну, відеоплеєр показує відео без затримок та розсинхронізації звуку або субтитрів.

3.2.4 Розробка інтерфейсу панелі керування

Графічний інтерфейс плагіну відеоплеєру mpv для інтерполяції частоти кадрів за допомогою моделі згорткової штучної нейронної мережі DAIN буде складатися з панелі керування, за допомогою якої можна керувати плагіном та слідкувати за станом пристрою.

Інтерфейс панелі керування є інтуїтивно зрозумілим та зручним для використання.

Панель керування плагіном, екранний вигляд у редакторі QtCreator якої представлено на рисунку 3.1.

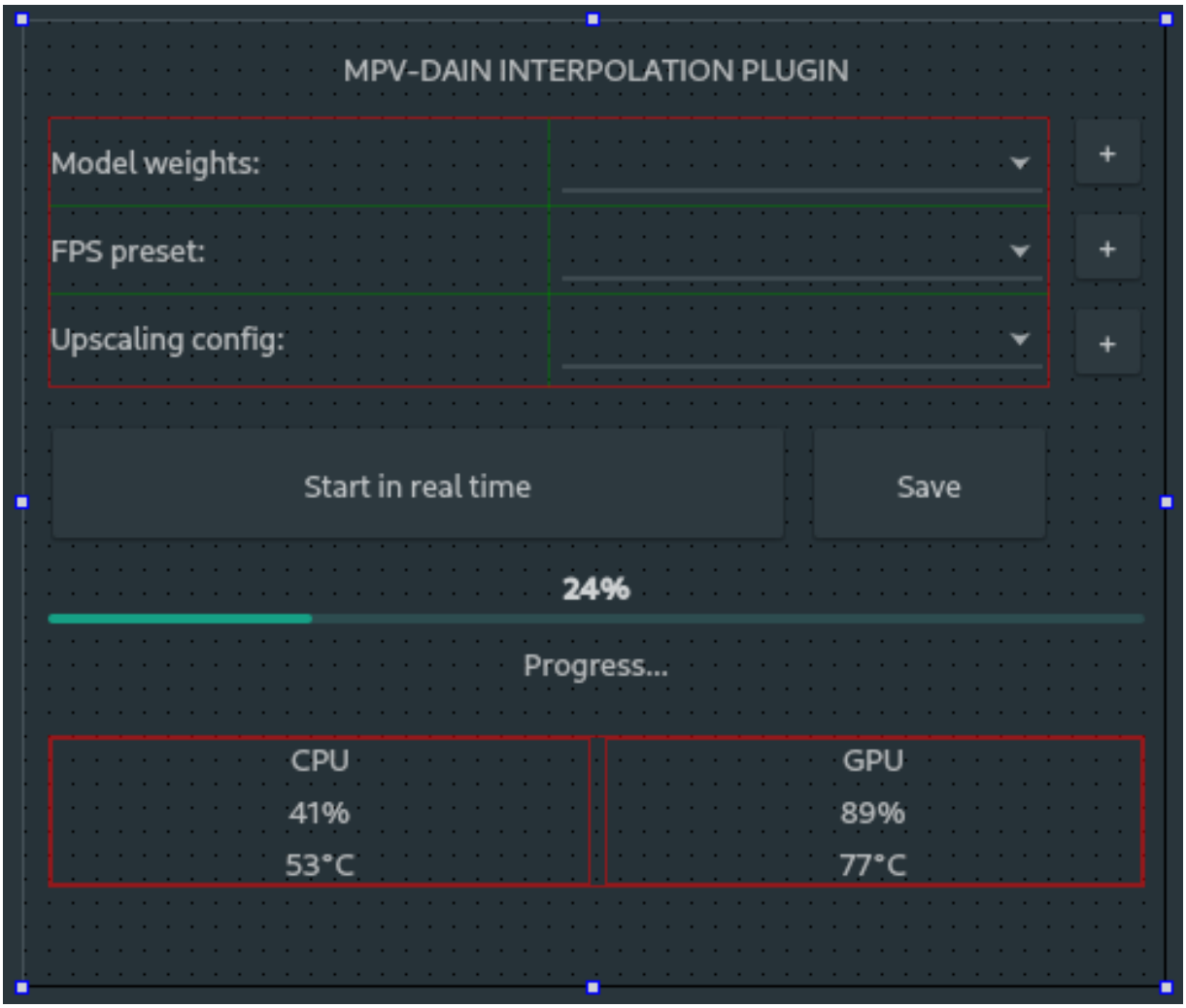

Рисунок 3.1 – Розробка інтерфейсу панелі керування плагіном.

Панель керування плагіном містить:

– Кнопку запуску інтерполяції;

– Кнопку експорту інтерпольованого відео;

– Вибір пресету ваг зв’язків моделі;

– Вибір кількості створюваних кадрів;

– Вибір конфігураційного файлу постобробки відеоплеєра mpv.

– Лінію прогресу інтерполяції частоти кадрів;

– Інформацію про стан пристрою.

Розробка панелі за допомогою PyQt дозволить запускати панель на будь-якій операційній системі, тому що обертає платформонезалежний Qt5 у інтерфейс на мові програмування Python. [20]

3.3 Розробка інтеграції модуля інтерполяції з API відеоплеєра

3.3.1 Створення обгортки для API відеоплеєра

Обгортка для API відеоплеєра, а саме для python-mpv використовує PyQt5 для відображення вікна відеоплеєра. Основною функцією обгортки є забезпечення взаємодії між модулем інтерполяції, що отримує відео-файл від відеоплеєра, та повертає для показу відео-файл з підвищеною частотою кадрів. Код показу відео з підвищеною частотою кадрів приведено у лістингу 3.3.

Лістинг 3.3 — Код показу відео з підвищеною частотою кадрів.

```
self.container = QWidget(self)
self.setCentralWidget(self.container)
self.container.setAttribute(Qt.WA_DontCreateNativeAncestors)
self.container.setAttribute(Qt.WA_NativeWindow)
player = mpv.MPV(wid=str(int(self.container.winId())), vo='x11', log_handler=print, loglevel='debug', osc=True,
        player_operation_mode='pseudo-gui', input_default_bindings=True, input_vo_keyboard=True)
player.play(interpolated_video)
```

3.3.2 Інтеграція обготки для API відеоплеєра з модулем інтерполяції

Інтеграція модуля інтерполяції з API відеоплеєра представляє собою взаємодію між класом обгортки для libmpv, що має назву LibmpbWrapper та класом обгортки для моделі згорткової штучної нейронної мережі DAIN, що має назву DAINModelWrapper усередині класу InterpolationModule. Згідно діаграмі взаємодії, що приведена у другому розділі, LibmpbWrapper віддає відео-файл класу DAINModelWrapper, у свою чергу DAINModelWrapper повертає кадри назад, клас склеювача проміжних кадрів, повертає відео-файл назад до класу LibmpbWrapper.

3.4 Тестування і представлення

3.4.1 Тестування і представлення модуля інтерполяції

На рисунку 3.2 представлено результат тестування модуля інтерполяції, а саме створення 28 проміжних кадрів між двох оригінальних.

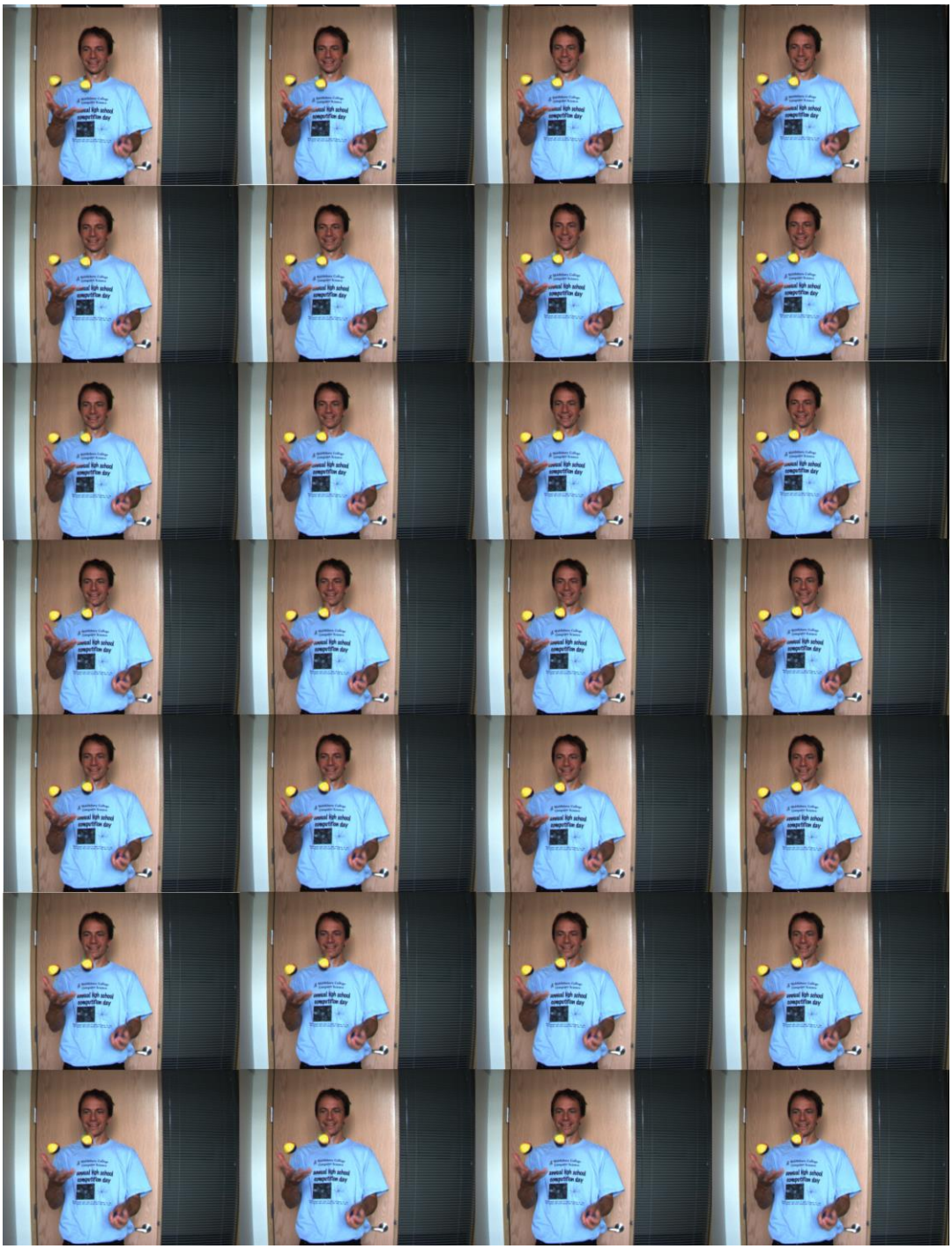

Рисунок 3.2 – 28 згенерованих кадрів між двох оригінальних

На рисунку 3.3 предсавлений оригінальний початковий кадр, на рисунку 3.4 представлено згенарований проміжний кадр.

Рисунок 3.3 – Оригінальний початковий кадр.

Рисунок 3.4 – Згенарований проміжний кадр.

3.4.2 Тестування і представлення плагіну

Екранний вигляд запущеного відеоплеєра mpv разом із запущеним плагіном, що чекає отримання відео-файлу представлений на малюнку 3.5.

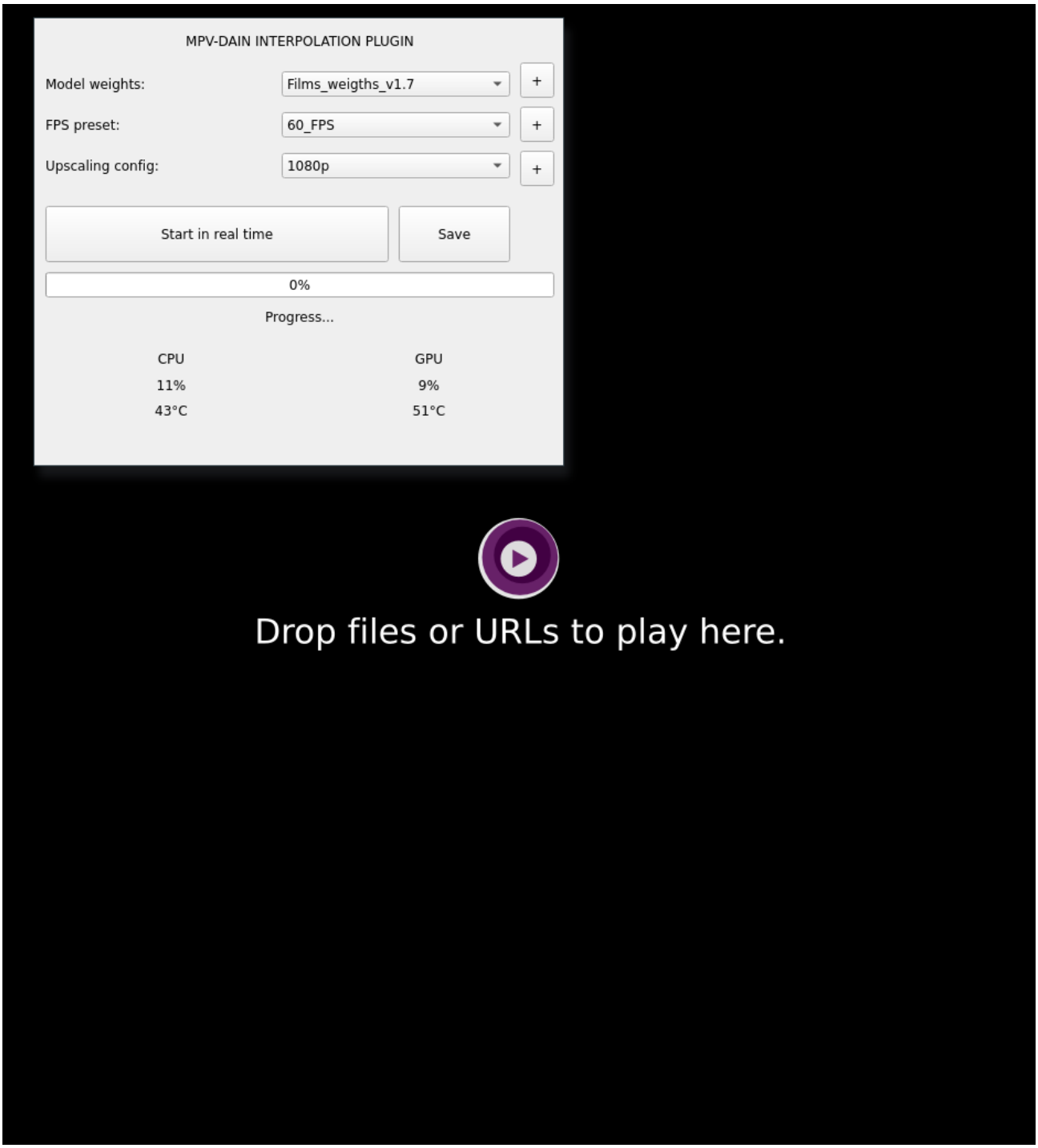

Рисунок 3.5 – Екранний вигляд запущеного відеоплеєра mpv разом із запущеним плагіном, що чекає отримання відео-файлу.

Екранний вигляд запущеного відеоплеєра mpv разом із запущеним плагіном, що інтерполює частоту кадрів для обраного відео-файлу у реальному часі представлений на малюнку 3.6.

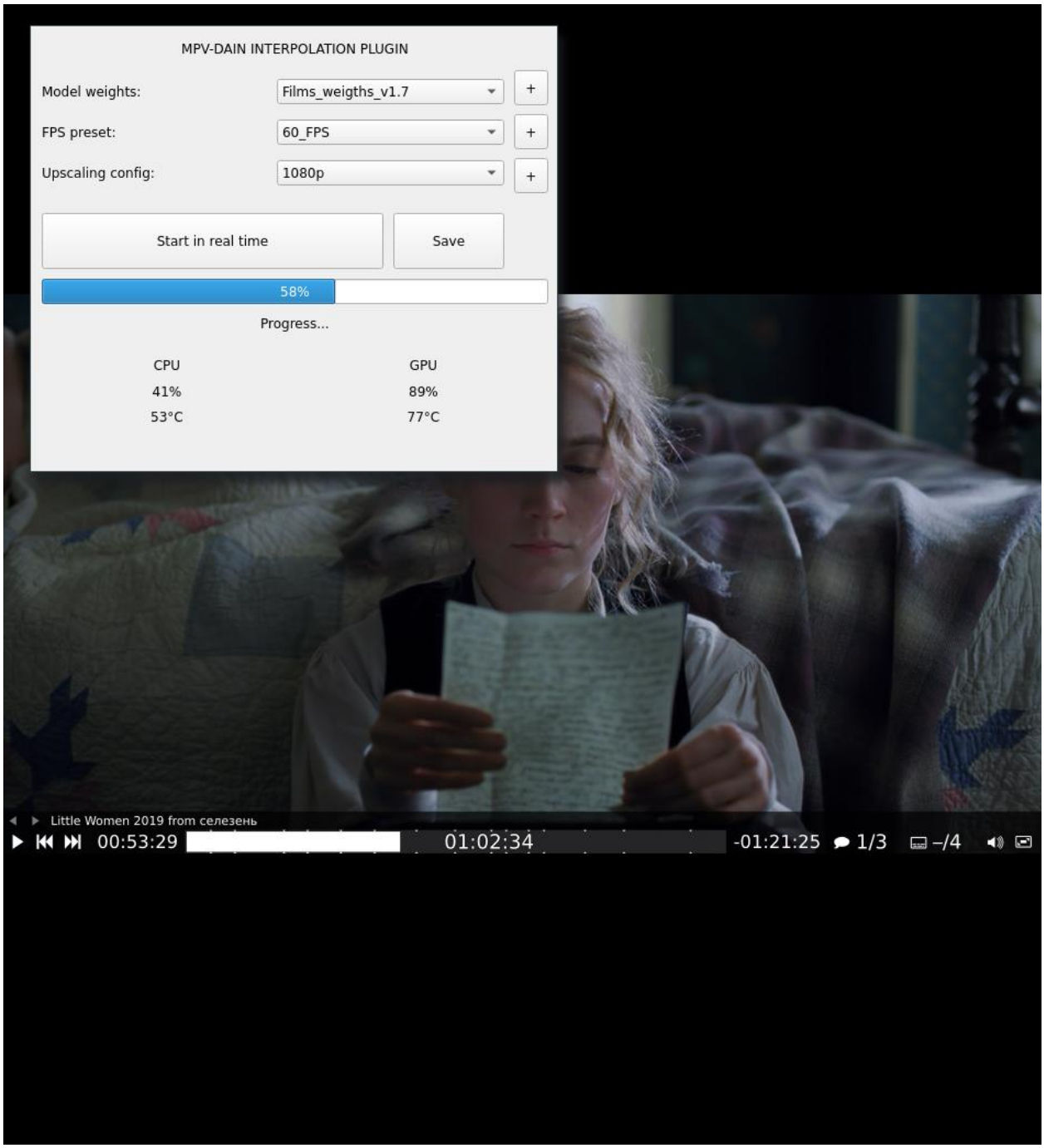

Рисунок 3.6 – Екранний вигляд запущеного відеоплеєра mpv разом із запущеним плагіном.

3.5 Висновки по розділу

Була розроблена архітектура плагіну відеоплеєру mpv для інтерполяції частоти кадрів за допомогою моделі згорткової штучної нейронної мережі DAIN, налаштоване середовище та система контролю версій.

Також був розроблений дизайн інтерфейсу плагіну, модуль інтерполяції частоти кадрів за допомогою моделі згорткової штучної нейронної мережі DAIN, цей модуль був інтегрований з API відеоплеєра mpv.

Був реалізований плагін для відеоплеєру згідно моделюванню та проектуванню у другому розділі.

Також виконання інтерполяції частоти кадрів у плагіні було протестовано.

ВИСНОВКИ

Було спроектовано та реалізовано плагін відеоплеєру для інтерполяції частоти кадрів для довільного відео-файлу.

Було проаналізовано предметну область, методи розв’язання проблеми, існуючі рішення а також технології та інструментальні засоби вирішення задачі.

Було виконано проектування всіх модулів розроблюваного плагіну за допомогою засобів UML та системного аналізу інформаційних процесів. Були використанні контекстна діаграма, діаграма декомпозиції, діаграма дерева вузлів, діаграма потоків даних, діаграма IDEF3, діаграма варіантів використання, діаграма пакетів, діаграма класів, діаграма станів, діаграма активностей та діаграма взаємодії. компонентів для проектування ядра системи на абстрактному рівні та діаграма класів для більш детального. Спроектовано модулі базової реалізації та інтерфейсу користувача.

Плагін було реалізовано за допомогою мови програмування Python версії 3, фреймворку глибокого навчання моделей штучних нейронних мереж PyTorch, платформонезалежного відеоплеєра mpv та його API libmpv, інтерфейс користувача за допомогою обгортки для Qt5, що має назву PyQt.

Розроблений плагін відповідає поставленій меті оскільки він успішно інтерполює частоту кадрів для довільного відео-файлу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Enticknap L. Moving Image Technology: From Zoetrope to Digital [Електронний ресурс] / Leo Enticknap. – 2005. – Режим доступу до ресурсу: https://www.academia.edu/170028/Moving_Image_Technology_From_Zoetrope_to_Digital.

2. 24 fps: how frame rate has become the standard and whether to change it [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: http://www.cinemotionlab.com/novosti/24_fps_kak_chastota_kadrov_stala_standartom_i_nuzhno_li_ee_izmenyat/.

3. Frame rate [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Frame_rate.

4. Navagin A. How many frames per second does the human eye see? [Електронний ресурс] / Alexander Navagin. – 2019. – Режим доступу до ресурсу: https://hype.tech/@ns3230/skolko-kadrov-v-sekundu-vidit-chelovecheskiy-glaz-4ylktl9l.

5. Elistratov V. How many frames per second does the human brain perceive [Електронний ресурс] / Vadim Elistratov. – 2017 – Режим доступу до ресурсу: https://dtf.ru/gamedev/3705-skolko-kadrov-v-sekundu-vosprinimaet-chelovecheskiy-mozg.

6. Meyer M. Restoration of Motion Picture Film / M. Meyer, P. Read., 2000. – 360 с.

7. Potter C. M. Detecting meaning in RSVP at 13ms per picture [Електронний ресурс] / C. M. Potter, C. E. Hagmann, S. E. McCourt // Springer US. – 2013. – Режим доступу до ресурсу: https://dspace.mit.edu/bitstream/handle/1721.1/107157/13414_2013_605_ReferencePDF.pdf;jsessionid=59F4CEA945929390C26DB529C3972961?sequence=1.

8. High Frame-Rate Television [Електронний ресурс] / M.Armstrong, D. Armstrong, M. Hammond, S. Jolly // BRITISH BROADCASTING

CORPORATION. – 2008. – Режим доступу до ресурсу: http://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP169.pdf.

9. Compare frames per second: which looks better? [Електронний ресурс] – Режим доступу до ресурсу: http://frames-per-second.appspot.com/.

10. Motion interpolation [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Motion_interpolation.

11. Beyers B. Help Key: Why 120Hz looks "weird" [Електронний ресурс] / Berkeley Beyers – Режим доступу до ресурсу: https://techcrunch.com/2009/08/12/help-key-why-hd-video-looks-weird/.

12. Ulyanov D. IMAGE GENERATION WITHCONVOLUTIONAL NEURAL NETWORKS : дис. докт. техн. наук : ВАК 05.13.15 / Ulyanov Dmitry – Moscow, 2019. – 133 с.

13. Python (programming language) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Python_(programming_language).

14. Python ecosystem [Електронний ресурс] – Режим доступу до ресурсу: https://curlie.org/Computers/Programming/Languages/Python.

15. Python.org [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/.

16. Mpv (media player) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Mpv_(media_player).

17. Mpv manual [Електронний ресурс] – Режим доступу до ресурсу: https://mpv.io/manual/master/.

18. Mpv user scripts [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/mpv-player/mpv/wiki/User-Scripts.

19. PyQt [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PyQt.

20. Fitzpatrick M. Qt5 Python GUI Programming Cookbook: Building responsive and powerful cross-platform applications with PyQt / Martin Fitzpatrick. – Manchester, 2019. – 258 с.

21. PyTorch [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/PyTorch.

22. PyTorch ecosystem [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/ecosystem/.

23. Autograd: Automatic Differentiation [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html.

24. DAIN (Depth-Aware Video Frame Interpolation) [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/baowenbo/DAIN.

25. Depth-Aware Video Frame Interpolation [Електронний ресурс] / W.Bao, W. Lai, C. Ma, Z. Gao // Institute of Image Communication and Network Engineering. – 2019. – Режим доступу до ресурсу: https://arxiv.org/pdf/1904.00830.pdf.

26. System context diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/System_context_diagram.

27. Decomposition (computer science) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Decomposition_(computer_science).

28. Tree structure [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Tree_structure.

29. Data-flow diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Data-flow_diagram.

30. IDEF3 [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/IDEF3.

31. Use case diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Use_case_diagram.

32. Class diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Class_diagram.

33. Package diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Package_diagram.

34. State diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/State_diagram.

35. Activity diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Activity_diagram.

36. Sequence diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Sequence_diagram.

37. MEMC-Net: Motion Estimation and MotionCompensation Driven Neural Network forVideo Interpolation and Enhancement [Електронний ресурс] / W.Bao, W. Lai, C. Ma, X. Zhang // IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. – 2018. – Режим доступу до ресурсу: https://arxiv.org/pdf/1810.08768.pdf.

38. Niklaus S. Context-aware Synthesis for Video Frame Interpolation [Електронний ресурс] / S. Niklaus, F. Liu // IEEE Conference on Computer Vision and Pattern Recognition. – 2018. – Режим доступу до ресурсу: http://www.sniklaus.com/ctxsyn.

39. Simon N. Video Frame Interpolation via Adaptive Separable Convolution [Електронний ресурс] / N. Simon, L. Mai, F. Liu // IEEE International Conference on Computer Vision. – 2017. – Режим доступу до ресурсу: https://arxiv.org/abs/1708.01692.

40. python-mpv [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/jaseg/python-mpv.

41. CUDA [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/CUDA.

42. NVIDIA cuDNN [Електронний ресурс] – Режим доступу до ресурсу: https://developer.nvidia.com/cudnn.

43. GNU Compiler Collection [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/GNU_Compiler_Collection.

44. NVIDIA CUDA Compiler [Електронний ресурс] – Режим доступу до ресурсу: https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html.

45. NumPy [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/.

46. SciPy.org — SciPy library [Електронний ресурс] – Режим доступу до ресурсу: https://www.scipy.org/.

47. Pillow — Pillow (PIL Fork) 7.1.2 documentation [Електронний ресурс] – Режим доступу до ресурсу: https://pillow.readthedocs.io/en/stable/.

48. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume [Електронний ресурс] / D.Sun, X. Yang, M. Liu, J. Kautz // NVIDIA. – 2018. – Режим доступу до ресурсу: https://arxiv.org/pdf/1709.02371.pdf.

49. Qt | Cross-platform software development [Електронний ресурс] – Режим доступу до ресурсу: https://www.qt.io/.

50. PyCharm: IDE [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/pycharm/.

51. pip (package manager) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Pip_(package_manager).

52. Git --local-branching-on-the-cheap [Електронний ресурс] – Режим доступу до ресурсу: https://git-scm.com/.

53. Vimeo90K | Video Enhancement with Task-Oriented Flow [Електронний ресурс] – Режим доступу до ресурсу: https://xinntao.github.io/open-videorestoration/rst_src/datasets_sr.html.

54. Войтенко В. П. Розпізнавання образів та обробка зображень. Методичні вказівки до виконання лабораторних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення» / Войтенко Володимир Павлович. – Чернігів: ЧНТУ, 2020. – 149 с.

55. opencv-python [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/opencv-python/.

56. PEP 8 -- Style Guide for Python Code [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/dev/peps/pep-0008/.