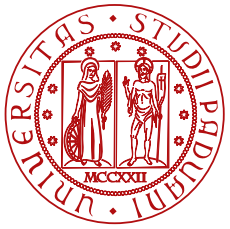


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in Ingegneria Informatica

Sviluppo e testing di Software conforme allo standard IEEE 3302-2022 per una codifica audio ad alta qualità per la conservazione del patrimonio culturale musicale utilizzando un approccio Test-Driven Development

Laureando

Alberto Pasqualetto

Matricola 2000162

Relatore

Prof. Sergio Canazza

Università degli Studi di Padova

Correlatori

Dott. Alessandro Russo

Università degli Studi di Padova

Dott. Matteo Spanio

Audio Innova

Anno Accademico

2022/2023

Sommario

L'uomo ha sempre sentito la necessità di tutelare le proprie opere; al giorno d'oggi la tecnica per preservare i manufatti dall'usura e dall'obsolescenza è la digitalizzazione.

Questo lavoro presenta lo stato dell'arte della digitalizzazione di audio analogico, nello specifico registrato su nastri magnetici a bobina aperta (open-reel tapes). Uno dei massimi esponenti nell'ambito è il Centro di Sonologia Computazionale del Dipartimento di Ingegneria dell'Informazione (DEI) dell'Università degli Studi di Padova, il quale ha sviluppato un software atto a riconoscere le discontinuità presenti su un nastro magnetico e rimediare agli errori di velocità ed equalizzazione nella traccia audio registrata, automatizzando un lavoro ripetitivo precedentemente svolto da operatori umani. Il software è ora parte di un framework appartenente al gruppo Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) ed è stato adottato dall'IEEE come standard col nome MPAI-CAE-ARP.

Lo scopo di quest'opera è trasporre in codice i test di conformità richiesti da MPAI utilizzando un approccio di Test Driven Development. Il testo descrive lo sviluppo dei test richiesti e come sono stati risolti alcuni problemi pre-esistenti nel codice.

Il codice sorgente citato ed utilizzato per quest'opera si trova nella repository GitHub:
`https://github.com/albertopasqualetto/Tesi-triennale`.

Indice

Elenco delle Figure

Elenco delle Tabelle

Elenco dei Frammenti di Codice

Elenco degli Acronimi

IA Intelligenza Artificiale

CSC Centro di Sonologia Computazionale

DEI Dipartimento di Ingegneria dell'Informazione

IEEE Institute of Electrical and Electronic Engineers

MPEG Moving Picture Experts Group

MPAI Moving Picture, Audio and Data Coding by Artificial Intelligence

MPAI-AIF AI framework

AIW AI workflow

AIM AI module

MPAI-CAE Context-based Audio Enhancement

EES Emotion Enhanced Speech

ARP Audio Recording Preservation

SRS Speech Restoration System

EAE Enhanced Audioconference Experience

ips inches per second

CCIR International Radio Consultative Committee, Comitè consultatif international pour la radio

NAB National Association of Broadcasters

KB Knowledge Base

DS dataset

TDD Test Driven Development

DRY Don't repeat yourself

PR Pull Request

MIME Multipurpose Internet Mail Extensions

MFCC Mel Frequency Cepstral Coefficients

PSNR peak signal-to-noise ratio

SSIM Structural similarity index measure

1

Digitalizzazione di audio analogico

Preservare dall'oblio il proprio patrimonio culturale è sicuramente una delle necessità più antiche dell'uomo.

Al contrario della conservazione passiva, che consiste nella sola salvaguardia dei documenti nella loro dimensione fisica; l'unica soluzione per preservare a lungo termine il materiale analogico è la digitalizzazione, ovvero una conservazione di tipo attivo. Tra il materiale da digitalizzare, quello audio-visivo è sicuramente uno dei più complicati per via della necessità di preservare sia il suo stato e la sua performance attuali, seppur il documento sia rovinato dal tempo o dall'usura, compreso di tutte le informazioni accessorie, sia di disporre di una copia cosiddetta "di accesso" in maniera tale da essere riprodotta agilmente con tecnologie odierne.

Molte organizzazioni quali archivi, librerie e musei non hanno ancora messo in pratica misure atte a preservare a tempo indeterminato il loro patrimonio culturale e ciò si verifica non solo nelle nazioni a più basso sviluppo umano come quelle dell'Africa subsahariana [18], ma anche in nazioni a più alto indice di sviluppo umano come l'Italia [17] presentando criticità non del tutto dissimili come, per esempio, la mancanza di fondi per via della poca importanza riposta nella causa.

I musei che negli anni e soprattutto con l'avvento del COVID-19 si sono dovuti adeguare introducendo la tecnologia digitale dapprima nelle loro strutture per migliorare l'esperienza dei visitatori e poi in rete creando delle mostre virtuali generando anche maggiori visite ed incassi [17], ora possono sfruttare la digitalizzazione con lo scopo accessorio di creare nuove esperienze online. Questo

sforzo è anche in linea con le raccomandazioni dell'UNESCO [21] in riferimento all'importanza della tecnologia in ambito educativo e culturale.

1.1 Filologia e Fedeltà

La maniera più basilare di procedere alla digitalizzazione è registrare soltanto l'informazione primaria ovvero, nel caso di un prodotto sonoro, il segnale audio. Per digitalizzare, invece, il prodotto in maniera filologicamente corretta è necessario memorizzare anche le informazioni ausiliarie come le annotazioni sul contenitore o sul supporto, i rumori presenti sul sistema di registrazione originale, le alterazioni fisiche del supporto, il marchio e modello del supporto ed altri metadati, oltre alla storia del tramandamento del documento (archiviazione, duplicazione, ecc.) [16, p. 59].

Per ottenere il massimo livello di aderenza alla realtà: la fedeltà, si deve assicurare, oltre che una riproduzione audio fedele, anche una simulazione dell'esperienza di interazione col dispositivo di riproduzione ed una riproduzione dei metadati e delle informazioni contestuali come ad esempio mostrato in Fantozzi et al. [2, cap. 3].

In certi casi può essere utile operare delle modifiche alla traccia digitalizzata o ad una sua copia nel caso in cui ci siano degli errori o delle corruzioni nel supporto originario o nella digitalizzazione. Tali modifiche possono essere correttive atte a ripristinare la corretta riproduzione secondo i canoni dettati dalla filologia, oppure possono essere utili a creare una copia di accesso a discapito della fedeltà assoluta.

1.2 Digitalizzazione manuale e automatizzazione

Il trasferimento da analogico a digitale dipende ancora dall'esperienza dell'operatore (dalle sue valutazioni e scelte se intervenire o meno), ciò può comportare l'introduzione di errori indesiderati causati dalla perdita di attenzione umana in seguito a numerose ore di lavoro con riflessi negativi sul valore dei documenti creati e sull'affidabilità dell'intera collezione. Per questo motivo è importante l'automatizzazione delle attività ripetitive in modo da minimizzare tali errori, ma anche da tagliare i costi e risparmiare tempo di lavoro [2, cap. 2.1; 10, es. 5].

1.3 Il Centro di Sonologia Computazionale ed il suo metodo di digitalizzazione

Il Centro di Sonologia Computazionale (CSC) del Dipartimento di Ingegneria dell'Informazione (DEI) dell'Università di Padova, che da circa 50 anni opera nella ricerca in ambito acustico [1], è uno dei poli più all'avanguardia per quanto concerne la digitalizzazione di supporti musicali, nello specifico di nastri magnetici a bobina aperta (open reel tape) ed è uno dei maggiori sostenitori dell'utilizzo di automazioni nel processo di trasporto da analogico a digitale.

Dato che documentare il processo che ha generato la copia di conservazione è importante nel campo dell'audio, visto che il supporto originario potrebbe diventare irrecuperabile in futuro, il CSC negli anni, collaborando con diversi archivi digitali, si è impegnato a sviluppare una tecnica di digitalizzazione "filologicamente informata" e a documentare la procedura in maniera accurata, verificabile ed oggettiva con l'obiettivo di creare uno standard.

L'innovazione principale in questo protocollo è la registrazione video del nastro magnetico contemporaneamente alla registrazione della traccia audio: viene effettuata la ripresa ai fini di catturare le immagini del nastro in corrispondenza della testina di registrazione ed in corrispondenza del capstan, le regioni interessate si possono trovare in figura 1.1.

La porzione vicina al capstan[1] serve ad individuare l'inizio della registrazione in quanto esso trasla verso il nastro nel momento in cui il motore viene azionato; mentre la sezione davanti alla testina di registrazione[2] serve ad individuare, tramite tecniche di intelligenza artificiale, tutte le alterazioni del retro del nastro che, da ora in poi, chiameremo "irregolarità"; alcuni esempi sono giunte, scritte e parti visivamente danneggiate o rovinate, vedi figura 1.2 [4]. Questi sono parte dei metadati citati all'inizio di questa sezione. L'utilità di raccogliere tali dati legati alla traccia audio è quella, oltre che di catturare il documento nella sua essenza più completa, anche di poter ricollegare la performance di una qualsiasi porzione della traccia audio ad una possibile irregolarità sul nastro ed eventualmente agire di conseguenza.

La procedura creata dal CSC prevede anche una gestione di tracce audio

[1] Riquadro rosso in figura 1.1

[2] Riquadro verde in figura 1.1

catturate a velocità o equalizzazione incorrette:

La velocità, misurata in pollici per secondo (*inches per second*, ips in inglese), è solitamente impostata a sottomultipli binari di $30\,\mathrm{ips}$ e viene regolata in base alla qualità di registrazione ed alla durata di un nastro che l'artista ha voluto ottenere, in generale maggiore è la velocità, maggiore sarà la qualità di registrazione e minore la durata del nastro. Può accadere che nello stesso nastro digitalizzato siano presenti tracce a velocità diverse; fra i motivi di tali irregolarità ci sono l'aggiunta in fase di montaggio di segmenti di altro nastro registrato a differenti velocità, la diminuzione della velocità verso la fine della registrazione, così da non dover cambiare supporto durante la performance, o la presenza di più contenuti non correlati tra loro registrati all'interno dello stesso nastro.

Per quanto riguarda l'equalizzazione invece, le curve di equalizzazione[3] solitamente sono quelle dettate dagli standard CCIR, prevalentemente utilizzato in Europa, e NAB, prevalentemente utilizzato in USA.

Il CSC utilizza ancora una volta un modello di machine learning allenato a riconoscere se la velocità e la curva di equalizzazione impostate al momento del trasferimento da analogico a digitale coincidono con quelle scelte dall'artista in origine in porzioni di audio di $500\,\mathrm{ms}$ ed a classificarle con le velocità ed equalizzazione eventualmente corrette; per fare ciò si da in pasto al modello pre-allenato il rumore estrapolato dai silenzi presenti nell'interezza della traccia audio ed i relativi primi 13 coefficienti spettrali mel (*Mel Frequency Cepstral Coefficients* (MFCC)[4]).

[3] L'equalizzazione è una tecnica di filtraggio che permette di variare l'ampiezza delle frequenze di un segnale audio; una curva di equalizzazione è ottenuta dalla combinazione dei risultati delle due curve $N(DB) = 10\log(1 + \frac{1}{4\pi^2 f^2 t_2^2}) - 10\log(1 + 4\pi^2 f^2 t_1^2)$ dove f è la frequenza in Hz e t_1 e t_2 sono costanti di tempo in secondi.

[4] I coefficienti spettrali mel sono una rappresentazione del segnale acustico, solitamente sono utilizzati per rappresentare le caratteristiche generali del segnale acustico e vengono spesso utilizzati in applicazioni di riconoscimento del parlato e recupero di informazioni musicali; sono calcolati a partire dal segnale audio su cui viene applicata la trasformata di Fourier con cui si calcola la potenza dello spettro e la si riscala con la scala mel (una scala basata sulla percezione dell'altezza del suono), dopodiché vengono applicati il logaritmo e la trasformata discreta del coseno; i coefficienti sono le ampiezze dello spettro risultante.

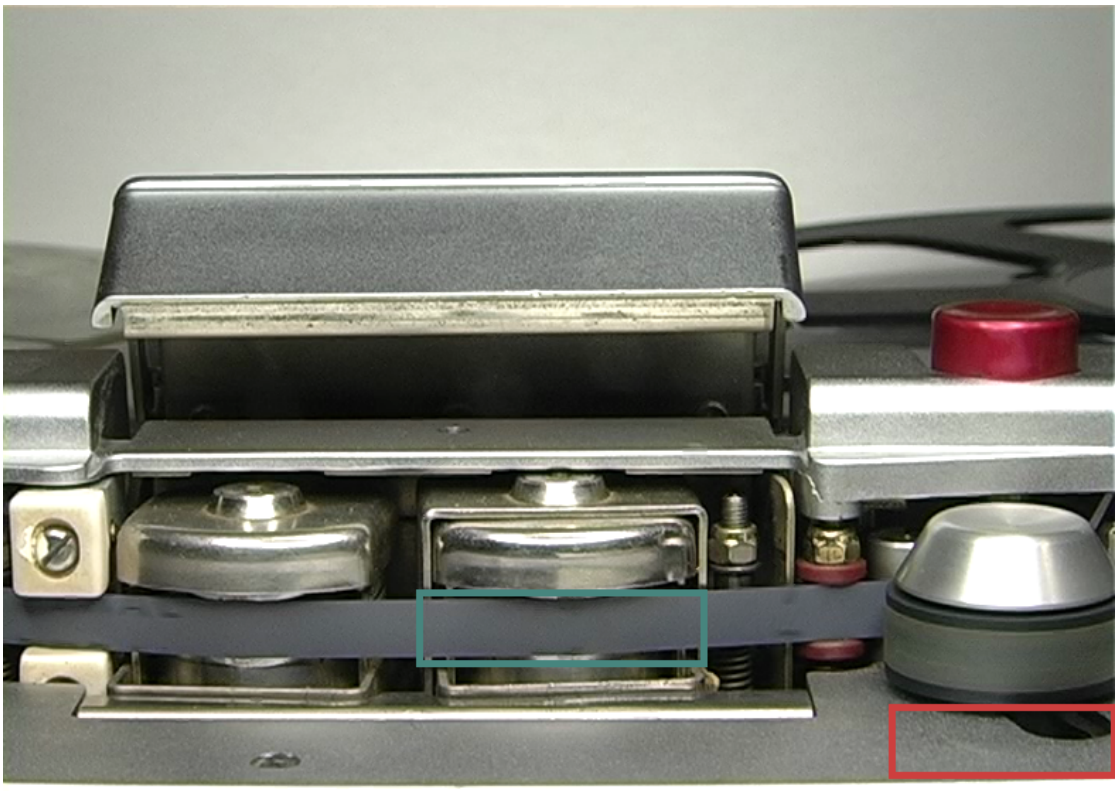
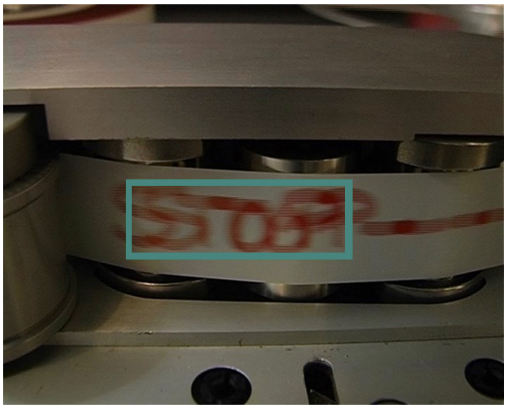


Figura 1.1: Regioni di interesse sotto alla testina di registrazione ed al capstan [19]

(a) Porzione di nastro rovinata [16]



(b) Testo su nastro [1]

Figura 1.2: Alcuni esempi di irregolarità

2

MPAI

Come evoluzione del rinomato Moving Picture Experts Group (MPEG), nel luglio 2020 nasce Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI)[1].

MPAI è un'organizzazione non profit che, ancora guidata da Leonardo Chiariglione, ha come obiettivo la promozione dell'uso efficiente dei dati[2] tramite lo sviluppo di specifiche tecniche per la codifica di qualunque tipo di dato facendo uso dell'intelligenza artificiale e la semplificazione dell'utilizzo di tali codifiche imponendo ai detentori di proprietà intellettuale di stabilire delle licenze per framework (MPAI-AIF), invece di dover essere legati ai brevetti e creare delle *patent pool* [8]. Sostanzialmente l'organizzazione si pone come missione quella di porre ordine nel mondo delle codifiche utilizzanti l'Intelligenza Artificiale (IA) e di farlo semplificando il metodo di accesso alle proprie tecnologie rispetto ad MPEG. MPAI opera attraverso la collaborazione delle varie parti interessate, tra cui l'Università di Padova tramite il suo spin-off Audio Innova.

In questi tempi l'utilizzo dell'intelligenza artificiale sta crescendo in maniera esponenziale e si sta avvicinando all'utente comune tramite la miriade di piattaforme online che sono nate, un esempio è ChatGPT che ha raggiunto quota 100 milioni di utenti attivi in soli 2 mesi, raggiungendo il primato di applicazione

[1] `https://mpai.community`

[2] Per dati MPAI intende, per esempio, dati mediatici, manifatturieri, automobilistici, sanitari e generici. [8].

ad uso privato con la crescita più veloce della storia.[3] Nonostante il suo, come visto, uso spropositato l'IA è di fatto una tecnologia difficilmente comprensibile dalle masse; ciò porta l'utente a non capire che il *chatbot* con cui sta interagendo non replichi utilizzando un principio di causalità, ma piuttosto segua dei pattern linguistici i quali non sempre portano a risposte corrette, nonostante l'autorevolezza col quale l'IA sembri scrivere.

Il comitato di MPAI si impegna ad affrontare col coinvolgimento di esperti esterni gli emergenti quesiti etici, i quali sono molto rilevanti a causa del rapido e soltanto recente sviluppo dell'IA.

Il progetto comprende diverse aree d'effetto tra cui il dialogo uomo-macchina, l'esperienza audio, la compressione video, l'esperienza di gioco online, la creazione di esperienze collaborative nel metaverso, la codifica di dati sanitari, i veicoli a guida autonoma e molti altri e la lista è in continua espansione.[4]

2.1 MPAI-AIF, AIW e AIMs

Ogni standard MPAI è un AI framework (MPAI-AIF) [11]: un ambiente che comprende diversi AI workflow (AIW), ognuno che descrive un certo caso d'uso. I blocchi costituenti un workflow sono detti AI modules (AIMs) ed ogni modulo è definito dalla sintassi e dalla semantica delle proprie interfacce di input e output, l'implementazione (hardware o software che sia) non è specificata; i vari moduli svolgono delle specifiche attività e sono interconnessi a formare un AIW come si può vedere dalla figura 2.1.

MPAI-AIF, il modello fondante gli altri standard dell'organizzazione è stato adottato dall'Institute of Electrical and Electronic Engineers (IEEE) col nome *IEEE 3301-2022* [5].

2.2 Struttura di uno standard MPAI

Lo sviluppo di uno standard MPAI segue le fasi mostrate in figura 2.2.

Uno standard MPAI è composto da un insieme di 4 documenti con i relativi software e dataset [14]:

[3] Fonte: `https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/`

[4] Tutti i vari progetti sono visualizzabili sul sito di MPAI alla voce *Standards*.

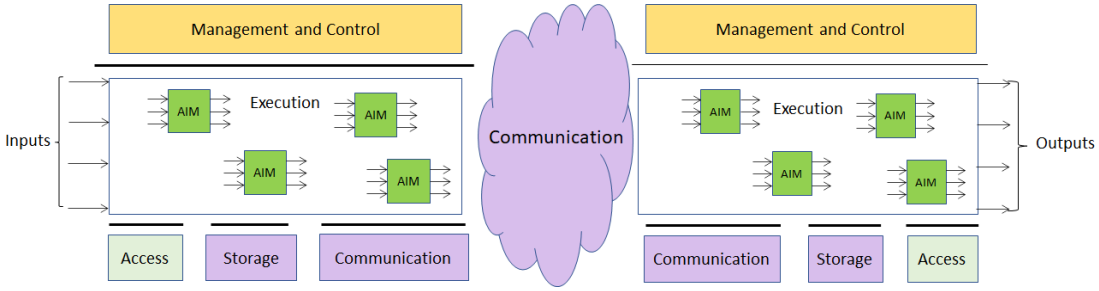


Figura 2.1: Architettura di MPAI-AIF [7]

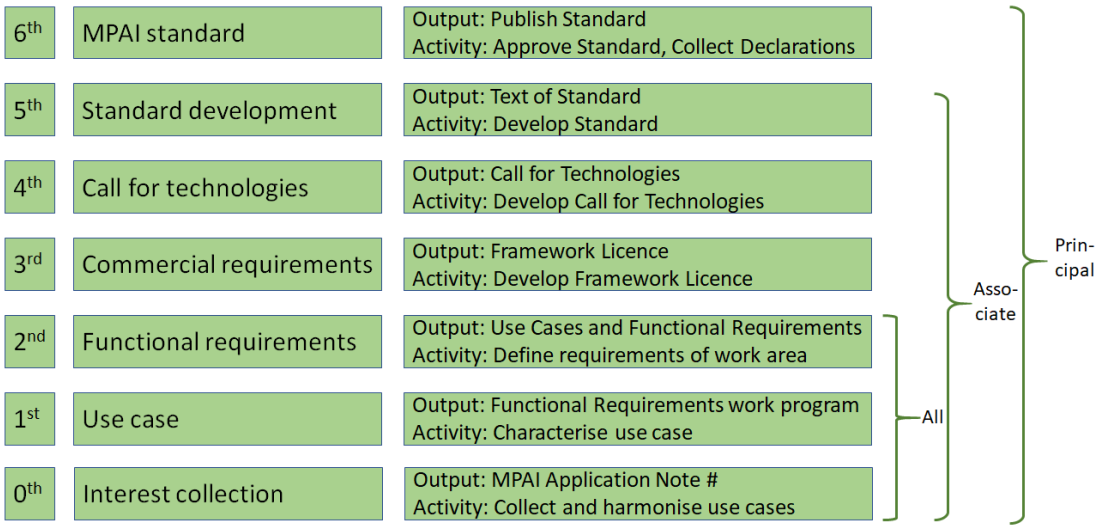


Figura 2.2: Le fasi dello sviluppo di uno standard MPAI [7]

Specifiche tecniche (*Technical Specification*) Contiene le norme che un'implementazione conforme deve necessariamente seguire; solitamente è un insieme di casi d'uso. Per ogni caso d'uso viene specificato l'AIW che lo implementa con le funzioni che esegue, la sintassi e la semantica dei suoi dati in input ed output; la topologia degli AIMs costituenti l'AIW e, per ogni AIM, la loro funzione e la sintassi e la semantica dei loro input ed output.

Software di riferimento (*Reference Software*) Contiene il codice sorgente dell'implementazione delle specifiche tecniche dell'MPAI-AIF e dei suoi AIW esponendo le interfacce dei propri AIMs. Inoltre il software deve essere fornito di un metodo per l'uso, la sua documentazione necessaria[5] ed

[5]Una Knowledge Base (KB)

eventualmente dei dati di esempio.

Test di conformità (*Conformance Testing*) Un insieme di vincoli relativi all'output generato da un dato input a cui un'implementazione deve sottostare per essere definita conforme. Questo documento viene trattato in maniera più approfondita al capitolo 3.

Valutazione delle prestazioni (*Performance Assessment*) Definisce gli attributi di Affidabilità (rispetto dello standard), Robustezza (capacità di gestione di nuovi dati), Equità (IA unbiased[6]) e Replicabilità (della valutazione) che vengono utilizzati per attribuire un voto all'implementazione (eventualmente dipendente da un certo dominio di applicazione).

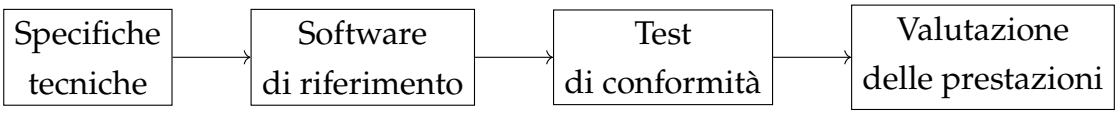


Figura 2.3: Documenti generati nel processo di creazione di uno standard MPAI

2.3 MPAI-CAE

Tra i vari standard/framework, Context-based Audio Enhancement (MPAI-CAE) si occupa di utilizzare le informazioni sul tipo di esperienza audio vissuta dall'utente (intrattenimento, teleconferenza, restauro, ...) e l'informazione del contesto in cui si trova (a casa, in auto, in mobilità, in studio, ...) per agire sul contenuto dell'audio in input e fornire i risultati desiderati [9].

Sono considerati 4 casi d'uso:

Emotion Enhanced Speech (EES) Permette all'utente di scegliere un'emozione ed ottenere successivamente una traccia audio di parlato con la tonalità specifica tipica dell'espressività indicata.

Audio Recording Preservation (ARP) Permette di creare copie di audio digitalizzato, valido per una conservazione a lungo termine e per una riproduzione corretta della registrazione.

[6]Un'intelligenza artificiale può essere *biased*, ovvero può "avere pregiudizi"; nel caso ottimo sono molto limitati (IA unbiased) perchè portano ad assunzioni errate.

Speech Restoration System (SRS) Permette di ripristinare un segmento danneggiato di traccia audio contenente il parlato di un singolo oratore sintetizzando la voce della parte corrotta.

Enhanced Audioconference Experience (EAE) Permette di migliorare la qualità sonora in un'audioconferenza utilizzando i segnali registrati da array di microfoni rimuovendo rumori di fondo e artefatti acustici.

MPAI-CAE è stato adottato dall'IEEE come standard *IEEE 3302-2022* [6].

Nel documento *IEEE Standard Adoption of Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Technical Specification Context-based Audio Enhanced (CAE) Version 1.4* si possono trovare tutte le informazioni sullo standard.

2.3.1 MPAI-CAE-ARP

La procedura di digitalizzazione del CSC introdotta nella sezione 1.3 è stata proposta ad MPAI ed è stata riconosciuta per la sua efficacia ed affidabilità, perciò è stata adottata come use case di MPAI-CAE col nome Audio Recording Preservation (ARP).[7]

Il CSC, con l'aiuto di vari collaboratori, ha prodotto anche un software di riferimento per ARP, il quale non è altro che una codifica[8] audio lossless.

Nel 2023 Audio Innova è stata insignita del *Cannes Neurons Award 2023 Palm d'Or* per il miglior progetto di utilizzo creativo di IA, ARP, al World AI Cannes festival.[9]

Dati in input (vedi figura 2.4):

Preservation Audio File La copia digitalizzata dell'audio.

Preservation Audio-Visual File Il file video prodotto dalla ripresa della testina di registrazione e del capstan (come in figura 1.1).

Si ottengono in output (vedi figura 2.4):

[7]Le informazioni di questa sottosezione sono tratte da [13]; maggiori informazioni nelle specifiche tecniche [6] e nella video presentazione del software di riferimento [15]

[8]Un codec (audio) è un software o un dispositivo che codifica o decodifica un segnale o uno stream di dati secondo una specifica convenzione.

[9]Fonte: `https://mpai.community/2023/02/17/mpai-member-audio-innova-srl-received-the-cannes-neurons-award-2023-palm-dor/`

Access Copy Files Il file audio restaurato, una lista delle modifiche effettuate, la lista delle irregolarità con relativa classificazione e le loro istantanee dal video.

Preservation Master Files Il file audio di input, il file video con l'audio sostituito da quello registrato e sincronizzato, la lista delle irregolarità con relativa classificazione e le loro istantanee dal video.

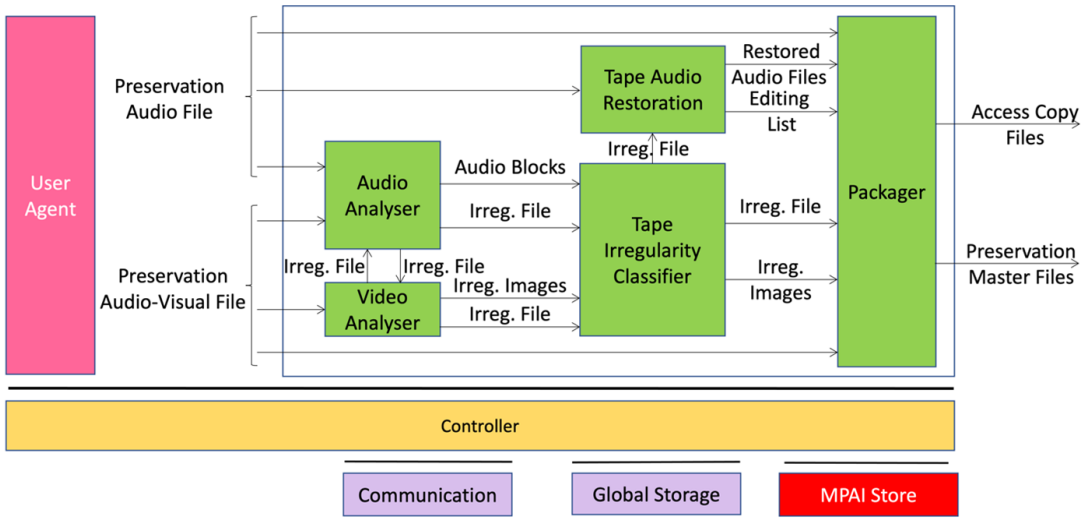


Figura 2.4: AIW di Audio Recording Preservation

Facendo riferimento alla figura 2.4 gli AIMs di ARP sono:

Audio Analyser È l'AIM che rileva le irregolarità nell'audio, estrae i segmenti di $500\,\mathrm{ms}$ in loro corrispondenza e li invia al classificatore.

Video Analyser È l'AIM che rileva le irregolarità nel video e cattura delle immagini in loro corrispondenza.

Tape Irregularity Classifier È l'AIM che classifica le irregolarità di audio e video a partire dalle irregolarità rilevate da Audio Analyser e Video Analyser.

Tape Audio Restoration È l'AIM che corregge velocità, equalizzazione e registrazione a rovescio dell'audio.

Packager È l'AIM che produce Access Copy Files e Preservation Master Files a partire dai file ricevuti dall'output degli altri AIMs.

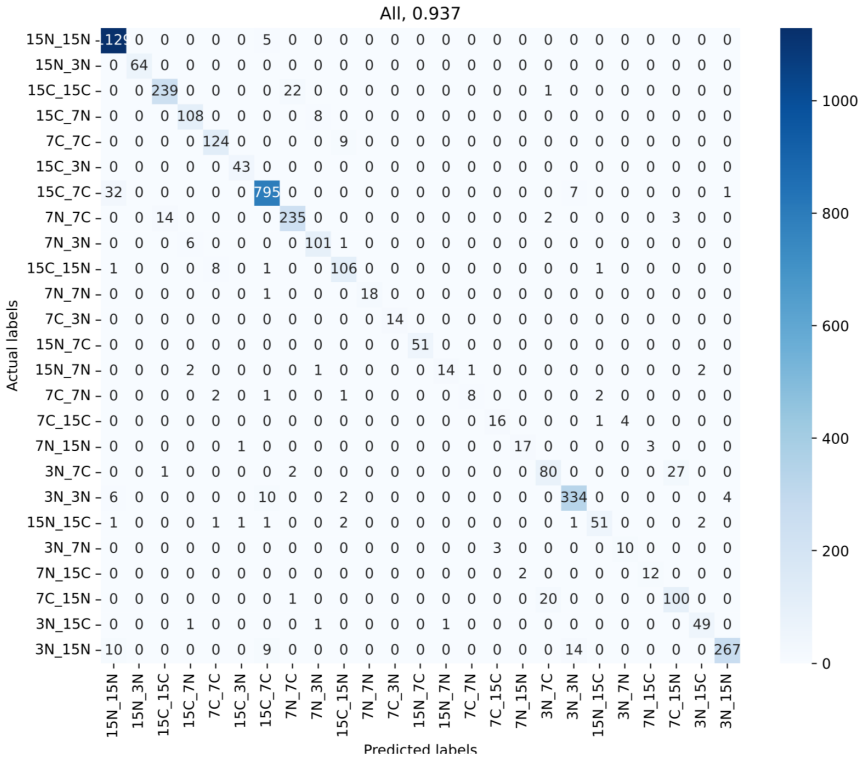
Si osserva nella letteratura afferente al CSC che la classificazione dei problemi dalla traccia audio ha un'accuratezza del $93,7\%$ (esempio in figura 2.5a) [15, min. 35:10], mentre per la classificazione dalle immagini si raggiungono valori $\geq 98,9\%$ (esempio in figura 2.5a) [16, fig. 3 e p. 70].

Le tipologie di irregolarità previste e riconosciute dallo standard sono riassunte nella tabella 2.1.

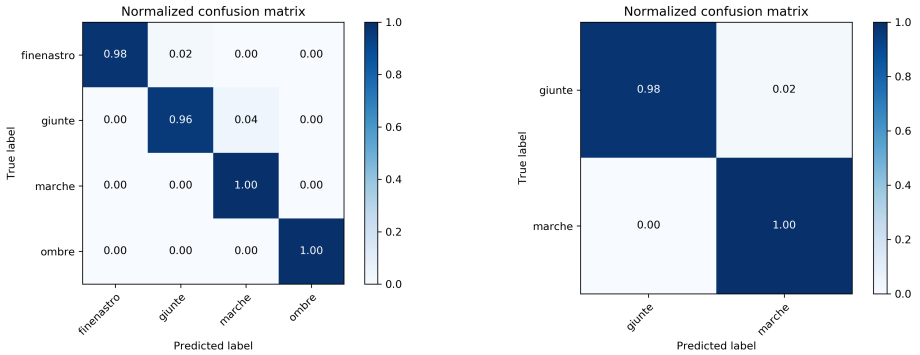
Acronym	**Meaning**
B	Brands on tape
DA	Damaged tape
DI	Dirt
EOT	Ends of tape
ESV	Equalization standard variation
M	Marks
PPS	Play, pause, stop
PSD	Power spectral density
RMSE	Root Mean Square Error
S	Shadows
SB	Signal Backward
SOT	Start of tape
SP	Splice
SSV	Speed standard variation
WF	Wow and flutter

Tabella 2.1: Irregolarità rilevabili da ARP [6, tab. 21-22]

L'albero delle cartelle previsto è riportato in figura 2.6.



(a) Matrice di confusione di classificatore audio [20, fig. 3.8]



(b) Matrice di confusione di classificatore video, esperimento a velocità 7.5 ips

(c) Matrice di confusione di classificatore video, esperimento a velocità 15 ips

(d) Matrici di confusione di classificatore video [4, fig. 4.11]

Figura 2.5: Matrici di confusione[10]da esperimenti in letteratura con classificatori per ARP.

[10]Una matrice di confusione, *confusion matrix* è una rappresentazione visuale dell'accuratezza di un classificatore, ogni colonna rappresenta i valori predetti ed ogni riga i valori reali, in corrispondenza di ogni intersezione si trova il valore assoluto o la percentuale di volte in cui si è verificata tale intersezione.

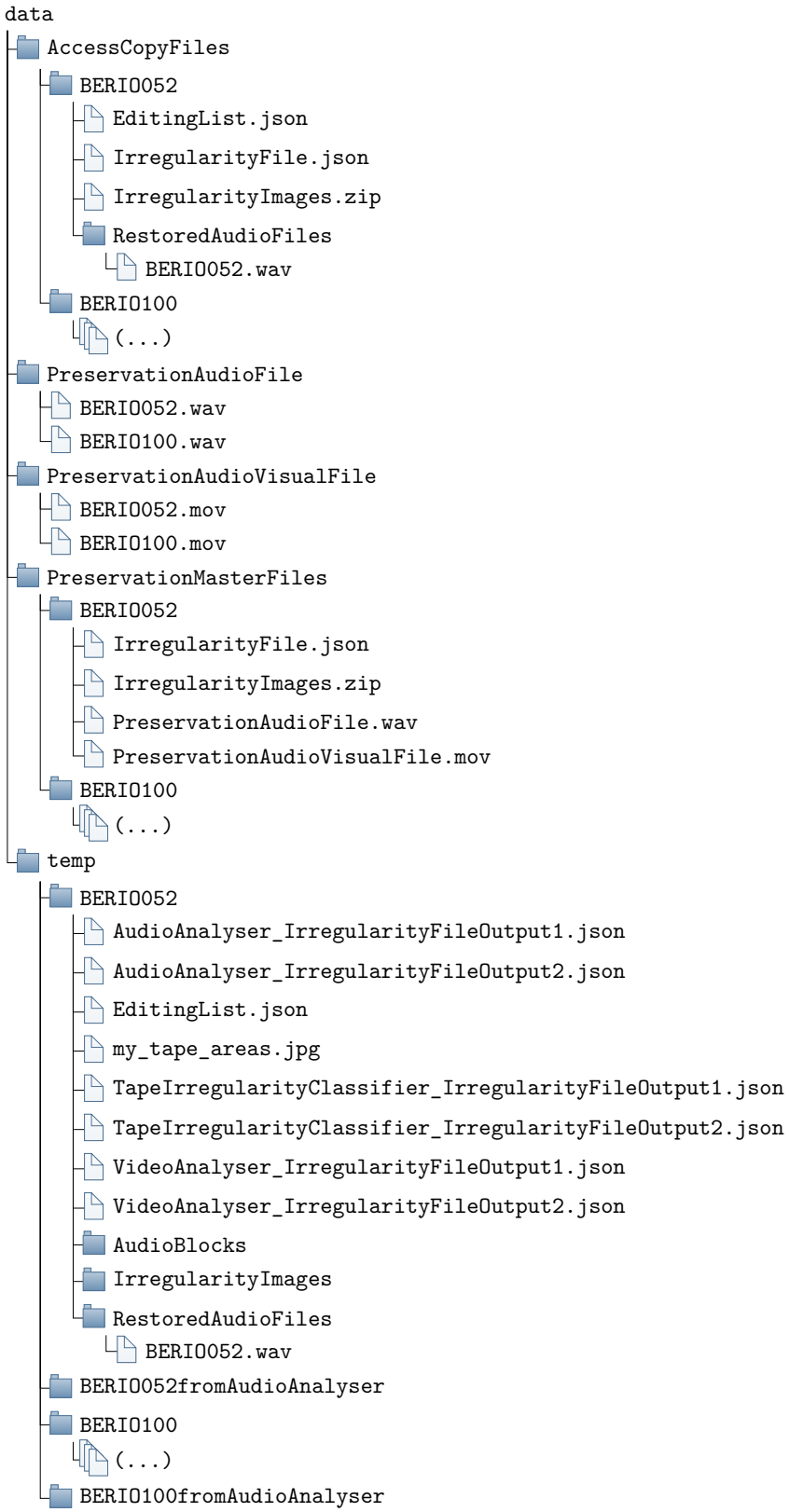


Figura 2.6: Albero delle cartelle previsto da ARP

3

Conformance Testing

I test di conformità, come anticipato nella sezione 2.2, sono un insieme di attività di verifica che determinano l'aderenza di un processo, prodotto o servizio a dei requisiti tecnici o a delle norme. Nel caso in esame, ARP, si tratta di verificare che il suo funzionamento segua, entro certi limiti (necessari anche perchè è una IA), le specifiche tecniche. In MPAI, questi test, sono definiti all'interno di un documento apposito.

L'obiettivo di questa tesi è quella di scrivere il codice dei test di conformità, basandosi sul documento che li descrive fornito da MPAI.

L'ambiente di sviluppo è un ambiente virtuale di Poetry basato su Python 3.10 (poi aggiornato a 3.11)[1], su una macchina Windows 11 (per alcuni confronti, occasionalmente è stata utilizzata una macchina virtuale con Ubuntu 20.04.6 LTS). L'ambiente di produzione è un server Docker con container Linux.[2] Per il versionamento del software viene usato git, utilizzando come server l'installazione GitLab del DEI.[3]

[1]Vedi sezione 3.5.4.

[2]Per questo motivo l'implementazione del CSC ha anche una modalità di esecuzione come server con protocollo gRPC, utilizzato per la comunicazione con i vari AIMs.

[3]Le repositories rilvanti ai fini di questo documento sono: AIM Packager; AIM Audio Analyser; libreria mpai-cae-arp.

3.1 Tests e Test Driven Development

Il software testing è il processo di valutazione e verifica del corretto funzionamento di un prodotto software rispetto alle aspettative; la creazione di test suites ha l'obiettivo di rilevare bug prima di rilasciare il prodotto. Solitamente si tende ad automatizzare i test attraverso alcuni framework in modo tale da poterli eseguire ad ogni modifica del codice.

Il *Test Driven Development* (TDD) è un approccio allo sviluppo di software che prevede la scrittura dei test prima di quella del codice ai quali deve essere sottoposto; inoltre i test vanno ripetuti parallelamente allo sviluppo del software. I test di conformità sono il documento ed i test stessi che l'implementazione software delle specifiche tecniche dovranno rispettare, quindi la loro scrittura e comunque una correzione del codice basandosi su di essi può essere riconosciuta come un approccio di TDD.

3.1.1 Pytest

Uno dei framework per il testing in Python più popolari è pytest, esso permette di ottenere informazioni dettagliate sul fallimento degli `assert` statements[4], di avere fixtures[5] modulari e di essere compatibile con numerosi plugin esterni.

pytest-json-report è un plugin che è stato utilizzato per creare i report richiesti come output del conformance testing in formato JSON.

pytest-xdist è un plugin che è stato utilizzato per parallelizzare l'esecuzione (vedi sezione 3.4 per il suo utilizzo).

Sono state utilizzate delle fixture per definire l'ambiente di test e per ottenere delle cartelle di test (`pytest_sessionstart`, `pytest_sessionfinish` e `tmp_path`), inoltre è stata parametrizzata l'esecuzione delle varie funzioni di test in modo tale da essere eseguite per ogni documento digitalizzato tramite il decoratore `@pytest.mark.parametrize`.

[4] `assert` è la parola chiave che permette di effettuare i test, nello specifico il test procede se il suo parametro è `True`, mentre viene lanciato `AssertionError` ed il test fallisce se il suo parametro è `False`.

[5] Una *fixture* è un elemento del software testing che viene utilizzato per definire un contesto per l'esecuzione di uno (o più) test.

3.2 MPAI-CAE-ARP Packager

Il primo AIM preso in esame è stato il Packager.

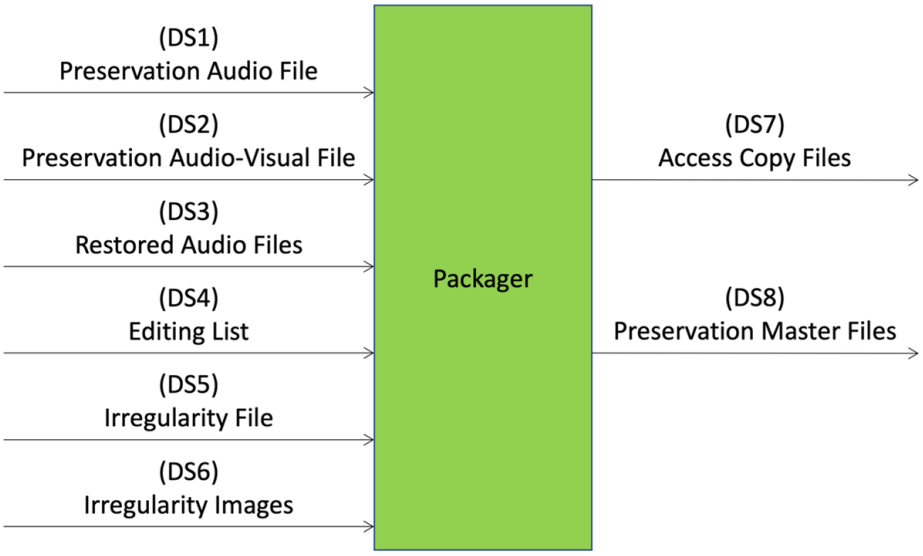


Figura 3.1: ARP Packager

Il Packager, come anticipato nella sottosezione 2.3.1, è l'ultimo AIM ad essere eseguito e si occupa di raccogliere tutti i file elaborati e di restituire in uscita all'AIW una cartella con la copia d'accesso dei file ed una con i file grezzi accompagnati da tutte le irregolarità trovate, vedi schema in figura 3.1.

Viene riportata la descrizione definita da MPAI dei conformance tests da seguire nella tabella 3.1.[6]

[6]Estratta dalla versione WD 0.15.2 del documento MPAI Conformance Testing per MPAI-CAE.

Means	**Actions**
Conformance Testing Dataset	DS1: n Preservation Audio Files. DS2: n Preservation Audio-Visual Files related to DS1. DS3: n Restored Audio Files arrays related to DS1 coming from Tape Audio Restoration. DS4: n Editing Lists related to DS3 coming from Tape Audio Restoration. DS5: n Irregularity Files related to DS1 coming from Tape Irregularity Classifier. DS6: n Irregularity Images related to DS5 coming from Tape Irregularity Classifier. DS7: n Access Copy Files. DS8: n Preservation Master Files.
Procedure	1. Feed Packager under test with DS1, DS2, DS3, DS4, DS5 and DS6. 2. Compare the output Access Copy Files with DS7. 3. Compare the output Preservation Master Files with DS8.
Evaluation	For a given input tuple, verify that: 1. The output Access Copy Files contain the Restored Audio Files, the Editing List, the Irregularity File and the set of Irregularity Images in a .zip file, and is therefore equal to DS7. 2. The output Preservation Master Files contain the Preservation Audio File, the Preservation Audio-Visual File with the audio of the Preservation Audio File, the Irregularity File and the Irregularity Images, and is therefore equal to DS8. An error on any of the output arrays will make the Packager under test not conformant.

Tabella 3.1: Test di conformità per ARP Packager

Ci si aspetta che tutti i dataset utilizzati per eseguire i test seguano la struttura dell'albero delle cartelle come nell'esempio in figura 2.6.

Ad ogni esecuzione dei test è richiesto di inserire i dati nel report mostrato in tabella 3.2.

<table>
<tr><td>Conformance Tester ID</td><td>Unique Conformance Tester Identifier assigned by MPAI</td></tr>
<tr><td>Standard, Use Case ID and Version</td><td>Standard ID and Use Case ID, Version and Profile of the standard in the form “CAE:ARP:1:0”.</td></tr>
<tr><td>Name of AIM</td><td>Packager</td></tr>
<tr><td>Implementer ID</td><td>Unique Implementer Identifier assigned by MPAI Store.</td></tr>
<tr><td>AIM Implementation Version</td><td>Unique Implementation Identifier assigned by Implementer.</td></tr>
<tr><td>Neural Network Version (optional)</td><td>Unique Neural Network Identifier assigned by Implementer.</td></tr>
<tr><td>Identifier of Conformance Testing Dataset</td><td>Unique Dataset Identifier assigned by MPAI Store.</td></tr>
<tr><td>Test ID</td><td>Unique Test Identifier assigned by Conformance Tester.</td></tr>
<tr><td>Actual output</td><td>Actual output provided as a matrix of n rows containing output assertions.
<table>
<tr><th>Output</th><th>Files</th><th>1</th><th>…</th><th>n</th></tr>
<tr><td rowspan="4">Access Copy Files</td><td>Restored Audio Files</td><td>T/F</td><td>…</td><td>T/F</td></tr>
<tr><td>Editing List</td><td>T/F</td><td>…</td><td>T/F</td></tr>
<tr><td>Irregularity File</td><td>T/F</td><td>…</td><td>T/F</td></tr>
<tr><td>Irregularity Images</td><td>T/F</td><td>…</td><td>T/F</td></tr>
<tr><td rowspan="4">Preservation Master Files</td><td>Preservation Audio File</td><td>T/F</td><td>…</td><td>T/F</td></tr>
<tr><td>Preservation Audio Visual File</td><td>T/F</td><td>…</td><td>T/F</td></tr>
<tr><td>Irregularity File</td><td>T/F</td><td>…</td><td>T/F</td></tr>
<tr><td>Irregularity Images</td><td>T/F</td><td>…</td><td>T/F</td></tr>
</table>
Final assertion: T/F</td></tr>
<tr><td>Execution time (optional)</td><td>Duration of test execution</td></tr>
<tr><td>Test comment (optional)</td><td>Comments on test results and possible needed actions.</td></tr>
<tr><td>Test Date</td><td>yyyy/mm/dd.</td></tr>
</table>

Tabella 3.2: Report da compilare ad ogni esecuzione del packager

Viene creato automaticamente il report sotto forma di file JSON contenente le informazioni conosciute dal software anche grazie all’aiuto della libreria pytest-json-report che raccoglie i risultati dei test in un formato machine-readable.

Il seguente listato ne è un esempio:

```
{
    "conformance_tester_id": "",
    "standard_usecaseid_version": "CAE:ARP:1.0",
    "name_of_aim": "Packager",
    "implementer_id": "",
    "neural_network_version": "",
    "identifier_of_conformance_testing_dataset": "",
    "test_id": "",
    "actual_output": {
        "BERI0052": {
            "access_copy_files": {
                "editing_list": true,
                "restored_audio_files": true,
                "irregularity_file": true,
                "irregularity_images": true
            },
            "preservation_master_files": {
                "preservation_audio_file": true,
                "irregularity_file": true,
                "irregularity_images": true,
                "preservation_audio_visual_file": true
            }
        },
        "BERI0100": {
            "access_copy_files": {
                "irregularity_file": true,
                "editing_list": true,
                "irregularity_images": true,
                "restored_audio_files": true
            },
            "preservation_master_files": {
                "preservation_audio_file": true,
                "irregularity_file": true,
                "irregularity_images": true,
                "preservation_audio_visual_file": true
            }
        },
        "final_assertion": true
    },
    "execution_time": 351.11644291877747,
    "test_comment": "",
    "test_date": "2023/09/16"
}
```


Codice 3.1: Esempio di report compilato sotto forma di file JSON, AIM Packager

3.2.1 Bugs ed altri problemi pre-esistenti

Prima di dedicarsi alla scrittura dei test, si è verificata la corretta funzionalità del software.

Sono stati riscontrati dei problemi importanti relativi alla creazione del Preservation Audio-Visual File:

```
1  # Create Preservation Audio-Visual File with substituted audio
2  video_file = files_name + '.mov'
3  pvf_path = os.path.join(working_path, 'PreservationAudioVisualFile/',
       video_file)
4  try:
5      audio = AudioFileClip(paf_path)
6      video = VideoFileClip(pvf_path)
7      # Open Irregularity File to get offset
8      irregularity_file_json = open(
9          os.path.join(temp_path, '
       TapeIrregularityClassifier_IrregularityFileOutput2.json')
10     )
11     irregularity_file = json.load(irregularity_file_json)
12     offset = irregularity_file['Offset']/1000
13     if offset > 0:
14         audio = audio.subclip(t_start=offset)
15     else:
16         video = video.subclip(t_start=offset)
17     video = video.set_audio(audio)
18     video.write_videofile(pmf_path + 'PreservationAudioVisualFile.mov
       ', bitrate='3000k', codec='mpeg4')
19     print("Preservation Audio-Visual File created")
20 except OSError:
21     pprint(f"Preservation Audio-Visual File file '{pvf_path}' not
       found!", color=Color.RED)
22     quit(os.EX_NOINPUT)
```

Codice 3.2: Codice iniziale, creazione Preservation Audio-Visual File (Packager)

Il primo problema è relativo alla sincronizzazione del video con l'audio: l'audio deve essere anticipato se viene trovato uno scostamento A/V positivo, altrimenti deve essere anticipato il video. Alle righe 12-16 si osserva che il comportamento non è quello richiesto: dato che a riga 16 l'offset ha valore negativo, allora MoviePy, libreria utilizzata per eseguire editing video, farà iniziare la traccia a partire da $|offset|$ secondi prima del termine della clip.[7]

Il secondo problema è relativo alla non specifica nel codice della codifica audio che porta la libreria MoviePy a ricadere nel comportamento di default ovvero generare un file con codifica MP3 (lossy) a $44\,100$ Hz e ad avere di conseguenza,

[7]Codice sorgente: `https://zulko.github.io/moviepy/_modules/moviepy/Clip.html#Clip.subclip`

in questo caso, una transcodifica, il che non è ottimale per un software che ha come obiettivo la conservazione di documenti audio in alta qualità.

Il terzo problema invece è provocato dalla libreria MoviePy che, per motivi sconosciuti e solo in alcuni casi, genera file corrotti nella traccia video, nello specifico che si bloccano dopo alcuni secondi.

Per risolvere questi due problemi si è scelto di sostituire MoviePy direttamente con FFmpeg, ottenendo come risultato il seguente codice:

```
# Create Preservation Audio-Visual File with substituted audio
video_file = files_name + '.mov'
pvf_path = os.path.join(working_path, 'PreservationAudioVisualFile/', video_file)
try:
    # Open Irregularity File to get offset
    irregularity_file_json = open(
        os.path.join(temp_path, 'TapeIrregularityClassifier_IrregularityFileOutput2.json')
    )
    irregularity_file = json.load(irregularity_file_json)
    offset = irregularity_file['Offset']
    command_to_run = ['ffmpeg',
                      '-y', '-hide_banner', '-loglevel', 'error']
    # If offset is positive, the audio is anticipated, otherwise video is anticipated (through seek)
    if offset > 0:
        command_to_run = command_to_run + ['-i', pvf_path,
                                           '-ss', str(offset)+'ms', '-i', paf_path]
    else:
        command_to_run = command_to_run + ['-ss', str(offset * -1)+'ms', '-i', pvf_path,
                                           '-i', paf_path]
    command_to_run = command_to_run + ['-c:v', 'mpeg4', '-c:a', 'copy',
                                       '-map', '0:v', '-map', '1:a',
                                       '-b:v', '3M', '-maxrate', '4M', '-bufsize', '4M',
                                       pmf_path + 'PreservationAudioVisualFile.mov']
    subprocess.run(command_to_run)
    print("Preservation Audio-Visual File created")
except OSError:
```

```
27      pprint(f"Preservation Audio-Visual File file '{pvf_path}' not
        found!", color=Color.RED)
28      quit(os.EX_NOINPUT)
```

Codice 3.3: Codice finale, creazione Preservation Audio-Visual File (Packager)

3.2.2 Come verificare l'uguaglianza tra tracce audio

Per verificare l'uguaglianza fra tracce audio, inizialmente si è provato un confronto tramite le librerie *hashlib* o *filecmp*, ma entrambi i test non hanno avuto successo perché in alcuni casi non è richiesto di verificare che il file sia lo stesso identico (stesso hash), ma piuttosto di controllare che il suono sia il medesimo.[8]

La soluzione trovata si basa sulla libreria chromaprint[9], la quale permette di ricavare un'impronta digitale (*fingerprint*) a partire da una traccia audio in maniera da poi confrontare due impronte tra loro per ottenere una misura della somiglianza tra le rispettive tracce.

È stata utilizzata in particolare una libreria Python, la quale non è altro che un wrapper di AcoustID, chiamata pyacoustid che aiuta lo sviluppatore Python esponendo direttamente le API di chromaprint ed un metodo per confrontare le impronte.

```
1 AUDIO_THRESHOLD = 0.7   # (ratio)
2 # [...]
3 input_fingerprint = acoustid.fingerprint_file(input_paf_path)
4 output_fingerprint = acoustid.fingerprint_file(output_paf_path)
5 assert acoustid.compare_fingerprints(input_fingerprint,
      output_fingerprint) > AUDIO_THRESHOLD, "PreservationAudioFile.wav
      is not the same as input"
```

Codice 3.4: Test di comparazione di due file audio tramite la loro impronta digitale

Come si può leggere dal codice, è stata scelta come soglia di somiglianza il 70%, non il 100% perché, ad esempio, nei casi in cui è stata effettuata la sincro-

[8] La differenza si osserva, per esempio, quando si hanno 2 file, uno scostato di qualche secondo rispetto all'altro; questo è ciò che accade nel caso in esame quando si esegue la sincronizzazione audio/video.

[9] chromaprint è la libreria che è alla base del progetto AcoustID, utilizzato nel piuttosto famoso software MusicBrainz (Picard) che serve a taggare le proprie tracce musicali.

nizzazione audio/video l'audio non è esattamente uguale; inoltre dai vari test eseguiti sui dataset forniti, questo valore ha funzionato correttamente.

Durante la scelta della soglia da adottare sono state eseguite diverse prove ed è stato verificato che, correttamente, anche nel caso dei dataset forniti, si ottengono dei risultati elevati, talvolta uguali a 1, se i file dovrebbero essere considerati uguali, mentre in caso di file diversi si ottiene un valore tendente a 0.[10]

La libreria pyacoustid, nelle prime fasi di scrittura dei test, non era ancora completamente funzionante né compresa del metodo per comparare le impronte digitali, infatti la sua versione 1.2.2, presente nella repository PyPi, non vedeva ancora implementata la funzione `compare_fingerprints`, già presente invece nella sua repository GitHub grazie al contributo di un utente; tale implementazione era, però, non funzionante a causa di alcuni errori del codice come si può vedere dai cambiamenti applicati successivamente e descritti qui di seguito:

```
@@ -382,7 +382,7 @@ def _match_fingerprints(a: List[int], b: List[int
    ]) -> float:
                if biterror <= MAX_BIT_ERROR:
                    offset = i - j + bsize
                    counts[offset] += 1
-    topcount = counts.max()
+    topcount = max(counts)
     return topcount / min(asize, bsize)


@@ -399,6 +399,6 @@ def compare_fingerprints(a, b) -> float:
     # decompress fingerprints
-    a = [int(x) for x in chromaprint.decode_fingerprint(a)[0]]
-    b = [int(x) for x in chromaprint.decode_fingerprint(b)[0]]
+    a = [int(x) for x in chromaprint.decode_fingerprint(a[1])[0]]
+    b = [int(x) for x in chromaprint.decode_fingerprint(b[1])[0]]
     return _match_fingerprints(a, b)
```

Codice 3.5: Correzioni applicate in pyacoustid nel codice del confronto tra impronte

Per applicare queste modifiche è stato necessario effettuare una Pull Request (PR)[11] sulla repository ufficiale a cui ha seguito uno scambio di messaggi con il manutentore e l'implementatore del codice;[12] questa è una pratica comune nel

[10] Alcune prove si possono trovare alla pagina: `https://github.com/albertopasqualetto/Tesi-triennale/blob/main/Notebooks/chromaprint_comparisons.ipynb`

[11] Una PR è una richiesta ai manutentori del codice di aggiungerne di proprio per implementare nuove funzioni o per correggere dei problemi.

[12] URL della PR: `https://github.com/beetbox/pyacoustid/pull/78`

mondo del software libero, il quale, tra i vari vantaggi, permette di leggere il codice sorgente e contribuire con dei miglioramenti alle tecnologie che si utilizzano.[13] Si è contribuito a pyacoustid anche con un'altra PR che permette l'utilizzo della libreria anche in Windows, la quale è stata implementata collaborando col manutentore.[14]

Ora la versione presente su PyPi è la 1.3.0, contenente tutte le modifiche citate.

3.2.3 Come verificare l'uguaglianza tra flussi video

Per verificare l'uguaglianza tra gli stream video è stato utilizzato il peak signal-to-noise ratio (PSNR) medio tra i frame dei 2 video, calcolato tramite FFmpeg.[15]

Il PSNR è una misura di qualità di un'immagine compressa rispetto all'originale; viene definito come il rapporto tra la massima potenza di un segnale (immagine originale) e la potenza del rumore rispetto alla sua rappresentazione (immagine compressa). In questo caso si utilizzano al posto delle immagini originale e compressa, i frame dei 2 video da confrontare rispettivamente.

```
VIDEO_THRESHOLD = 25    # dB
# [...]
psnr_out = subprocess.run(["ffmpeg",    # video
                           "-i", input_pvf_path,
                           "-i", tmp_path / (files_name+"_PreservationAudioVisualFile_output_video.mov"),
                           "-filter_complex", "psnr",
                           "-f", "null",
                           "-"],
                          capture_output=True)
psnr_out = psnr_out.stderr.decode("utf-8")
psnr = psnr_out[psnr_out.find('average:') + 8:psnr_out.find(' ', psnr_out.find('average:'))]
print("psnr_out=", psnr)
assert psnr == 'inf' or float(psnr) > VIDEO_THRESHOLD, "PreservationAudioVisualFile.mov is not the same as input"
```

Codice 3.6: Test di comparazione di due file audio tramite il PSNR

[13] https://opensource.guide/how-to-contribute/#improve-software-you-rely-on

[14] URL della PR: https://github.com/beetbox/pyacoustid/pull/79

[15] https://ffmpeg.org/ffmpeg-all.html#psnr

Come si può osservare dal codice, è stato individuato un PSNR soglia di $25\,\mathrm{dB}$, ancora una volta non è massimo (infinito) perché nei casi di sincronizzazione in cui si taglia una parte di video, si ottiene un valore minore (e nemmeno elevato) di PSNR, ma comunque in grado di discernere video "uguali" da video veramente distinti.

Esistono altri metodi per comparare dei video e tra questi vi sono il Structural similarity index measure (SSIM) oppure, ancora una volta, qualche algoritmo di fingerprinting, ma si è visto che il PSNR è funzionale allo scopo ed è il più rapido ad essere eseguito; il codice potrebbe essere migliorato per avere più certezza dell'uguaglianza, ma questo non è un problema semplice considerando che i video in questione sono identici a meno di un'eventuale taglio della parte iniziale, un'idea potrebbe essere quella di calcolare il PSNR a ritroso e considerando i video solo per la durata del più breve, ciò però non considererebbe il caso limite di file diversi con la parte terminale in comune.

3.2.4 Refactoring del codice della libreria

A termine del lavoro è stato eseguito un refactoring del codice della libreria ricalcando lo stile del TDD. Si è cercato di seguire il principio Don't repeat yourself (DRY), ovvero di limitare la ridondanza del codice creando delle funzioni autoesplicative per ogni macro azione del codice. Facendo ciò si è limitata la ridondanza sia all'interno dello stesso file, sia si è rimossa la duplicazione tra client e server, che ora utilizzano entrambi le funzioni di una libreria condivisa.

Per le funzioni appena citate sono stati creati degli unit tests per le funzionalità non ancora collaudate dai conformance tests.

Ove opportuno sono state scritte le docstings per le funzioni, ovvero la loro documentazione.

Sono state anche aggiunte delle strutture di controllo per assicurare l'esecuzione da capo a fine in Windows, dato che alcune funzionalità non sono supportate dal sistema operativo di Microsoft:

```
1 - quit(os.EX_CONFIG)
2 + if not sys.platform.startswith(('win', 'cygwin')):
3 +     quit(os.EX_CONFIG)  # `os.EX_CONFIG` is not compatible with the
      above platforms in Python 3.10
4 + else:
5 +     quit()
```

Codice 3.7: Modifiche per permettere la compatibilità di Packager con Windows

Ciò è stato fatto nell'ottica di garantire l'esecuzione multipiattaforma, nonostante in produzione il Packager sia eseguito in un container Linux.

3.3 MPAI-CAE-ARP Audio Analyser

Successivamente è stato preso in considerazione l'AIM Audio Analyser.

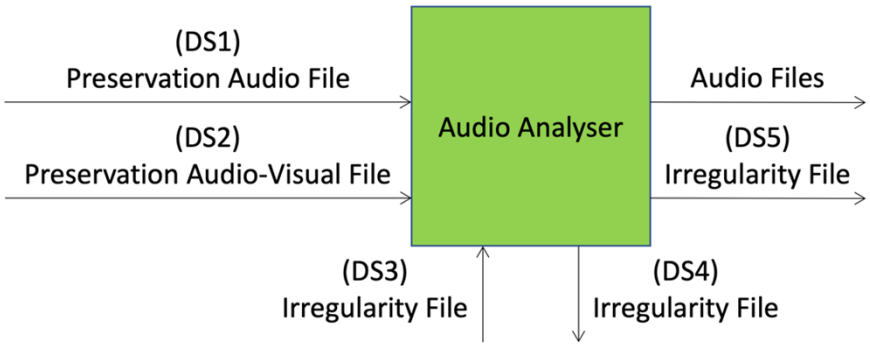


Figura 3.2: ARP Audio Analyser

L'Audio Analyser, come anticipato nella sottosezione 2.3.1, è l'AIM che si occupa di individuare le irregolarità nell'audio, di fornirle al Video Analyser e, una volta ottenute anche le irregolarità del video, di fornirle al Tape Irregularity Classifier (il classificatore delle irregolarità), oltre a ciò si occupa di passare al classificatore appena citato anche dei frammenti di audio lunghi $500\,\mathrm{ms}$ in corrispondenza di ogni irregolarità, vedi schema in figura 3.2.

Viene riportata la descrizione definita da MPAI dei conformance tests da seguire nella tabella 3.3.[16]

[16]Estratta dalla versione WD 0.15.2 del documento MPAI Conformance Testing per MPAI-CAE.

Means	**Actions**
Conformance Testing Dataset	DS1: n* Preservation Audio Files. DS2: n Preservation Audio-Visual Files related to DS1. DS3: n Irregularity Files related to DS2. DS4: n output Irregularity Files in the format of port IrregularityFileOutput_1 with all Irregularities correctly identified. DS5: n output Irregularity Files in the format of port IrregularityFileOutput_2 with the real offset and all Irregularities correctly identified and included from DS3. * A reasonable n for testing is 5<n<=10, since each file generates multiple irregularities to classify
Procedure	1. Feed Audio Analyser under test with DS1, DS2 and DS3. 2. Compare the computed offsets with the ones contained in DS5. 3. Analyse the Irregularity Files resulting from port IrregularityFileOutput_1. 4. Analyse the Irregularity Files resulting from port IrregularityFileOutput_2.
Evaluation	1. Verify the conditions: a. The Irregularity Files are syntactically correct and conforming to the JSON schema provided in CAE Technical Specification. b. All Irregularities from DS3 are included in the Irregularity Files coming from port IrregularityFileOutput_2. c. $\lvert O_c - O_r \rvert < 3 \times \lceil \frac{1000}{FPS_{DS3}} \rceil ms$, where O_c is the offset computed by the Audio Analyser under test, O_r is the real offset and FPS_{DS3} is the number of frames per second at which the DS3 video has been recorded. d. All output Audio Files are conforming to RF64 file format. e. For each of the n tuples of input records, the output Audio Files are extracted from the corresponding input Preservation Audio File at the Time Labels indicated in the Irregularity File coming from port IrregularityFileOutput_2. 2. By inspecting the Irregularity Files resulting from port IrregularityFileOutput_1, for each of the n tuples of input records, compute the values of Recall (R) and Precision (P). 3. Compute the average value of Recall ($\tilde{R}$) and Precision ($\tilde{P}$) measures obtained at point 2. 4. Accept the AIM under test if: a. $\tilde{R} > 0.9$ b. $\tilde{P} > 0.9$

Tabella 3.3: Test di conformità per ARP Audio Analyser

Ci si aspetta che tutti i dataset utilizzati per eseguire i test seguano la struttura dell'albero delle cartelle come nell'esempio in figura 2.6.

Ad ogni esecuzione dei test è richiesto di inserire i dati nel report mostrato in tabella 3.4.

<table>
<tr><td>Conformance Tester ID</td><td>Unique Conformance Tester Identifier assigned by MPAI</td></tr>
<tr><td>Standard, Use Case ID and Version</td><td>Standard ID and Use Case ID, Version and Profile of the standard in the form “CAE:ARP:1:0”.</td></tr>
<tr><td>Name of AIM</td><td>Audio Analyser</td></tr>
<tr><td>Implementer ID</td><td>Unique Implementer Identifier assigned by MPAI Store.</td></tr>
<tr><td>AIM Implementation Version</td><td>Unique Implementation Identifier assigned by Implementer.</td></tr>
<tr><td>Neural Network Version (optional)</td><td>Unique Neural Network Identifier assigned by Implementer.</td></tr>
<tr><td>Identifier of Conformance Testing Dataset</td><td>Unique Dataset Identifier assigned by MPAI Store.</td></tr>
<tr><td>Test ID</td><td>Unique Test Identifier assigned by Conformance Tester.</td></tr>
<tr><td>Actual output</td><td>Actual output provided as a matrix of n rows containing R and P values.
<table>
<tr><td>Tuple #</td><td>R</td><td>P</td></tr>
<tr><td>1</td><td>Measure 1</td><td>Measure 1</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
<tr><td>n</td><td>Measure n</td><td>Measure n</td></tr>
</table></td></tr>
<tr><td>Execution time (optional)</td><td>Duration of test execution.</td></tr>
<tr><td>Test comment (optional)</td><td>-</td></tr>
<tr><td>Test Date</td><td>yyyy/mm/dd.</td></tr>
</table>

Tabella 3.4: Report da compilare ad ogni esecuzione dell'Audio Analyser.

Il report viene compilato automaticamente in un file JSON come spiegato anche per il Packager (sottosezione 3.2.1). Un esempio è riportato di seguito:

```
{
    "conformance_tester_id": "",
    "standard_usecaseid_version": "CAE:ARP:1.0",
    "name_of_aim": "Audio Analyser",
```

```
    "implementer_id": "",
    "neural_network_version": "",
    "identifier_of_conformance_testing_dataset": "",
    "test_id": "",
    "actual_output": {
        "BERI0052": {
            "recall": 1.0,
            "precision": 1.0
        },
        "BERI0100": {
            "recall": 1.0,
            "precision": 1.0
        }
    },
    "execution_time": 418.7547469139099,
    "test_comment": "",
    "test_date": "2023/09/01"
}
```

Codice 3.8: Esempio di report compilato sotto forma di file JSON, AIM Audio Analyser

3.3.1 Problemi pre-esistenti

Alla prima esecuzione del modulo, ancora prima di scrivere i test richiesti, si è notato che l'offset calcolato fosse estremamente diverso da quello fornito nei dataset di esempio, nello specifico si osservava addirittura uno scostamento di segno opposto. Indagando si è riscontrato che il problema si presentasse solo nel caso di utilizzo del codice su piattaforme Windows ed infine si è visto che l'errore fosse dato dal fatto che il tipo di dato di default per NumPy in Windows è `int32` anche in macchine a 64 bit, mentre in ambiente Unix (ancora una volta si far riferimento a problemi non riscontrabili in produzione) il default è `int64`; passando il tipo necessario manualmente, il problema di overflow alla chiamata di `scipy.signal.correlate` non si presenta più.[17] Questa problematica risiede nella libreria `mpai-cae-arp`, la quale viene discussa nella sezione 3.5.

Sono stati sistemati degli errori di battitura, alcuni dei quali impedivano il funzionamento del codice (vedi l'uso ambiguo di Pydantic precedentemente permesso alla sottosezione 3.5.1)

È stata aggiornata la versione richiesta della libreria mpai-cae-arp su cui sono basati tutti gli AIM di ARP:

```
@@ -21,5 +21,5 @@
```

[17] `https://github.com/numpy/numpy/issues/9464`

```
2     pandas = "^2.0.0"
3     scikit-learn = "^1.2.2"
4     grpcio-tools = "^1.53.0"
5 -   mpai-cae-arp = "^0.3.0"
6 +   mpai-cae-arp = "^0.5.0"
7     ffmpeg-python = "^0.2.0"
```

Codice 3.9: Aggiornamento della libreria `mpai-cae-arp` in Audio Analyser, file `pyproject.toml`

Si nota che la specifica `^0.3.0` (basata sul versionamento semantico) permette solo le versioni $\geq 0.3.0, < 0.4.0$, quindi l'aggiornamento fino a prima della successiva MINOR, differendo dal normale funzionamento in presenza di versioni con MAJOR≥ 1 ($\geq 1.0.0$) per cui si permette l'aggiornamento fino a prima della successiva MAJOR.[18]

Infine nel codice dei test si dovrà fronteggiare il fatto che, in maniera incorretta, nel video analyser si utilizzano come divisore tra secondi e millisecondi i due punti (:) (si può vedere nei file `*_IrregularityFileOutputN.json`), quindi, alla lettura di file contenenti irregolarità provenienti dal video analyser, si dovranno applicare delle soluzioni per aggirare il problema ed inoltre si vedranno fallire i test che chiedono di verificare la sintassi dello schema JSON nel caso contengano le irregolarità già citate.

3.3.2 Come verificare che l'offset calcolato sia compatibile con quello reale

Nella tabella di valutazione 3.3 al punto 1.c viene richiesto di controllare che lo scostamento calcolato dall'Audio Analyser sia sufficientemente vicino quello reale, fornito nel dataset 5. Per fare ciò si richiede che la differenza in valore assoluto tra i 2 offset sia minore del tempo necessario a mostrare 3 frame del Preservation Audio-Visual File in millisecondi e si calcola tramite la seguente disuguaglianza:

$$|O_c - O_r| < 3 \times \lceil \frac{1000}{FPS_{DS3}} \rceil ms \tag{3.1}$$

dove O_c è l'offset calcolato, O_r è l'offset reale e FPS_{DS3} sono gli FPS del Preservation Audio-Visual File.

[18] Fonte: documentazione di Poetry.

Gli FPS del video sono ottenuti tramite ffprobe, una utility di FFmpeg votata all'acquisizione di informazioni da file multimediali, tramite la richiesta di mostrare lo stream `r_frame_rate`.[19]

Il codice utilizzato è il seguente:

```
@pytest.mark.parametrize("files_name", files_names)
def test_offset_difference(files_name: str):
    """Conformance testing specification point c.

    Test if the offset difference between Audio Analyser and DS5 real
     offset is small.
    """
    aa_if1_path = temp_path / files_name / "
AudioAnalyser_IrregularityFileOutput1.json"
    test_if2_path = oldify(temp_path / files_name / "
AudioAnalyser_IrregularityFileOutput1.json")
    assert aa_if1_path.is_file(), "
AudioAnalyser_IrregularityFileOutput1.json not found"
    with open(aa_if1_path, "r") as aa_if_file:
        aa_if_dict = json.load(aa_if_file)
    with open(test_if2_path, "r") as test_if_file:
        test_if_dict = json.load(test_if_file)
    aa_offset = int(aa_if_dict["Offset"])
    real_offset = int(test_if_dict["Offset"])

    out = subprocess.run(["ffprobe", "-v", "0", "-of", "csv=p=0", "-
select_streams", "v", "-show_entries", "stream=r_frame_rate", str(
working_path / "PreservationAudioVisualFile" / (files_name+".mov")
)], capture_output=True)
    rate = out.stdout.decode("utf-8").split('/')
    fps = int(rate[0]) / int(rate[1])

    assert abs(aa_offset - real_offset) < 3 * math.ceil(1000 / fps),
f"Offset difference is too big: {abs(aa_offset - real_offset)}ms"
```

Codice 3.10: Codice del test di conformità sulla differenza tra scostamento reale e calcolato, Audio Analyser

[19] Si usa il comando `ffprobe -v 0 -of csv=p=0 -select_streams v -show_entries stream=r_frame_rate video_path.mov`

3.3.3 Come verificare che dei file siano in formato WAV

Nella tabella 3.3 al punto 1.d viene richiesto di controllare che i segmenti audio estratti dall'Audio Analyser siano in formato RF64: un formato compatibile col WAV; in realtà in questo caso è sufficiente verificare che tali segmenti siano in formato WAV.

Per fare ciò è stata utilizzata la libreria Python filetype che legge i magic numbers: delle stringhe costanti nell'header di un file utilizzati per identificarne la tipologia. La firma di un WAV in formato esadecimale è "`52 49 46 46 ?? ?? ?? ?? 57 41 56 45`".[20] La libreria citata cerca di indovinare il tipo del file e ne restituisce il tipo MIME, `audio/x-wav` nel caso del WAV.[21]

3.3.4 Come verificare che la classificazione sia coerente

Per verificare se la rilevazione e la classificazione delle irregolarità sono coerenti con il dataset di test, è stata creata la funzione `check_classification_results` che confronta 2 dizionari contenenti irregolarità e restituisce il numero di elementi diversi e dove si trovano queste diversità:

```
1  def check_classification_results(if_dict_1: dict, if_dict_2: dict,
       print_differences: bool = False) -> (bool, int, int, int):
2      """
3      Check if the irregularity files' classification results in `
       if_dict_1` are exactly equal to `if_dict_2`.
4
5      Parameters
6      ----------
7      if_dict_1 : dict
8          First irregularity file dictionary (first file to compare).
9      if_dict_2 : dict
10         Second irregularity file dictionary (second file to compare).
11     print_differences : bool, optional
12         If True, print the differences between the two files, by
       default False.
13
14     Returns
```

[20] Gli `??` indicano che i valori in quelle posizioni non sono fissi.

[21] *Multipurpose Internet Mail Extensions* (MIME) è uno standard che estende le email e fornisce il supporto a dati diversi dal solo testo. Il suo `Content-Type` indica il tipo di dato.

```
    -------
    tuple[bool, int, int, int]
        Tuple of:
            1) if the two files are exactly equal;
            2) number of elements in common;
            3) number of extra elements in the first file;
            4) number of extra elements in the second file.
    """
    def check_all_combinations(list1: list[dict], idxs1: list[int], list2: list[dict], idxs2: list[int]) -> (int | None, int | None):
        """
        Check if any combination of 2 elements with specified indexes in two different lists is equal and return the indexes of the first equal elements found.
        # [...]
        """
        def filtered_dict(dict_obj: dict, filter: list) -> dict:
            # [...]

        # [...]

    exact_equality = True
    found_n = 0
    extra_in_1 = 0
    extra_in_2 = 0
    irrs_1 = if_dict_1["Irregularities"]
    irrs_1.sort(key=lambda irr: int(irr["TimeLabel"].replace(":", "").replace(".", "")))  # FIXME workaround for ":" timelabel problem from video analyser
    irrs_2 = if_dict_2["Irregularities"]
    irrs_2.sort(key=lambda irr: int(irr["TimeLabel"].replace(":", "").replace(".", "")))

    timelabels_1 = [irr["TimeLabel"] for irr in irrs_2]
    for timelabel in list(dict.fromkeys(timelabels_1)):     # remove duplicates
        irrs_1_idxs = get_irr_idxs_by_timelabel(irrs_1, timelabel)
        irrs_2_idxs = get_irr_idxs_by_timelabel(irrs_2, timelabel)

        while True:
            irr_1_idx, irr_2_idx = check_all_combinations(irrs_1, irrs_1_idxs, irrs_2, irrs_2_idxs)
            if irr_1_idx or irr_2_idx:  # first check exact equality
```

```
50                pass
51            else:  # then the others remained as couples (equal
    length already checked)
52                if not len(irrs_1_idxs) or not len(irrs_2_idxs):
53                    break
54                exact_equality = False
55                irr_1_idx = irrs_1_idxs[0]
56                irr_2_idx = irrs_2_idxs[0]
57                if print_differences:
58                    print("Irregularity", irrs_1[irr_1_idx], "\n\tnot
     exactly equal to\n\t", irrs_2[irr_2_idx])
59            found_n += 1
60            del irrs_1[irr_1_idx]
61            del irrs_2[irr_2_idx]
62            irrs_1_idxs.remove(irr_1_idx)
63            irrs_2_idxs.remove(irr_2_idx)
64            irrs_1_idxs = [idx - 1 if idx > irr_1_idx else idx for
    idx in irrs_1_idxs]
65            irrs_2_idxs = [idx - 1 if idx > irr_2_idx else idx for
    idx in irrs_2_idxs]
66        if len(irrs_2_idxs):
67            extra_in_2 += len(irrs_2_idxs)
68
69    extra_in_1 += len(irrs_1)
70
71    return exact_equality, found_n, extra_in_1, extra_in_2
```

Codice 3.11: Codice che testa la costanza dei risultati della classificazione in Audio Analyser

Quando si ha a che fare con algoritmi basati sul machine learning è la norma avere risultati non deterministici per via della componente di casualità che spesso vi è inserita. Infatti il test che verifica l'esatta uguaglianza tra le 2 collezioni di irregolarità (`test_audio_classification`) lo si è sempre visto fallire durante le sue esecuzioni, infatti esso non è richiesto dal documento descrivente i test di conformità. Ciò che è richiesto e che si basa sul codice appena citato, invece, è di controllare che tutte e sole le irregolarità presenti nel file di test siano presenti anche nel file in output ad una nuova esecuzione dell'Audio Analyser e che siano correttamente estratti i frammenti di audio agli istanti di tempo richiesti

(punto 1.e di 3.3) e che inoltre la precisione ed il recupero[22] medi delle irregolarità trovate rispetto al dizionario delle irregolarità di riferimento, il dataset 4, siano entrambi > 0.9 considerando l'istante di tempo ed i canali audio a cui appartengono le irregolarità (punto 2 di 3.3).

3.4 Parallelizzare o no? confronto di velocità

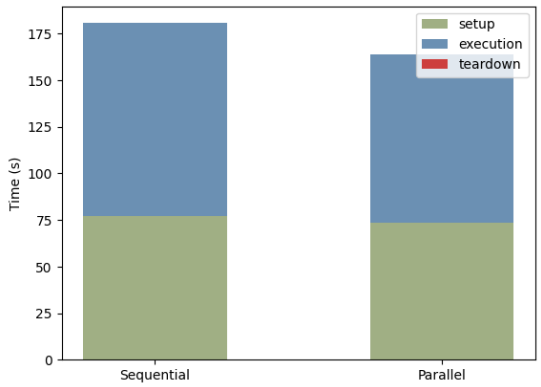
Il plugin di pytest pytest-xdist, già citato nella sottosezione 3.1.1, permette di parallelizzare l'esecuzione dei test.

I test di conformità sono stati eseguiti più volte su una macchina Windows 11 con una CPU Intel i7-1165G7 e 16 GB di RAM DDR4 a 3200 MHz, sia in modalità sequenziale, sia abilitando la parallelizzazione mediante l'opzione `-n auto` (numero di workers uguale al numero di CPUs disponibili) inserita nei files `pytest.ini` e si è osservato innanzitutto che gran parte del tempo utilizzato consiste nell'esecuzione dell'AIM (grafici 3.3a e 3.3b), i quali non sono affetti dall'abilitazione della parallelizzazione ed inoltre che non c'è un vantaggio tangibile ad abilitarla perché la differenza di durata tra le due modalità è spesso quasi nulla. Nel caso del Packager, come si osserva nel grafico in figura 3.3c, i tempi di esecuzione sono comparabili, sebbene si noti una tendenza ad ottenere tempi leggermente minori con l'esecuzione in parallelo, come si può notare anche nel grafico 3.3a. Per quanto concerne l'Audio Analyser, invece, si osserva nel grafico 3.3b che più del 99% della durata totale è occupata dall'esecuzione dell'AIM stesso e che il tempo di esecuzione dei test è nell'ordine del secondo; in questo periodo così ridotto il tempo richiesto per generare e gestire i worker è decisamente rilevante, infatti si osserva che il tempo di esecuzione più che raddoppia in media abilitando la parallelizzazione. Una rappresentazione dei dati è presente in figura 3.3.[23]

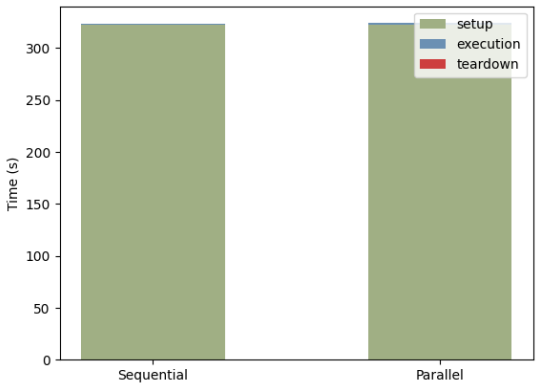
Si è scelto di disabilitare la parallelizzazione a causa degli scarsi risultati

[22] Precisione e recupero (*Precision* e *Recall*) sono entrambe metriche sulle prestazioni di una classificazione: $Precision = \frac{True\ Positives}{True\ Positives+False\ Positives}$; $Recall = \frac{True\ Positives}{True\ Positives+False\ Negatives}$.

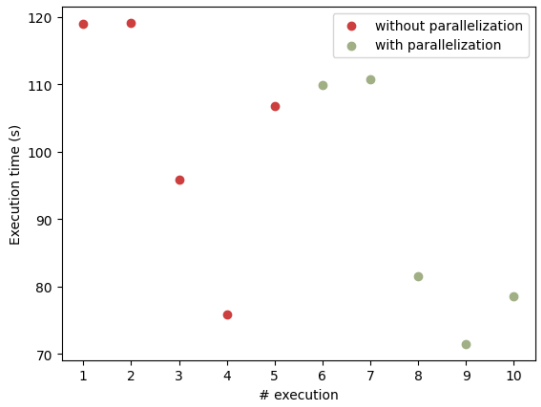
[23] Dati grezzi: `https://github.com/albertopasqualetto/Tesi-triennale/blob/main/Notebooks/res/pkg_timings.json`, `https://github.com/albertopasqualetto/Tesi-triennale/blob/main/Notebooks/res/aa_timings.json`;
grafici e calcoli eseguiti nel notebook Python: `https://github.com/albertopasqualetto/Tesi-triennale/blob/main/Notebooks/create_graphs.ipynb`



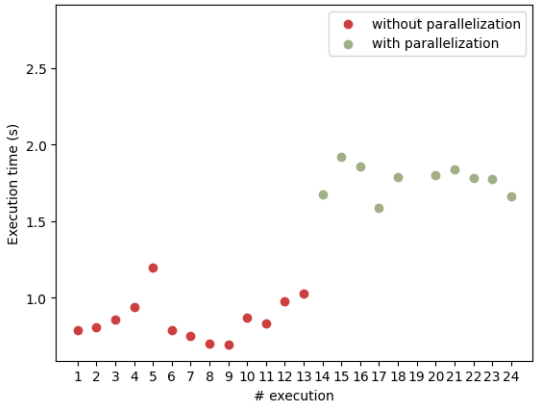
(a) Distribuzione dei tempi tra setup (esecuzione AIM), tests e teardown (pulizia files); AIM Packager



(b) Divisione dei tempi tra setup (esecuzione AIM), tests e teardown (pulizia files); AIM Audio Analyser



(c) Tempo di esecuzione, parallelo vs sequenziale; AIM Packager



(d) Tempo di esecuzione, parallelo vs sequenziale; AIM Audio Analyser

Figura 3.3: Tempi di esecuzione dei test eseguiti sui dataset relativi a 2 nastri

ottenuti con essa abilitata e per consentire l'utilizzo dei CPU cores, ora liberi, agli altri programmi. Ciò non toglie che con un numero maggiore di pacchetti di dataset, la parallelizzazione potrebbe diventare più rilevante in termini di tempo e che si potrebbe considerare di supportarla anche nella fase di esecuzione dell'AIM che, come visto, è molto rilevante nel computo totale di tempo utilizzato.

3.5 Libreria `mpai-cae-arp`

La libreria mpai-cae-arp contiene le classi, le costanti e le funzioni su cui si basano tutti gli AIMs di ARP e che permettono loro di interoperare. Per via della

sua ampia influenza, alcuni problemi relativi a questa libreria sono stati discussi in altre sezioni (vedi 3.3.1).

3.5.1 I problemi di Pydantic

È stato risolto un problema relativo all'uso non univoco di Pydantic, il suo scopo principale all'interno di ARP è quello di fornire alias corrispondenti alla nomenclatura richiesta dalle specifiche tecniche per i parametri delle funzioni in maniera tale da utilizzare il nome dell'alias per l'output verso file (`EditingList.json`, `*_IrregularityFileOutputN.json`, ...); in precedenza l'utilizzo del nome del parametro o del suo alias non era regolato e questo ha portato a degli errori che impedivano la corretta esecuzione del software. La risoluzione applicata consiste nell'utilizzo di `serialization_alias` al posto di `alias`, dato l'utilizzo primario degli alias.

Contestualmente, per favorire l'utilizzabilità nel futuro dell'applicativo, oltre che per limitarne la presenza di bug, si è scelto di aggiornare la libreria alla successiva major release (`^2.3.0`).

Queste modifiche implicano il dover migrare ARP, modificando i nomi dei campi utilizzati in tutti gli AIMs ed impedendo la retrocompatibilità.

3.5.2 Bugfix: formato incorretto delle EditingList scritte su file

Un bug presente nel codice di questa libreria salvava nel filesystem dei file con una formattazione errata, un esempio ne è l'`EditingList.json` che veniva salvato nella seguente maniera:

```
1 "{\n    \"OriginalSpeedStandard\": 7.5,\n    \"
    OriginalEqualisationStandard\": \"IEC2\",\n    \"
    OriginalSampleFrequency\": 96000,\n    \"Restorations\": [\n
      {\n            \"RestorationID\": \"e48cde17-7e5d-4a08-aca9-
    e65c375d0dd6\",\n            \"PreservationAudioFileStart\":
    \"00:00:00.000\",\n            \"PreservationAudioFileEnd\":
    \"00:10:46.400\",\n            \"RestoredAudioFileURI\": \"/data/
    temp/BERIO100/RestoredAudioFile/BERIO100.wav\",\n            \"
    ReadingBackwards\": false,\n            \"AppliedSpeedStandard\":
    7.5,\n            \"AppliedSampleFrequency\": 96000,\n
    \"AppliedEqualisationStandard\": \"IEC2\"\n        }\n    ]\n}"
```

Codice 3.12: `EditingList.json` salvato tramite la precedente modalità (errata): contiene solo una stringa

Il JSON di cui sopra è composto da una sola stringa che contiene la rappresentazione dell'oggetto Python ed i caratteri di escape mostrati, ciò portava al fallimento del relativo test di conformità del Packager (3.2).

Il problema risiedeva nella chiamata di `json.dump` sulla stringa contenente l'oggetto da salvare in JSON generata da `model.json` (in Pydantic V2 chiamata `model.model_dump_json`), questo comportamento probabilmente non era chiaro all'autore di questo codice per via del suo nome fuorviante; utilizzando, invece, la funzione di Pydantic V2 `model.model_dump`, chiamata con parametro `mode='json'` per abilitare la serializzazione degli oggetti presenti nei campi della classe, si ottiene un dizionario Python che può essere scritto correttamente su file JSON (in Pydantic V1 la funzione corrispondente sarebbe stata `model.dict`, ma essa non fornisce la possibilità di serializzare gli oggetti). Ora il file precedente appare correttamente come:

```
{
    "OriginalSpeedStandard": 7.5,
    "OriginalEqualisationStandard": "IEC2",
    "OriginalSampleFrequency": 96000,
    "Restorations": [
        {
            "RestorationID": "e48cde17-7e5d-4a08-aca9-e65c375d0dd6",
            "PreservationAudioFileStart": "00:00:00.000",
            "PreservationAudioFileEnd": "00:10:46.400",
            "RestoredAudioFileURI": "/data/temp/BERIO100/
    RestoredAudioFile/BERIO100.wav",
            "ReadingBackwards": false,
            "AppliedSpeedStandard": 7.5,
            "AppliedSampleFrequency": 96000,
            "AppliedEqualisationStandard": "IEC2"
        }

    ]
}
```

Codice 3.13: `EditingList.json` salvato tramite l'attuale modalità (corretta): contiene il dizionario con i suoi valori serializzati

3.5.3 Bugfix: oggetti AudioWave non funzionanti, overflow e problemi con tracce a singolo canale

La problematica di overflow limitata ai sistemi Windows descritta alla sottosezione 3.3.1 è stata corretta col seguente codice:

```
@@ -202,7 +202,7 @@ class AudioWave:
  data = np.array([
      int.from_bytes(raw_data[i:i + bit // 8], byteorder='little',
    signed=True)
```

```
4         for i in range(0, len(raw_data), bit // 8)
5 - ])
6 + ], dtype=np.int64)  # int32 (default in Windows) leds to overflow
7   data = np.reshape(data, (-1, channels))
8   return AudioWave(data, bit, channels, samplerate)
```

Codice 3.14: Risoluzione del problema di overflow degli oggetti AudioWave

Questo, però, non era l'unico problema della classe AudioWave, infatti non erano supportate le tracce a singolo canale dato che l'indicizzazione di NumPy (operatore `[ ]`) non permette di operare su un array 1D come se fossero multidimensionali (ad esempio indicando il primo canale con `array[:,1]`).

```
1 @@ -369,39 + 369, 43 @@ class AudioWave:
2   def get_channel(self, channel: int):
3       # [...]
4       if channel not in range(self.channels):
5           raise IndexError("Channel not found")
6 -     return AudioWave(self.array[:, channel], self.bit, 1, self.
    samplerate)
7 +
8 +     if self.channels == 1:
9 +         return AudioWave(self.array.squeeze(), self.bit, 1, self.
    samplerate)
10 +     else:
11 +         return AudioWave(self.array[:, channel], self.bit, 1, self.
    samplerate)
```

Codice 3.15: Modifiche ad AudioWave.get_channel che permettono di utilizzarlo con tracce ad 1 canale

È necessario, perciò, introdurre una condizione sul numero di canali presenti e, nel caso ve ne sia uno solo, non operare con l'indicizzazione, come si può osservare dal codice mostrato.

3.5.4 Aggiornamenti di versione

Per lo stesso motivo descritto nella sottosezione 3.5.1, non solo Pydantic è stato aggiornato, ma anche le altre librerie necessarie.

La libreria librosa (analisi audio) richiede la libreria Numba (compilatore JIT) che, a sua volta, richiede llvmlite (Python bindings per LLVM: un'infrastruttura di compilazione) e NumPy (supporto ed alta efficenza computazionale per array e matrici di grandi dimensioni e funzioni matematiche); tutte queste librerie sono state aggiornate all'ultima versione compatibile tra loro, rimuovendo la dipendenza diretta `llvmlite = "^0.39.1"` ed il blocco alla versione di NumPy 1.23.3 precedentemente impostati.

```
1 @@ -9,8 +9,7 @@
```

```
2   [tool.poetry.dependencies]
3   python = "^3.10"
4 - numpy = "1.23.3"
5 + numpy = "~1.24.4"    # because of numba (from librosa) requirement
      <=1.24
6 - pydantic = "^1.10.7"
7 + pydantic = "^2.3.0"
8   pyyaml = "^6.0"
9 - llvmlite = "^0.39.1"
10 - librosa = "^0.10.0.post2"
11 + librosa = "^0.10.1"
12  grpcio-tools = "^1.53.0"
```

Codice 3.16: `pyproject.toml`, aggiornamento delle dipendenze

Questa modifica è possibile solo in seguito all'1 maggio 2023 (data successiva all'ultimo commit presente nella repository prima dell'inizio di questo lavoro), quando le librerie citate hanno introdotto la cross-compatibilità tra le nuove versioni.[24]

Questi aggiornamenti hanno permesso la possibilità, prima non presente, di utilizzare la libreria, e quindi di conseguenza ARP con la versione di Python 3.11, rendendo la libreria più "a prova di futuro".

[24]Dipendenza di librosa da Numba: `https://github.com/librosa/librosa/blob/43d4427f7a389acc79b5be3c924646701f19bb46/setup.cfg#L67`;
Dipendenze di Numba da llvmlite e NumPy: `https://github.com/numba/numba/blob/596e8a55334cc46854e3192766e643767bd7c934/setup.py#L367`;
Numba 0.57 introduce il supporto a Python 3.11 e NumPy 1.24: `https://numba.readthedocs.io/en/stable/release-notes.html#version-0-57-0-1-may-2023`;
llvmlite 0.40.0 introduce il supporto a Python 3.11: `https://llvmlite.readthedocs.io/en/latest/release-notes.html#v0-40-0-may-1-2023`.

4 Conclusioni

In conclusione si è visto perché e come conservare opere musicali seguendo la traccia del software del CSC. I test di conformità sono stati scritti seguendo il documento che li descrive; talvolta si è percepita la pesantezza dell'articolata struttura di MPAI per via dei molteplici documenti, anche ridondanti, che vengono prodotti, ma ciò probabilmente è dovuto alle dimensioni, le estensioni e la voglia di crescere dell'organizzazione. Tale "pesantezza" non è, però, avvertita nello sviluppo software, che invece è gestito dai vari sottogruppi.

Non sono stati riscontrati problemi irrisolti durante lo sviluppo del codice.

4.1 Cosa manca

Per completare i lavori mancano i test di conformità per gli altri 3 AIMs, che non sono stati codificati per limitazioni di tempo.

Inoltre sarebbe opportuno ritoccare le soglie del controllo di uguaglianza per audio e video nei test del Packager (sottosezioni 3.2.2 e 3.2.3) potendo disporre di più pacchetti di datasets.

4.2 Obiettivi futuri

Come anticipato nella relativa sottosezione 3.2.3, si può studiare un metodo più preciso nel confrontare i flussi audio rispetto a quello presentato.

Inoltre, ai fini di migliorare il prodotto finale si potrebbe:

- Aggiungere uno stadio di miglioramento dell'audio rimuovendo le alterazioni eventualmente incontrate nella traccia audio (click, rumore, …) [3]. Una proposta simile era già stata ipotizzata in MPAI-CAE, come si può vedere dalla presenza di un AIM chiamato "Audio Enhancer" in [12] nel 2021, quest'ultimo è stato rimosso nelle iterazioni successive, le quali hanno portato alla versione che attualmente è standard: *IEEE Standard Adoption of Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Technical Specification Context-based Audio Enhanced (CAE) Version 1.4*.

- Potenziare l'AIM Tape Audio Restoration abilitandolo a modificare, magari tramite l'uso di Intelligenza Artificiale, l'audio della copia di accesso in corrispondenza delle irregolarità video grazie alle informazioni sulla loro tipologia. Tali discontinuità possono aver deteriorato la resa acustica del nastro.

L'aspetto negativo di queste proposte è la diminuzione della fedeltà dell'audio in output, ma ciò è completamente accettabile se operato solo e soltanto sulla copia di accesso.

Bibliografia

[1] Sergio Canazza, Giovanni De Poli e Alvise Vidolin. «Gesture, Music and Computer: The Centro di Sonologia Computazionale at Padova University, a 50-Year History». In: *Sensors* 22.9 (2 mag. 2022), p. 3465. ISSN: 1424-8220. DOI: 10.3390/s22093465. URL: https://www.mdpi.com/1424-8220/22/9/3465.

[2] Carlo Fantozzi et al. «Tape music archives: from preservation to access». In: *International Journal on Digital Libraries* 18.3 (set. 2017), pp. 233–249. ISSN: 1432-5012, 1432-1300. DOI: 10.1007/s00799-017-0208-8. URL: http://link.springer.com/10.1007/s00799-017-0208-8.

[3] Simon Godsill, Peter Rayner e Olivier Cappé. «Digital Audio Restoration». In: (1998).

[4] Ylenia Grava. «Progettazione e sviluppo di software per il riconoscimento e la classificazione di discontinuità nei nastri magnetici audio, basato su tecniche di computer vision e neural network». Tesi di laurea Magistrale. Università degli Studi di Padova, 4 gen. 2019. URL: https://hdl.handle.net/20.500.12608/27443.

[5] IEEE e MPAI. *IEEE Standard Adoption of Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Technical Specification Artificial Intelligence Framework (AIF) 1.1*. ISBN: 9781504493345. IEEE, 2022. DOI: 10.1109/IEEESTD.2023.10112600. URL: https://ieeexplore.ieee.org/document/10112600/ (visitato il 07/09/2023).

[6] IEEE e MPAI. *IEEE Standard Adoption of Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) Technical Specification Context-based Audio Enhanced (CAE) Version 1.4*. ISBN: 9781504493420. IEEE, 2022. DOI: 10.1109/IEEESTD.2023.10112597. URL: https://ieeexplore.ieee.org/document/10112597/ (visitato il 07/09/2023).

[7] Leonardo. *A new way to develop useful standards*. Leonardo's Blog. 20 Nov. 2020. URL: https://blog.chiariglione.org/a-new-way-to-develop-useful-standards/ (visitato il 07/09/2023).

[8] MPAI. *About - MPAI community*. MPAI community. URL: https://mpai.community/about/ (visitato il 02/09/2023).

[9] MPAI. *About MPAI-CAE*. MPAI community. URL: https://mpai.community/standards/mpai-cae/about-mpai-cae/ (visitato il 06/09/2023).

[10] MPAI. *Application Note #1 Rev. 1 - MPAI CAE*. MPAI community. URL: https://mpai.community/standards/mpai-cae/mpai-application-note-1-rev-1/ (visitato il 07/09/2023).

[11] MPAI. *MPAI-AIF - MPAI community*. MPAI community. URL: https://mpai.community/standards/mpai-aif/ (visitato il 04/09/2023).

[12] MPAI. *MPAI-CAE Use Cases and Functional Requirements*. 17 Feb. 2021.

[13] MPAI. *Some MPAI data coding standards*. MPAI community. URL: https://mpai.community/some-mpai-data-coding-standards/ (visitato il 05/09/2023).

[14] MPAI. *Structure of MPAI standards*. MPAI community. URL: https://mpai.community/structure-of-mpai-standards/ (visitato il 05/09/2023).

[15] MPAIstandards. *MPAI presents Context-based Audio Enhancement and Reference Software online 2023-07-07*. 10 Lug. 2023. URL: https://www.youtube.com/watch?v=32M2Bc5tbI0 (visitato il 05/09/2023).

[16] Niccoló Pretto et al. «Computing Methodologies Supporting the Preservation of Electroacoustic Music from Analog Magnetic Tape». In: *Computer Music Journal* 42.4 (1 dic. 2018), pp. 59–74. ISSN: 0148-9267, 1531-5169. DOI: 10.1162/comj_a_00487. URL: https://direct.mit.edu/comj/article/42/4/59/94833/Computing-Methodologies-Supporting-the.

[17] Nicola Raimo et al. «Digitalization in the cultural industry: evidence from Italian museums». In: *International Journal of Entrepreneurial Behavior & Research* 28.8 (22 nov. 2022), pp. 1962–1974. ISSN: 1355-2554. DOI: 10.1108/IJEBR-01-2021-0082. URL: https://www.emerald.com/insight/content/doi/10.1108/IJEBR-01-2021-0082/full/html.

[18] Donald Rakemane e Olefhile Mosweu. «Challenges of managing and preserving audio-visual archives in archival institutions in Sub Saharan Africa: a literature review». In: *Collection and Curation* 40.2 (8 apr. 2021), pp. 42–50. ISSN: 2514-9326, 2514-9326. DOI: 10.1108/CC-04-2020-0011. URL: https://www.emerald.com/insight/content/doi/10.1108/CC-04-2020-0011/full/html.

[19] Alessandro Russo, Matteo Spanio e Sergio Canazza. «Enhancing Preservation and Restoration of Open Reel Audio Tapes Through Computer Vision». 2023.

[20] Matteo Spanio. «A study on Equalization Curve Detection in Audio Tape Digitization process using Artificial Intelligence». Tesi di laurea Triennale. Università Ca' Foscari Venezia, mar. 2023. DOI: 10.13140/RG.2.2.36838.50247. URL: https://rgdoi.net/10.13140/RG.2.2.36838.50247.

[21] Unesco. *Recommendation concerning the Protection and Promotion of Museums and Collections, their Diversity and their Role in Society adopted by the General Conference at its 38th Session, Paris, 17 November 2015*. 17 Nov. 2015.

Ringraziamenti

Non posso fare a meno che ringraziare il prof. Sergio Canazza, nonchè mio relatore, che mi ha proposto temi molto interessanti, tra i quali ARP, rendendo possibile il mio contributo al progetto; lo ringrazio anche per la sua costante disponibilità e celerità per qualsiasi chiarimento.

Inoltre ringrazio il dott. Alessandro Russo che mi ha introdotto al mondo della digitalizzazione audio e, soprattutto, ringrazio il dott. Matteo Spanio, anche lui sempre disponibile, che mi ha seguito durante tutto lo sviluppo del software rispondendo alle mie email notturne.

Ringrazio tutti i miei colleghi, i quali sono stati una parte fondamentale dell'esperienza universitaria.

Ci tengo, infine, a ringraziare Giulia, la mia ragazza, che mi ha saputo ascoltare e che mi ha aiutato a rileggere e rifinire l'elaborato.