Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

Estrazione del Pattern Noise da video per un processo di identificazione di una fotocamera sorgente

Relatore:
Chiar.mo Prof.
Danilo Montesi

Presentata da:
Enrico Valguarnera

Correlatore:
Ph.D. Flavio Bertini

Sessione III
Anno Accademico 2018-19

"Noi tutti - filosofi, scienziati e gente comune - dovremmo allora essere in grado di partecipare alla discussione del problema del perché noi e l'universo esistiamo. Se riusciremo a trovare la risposta a questa domanda, decreteremo il trionfo definitivo della ragione umana: giacché allora conosceremo la mente di Dio."

Stephen Hawking

Introduzione

Oggigiorno, devices digitali quali telefoni cellulari, smartphone, tablet e palmari sono diventati di uso comune e ancor di più costituiscono oggetti di utilizzo personale, rivelandosi ormai indispensabili. Tali devices posseggono un elevato numero di sensori, come la fotocamera, tramite i quali riusciamo ad ottenere numerose e rilevanti informazioni.
Ogni giorno viene prodotta una quantità spropositata di dati digitali e proprio per questo motivo l'abuso dell'utilizzo di tali dati potrebbe comportare danni irreparabili. Registrazioni illegali di proiezioni di film nei cinema hanno causato ingenti danni. Inoltre è possibile manipolare i contenuti digitali tramite tools che permettono di modificarne i metadati come ad esempio quelli presenti nell'header di un file video, quali brand e modello del dispositivo samrtphone, data, ora, etc. Modificando tali informazioni è facile compromettere contenuti digitali che potrebbero essere oggetto di prova di un processo giuridico. Dunque acquista maggiore importanza il ruolo dell'informatica forense alla quale spetta il compito di assicurare l'integrità dell'analisi di contenuti digitali. Si può quindi pensare di ottenere a partire da un'immagine o da un video, effettuando studi mirati, l'unicità di tale dato e sapere da quale sorgente è stato catturato. Quest'area di ricerca è conosciuta come *source camera identification*. L'elaborato si prefigge di analizzare un dataset di video, messo a disposizione dal dipartimento di Informatica dell'Università di Bologna, al fine di effettuare un procedimento di estrazione del rumore caratteristico tramite una tecnica di eliminazione del rumore, in modo da costruire un insieme di firme identificative, denominate nel resto

dell'elaborato come *fingerprint*, infine confrontando i risultati estratti in una fase di correlazione.

Sarebbe inoltre interessante ottenere informazioni significative su un contenuto digitale, come ad esempio un video, che è stato postato su un social network da un profilo sconosciuto, magari utilizzato per propositi illegali, e determinare se tale video è stato generato da una fotocamera di un altro video postato da un profilo conosciuto, quindi associare i due profili come derivanti da uno stesso utente. Proprio per questo i video del dataset verranno sottoposti ad una routine di upload e download sulla piattaforma di messaggistica Whatsapp, e successivamente verrà effettuato il procedimento di estrazione anche dai video così scaricati.
Infine, è effettuata un'analisi di correlazione lineare delle firme. Ci si aspetta che le fingerprint estratte da video catturati da una stessa sorgente abbiano un alto valore di correlazione.
La tesi è stata articolata nel seguente modo:

Capitolo 1
Nel primo capitolo verrà introdotto l'argomento discutendo sullo stato dell'arte, facendo una rassegna sulle tecniche principali per effettuare un processo di identificazione della fotocamera sorgente, identificando la tecnica basata sull'estrazione del Photo Response Non-Uniformity come quella più adatta al processo che verrà portato avanti nel resto dell'elaborato. L'ultimo paragrafo, presenta l'algoritmo utilizzato per effettuare il processo di eliminazione del rumore basato su tale tecnica, proposto dall'Università di Finlandese di Tampere.

Capitolo 2
Nel secondo capitolo si passa alla descrizione della parte pratica. Sono discussi gli esperimenti preliminari, che hanno avuto luogo nella prima fase di lavoro di tesi, di fondamentale importanza per la fase di approccio al problema e per la scelta finale della strategia da attuare. Viene presentato il

dataset, ovvero l'insieme di video sui quali sarà applicato il procedimento di estrazione del rumore, disquisendo sulla sua stessa conformazione e sulle sue particolarità. E' presentata la strategia di estrazione progettata ed applicata ai fini del raggiungimento dell'obiettivo, focalizzando l'attenzione sull'organizzazione degli spazi di lavoro, analizzando i file che li compongono. Si focalizza inoltre l'attenzione sulla routine al quale sono stati sottoposti i video del dataset per il caricamento e il conseguente scaricamento sull'applicazione Whatsapp. Infine, è riassunto l'output finale che ci si aspetta al termine del procedimento di estrazione.

Capitolo 3

In questo capitolo viene presentata la fase di manipolazione sui dati grezzi ottenuti dalla fase di estrazione. E' riassunta la strategia di ricombinazione delle fingerprint, sono presentate le modalità del calcolo della media che vengono applicate alle fingerprint e il necessario e preventivo ridimensionamento delle matrici prima della fase di correlazione. Pertanto è presentata anche la tecnica di correlazione lineare o calcolo del coefficiente di correlazione di Pearson.

Capitolo 4

Infine, nel capitolo 4, è presentata l'analisi sui dati estratti, effettuata mettendo in correlazione diversi gruppi di coppie di fingerprint, così producendo in output diversi grafici esplicativi.
Si discuterà dei diversi metodi applicati per l'ottenimento delle fingerprint finali, traendo delle conclusioni sull'affidabilità di tali metodi, al fine di costruire una buona fingerprint.

Conclusioni

Alla fine dell'elaborato, verranno presentate le conclusioni finali tratte dalle varie analisi effettuate.

Indice

Introduzione **3**

Elenco dei listati **13**

1 Stato dell'arte **15**

1.1 Photo Response Non-Uniformity 16

1.2 Filtro di denoising V-BM4D 18

1.3 Conditional Probability Features 20

1.4 SCI con deep learning 21

2 Estrazione delle Fingerprints **23**

2.1 Esperimenti preliminari 24

2.2 Dataset 27

2.3 Procedimento di estrazione 29

2.3.1 Strategia di estrazione 30

2.3.2 Cartella di progetto in remoto 32

2.3.3 Filtro V-BM4D 37

2.3.4 Cartella di progetto in locale 39

2.3.5 Routine sul social Whatsapp 40

2.3.6 Output finale 41

3 Manipolazione sui risultati estratti **45**

3.1 Ricombinazione delle fingerprint 46

3.1.1 Calcolo della media su tutti i frame 46

3.1.2 Calcolo della media su I-Frame 47
3.2 Correlazione di Pearson 49
3.2.1 Ridimensionamento delle fingerprint mediate 50

4 Analisi dei risultati: correlazione tra le fingerprint 53
4.1 Correlazioni: istogrammi 55
4.2 Correlazioni: confronto totale tra le fingerprint 62

Conclusioni 67

A Codice sorgente per l'esecuzione dell'estrazione 69
A.1 Bash Script 69
A.1.1 Funzione iframe_indices_extraction.sh 71
A.2 MATLAB Script 71
A.2.1 Funzione Start_video_analysis.m 71
A.2.2 Funzione Analyze_video_dataset.m 72
A.2.3 Funzione read_video_range.m 77
A.2.4 Funzione sendmail_cbd.m 78

B Codice sorgente per la fase di manipolazione sulle fingerprint 81
B.1 Calcolo della media delle fingerprint 81
B.1.1 Funzione fingerprint_normal_avarage.m 81
B.1.2 Funzione fingerprint_iframe_avarage.m 83
B.1.3 Funzione getIFramesIndices 85

Bibliografia 87

Elenco delle figure

2.1 Struttura del database SMART_VIDEO 28
2.2 Scomposizione della fingerprint estratta da un singolo video . . 30
2.3 Struttura della directory Fingerprint 33
2.4 Struttura della directory Fingerprint 38

3.1 Grafico per il confronto delle metriche per il ridimensionamento delle fingerprint 51

4.1 Istogrammi - correlazioni fingerprint originali 57
4.2 Istogrammi - correlazioni fingerprint scaricate 58
4.3 Istogrammi - correlazioni fingerprint originali solo I-Frame . . 60
4.4 Istogrammi - correlazioni fingerprint scaricate solo I-Frame . . 61
4.5 Totalità delle correlazioni tra le medie delle fingerprint originali 63
4.6 Totalità delle correlazioni tra le medie delle fingerprint estratte da video scaricati 65
4.7 Correlazioni originali e scaricati su I-Frame 66

Elenco delle tabelle

2.1 Elenco dei dispositivi 27
2.2 Dimensione dei file di output parziali 42
2.3 Dimensione dei file di output totali 43

4.1 Proiezione degli utenti e le fotocamere utilizzate 55

Elenco dei listati

2.1 Start_video_analysis script 35
3.1 calcolo media 47
3.2 calcolo media 49
A.1 bash_script_parallel.sh 69
A.2 iframe_indices_extraction.sh 71
A.3 Start_video_analysis.m 72
A.4 Analyze_video_dataset.m 72
A.5 read_video_range.m 77
A.6 Start_video_analysis.m 78
B.1 fingerprint_normal_avarage.m 82
B.2 fingerprint_iframe_avarage.m 84
B.3 getIFramesIndices.m 86

Capitolo 1

Stato dell'arte

Un procedimento di *Source Camera Identification* (SCI), o processo di identificazione della fotocamera sorgente, si prefigge di identificare una firma unica, che possa essere utilizzata per affermare con certezza che un determinato contenuto digitale, come un'immagine o un video, sia stato catturato da una certa fotocamera di un particolare dispositivo elettronico. Questo tipo di processo si differenzia dalla verifica di integrità, nella quale si cerca di determinare se un contenuto digitale sia stato manomesso.

La firma unica è chiamata *fingerprint*. Come studiato in [1] tale *fingerprint* risulta essere un oggetto identificativo per la fotocamera del dispositivo dal quale è stata catturata la foto o il video. Il procedimento di estrazione di fingerprint da video si basa principalmente su tecniche di eliminazione del rumore dai contenuti digitali o processi di *denoising*.
Un'immagine può essere rappresentata con la seguente formula:

$$I = I_0 + K \cdot I_0 + N_t, \tag{1.1}$$

dove N_t corrisponde alla combinazione di componenti indipendenti di rumore casuale, I_0 è l'immagine originale e K corrisponde esattamente alla fingerprint unica della camera.

Per ottenere la fingerprint FP, si applica un filtro di denoising, cioè un filtro di eliminazione del rumore dal video in questione e successivamente viene

calcolato il valore finale effettuando una differenza tra l'immagine originale e l'immagine ottenuta in output dopo l'applicazione del filtro:

$$FP = I - F(I), \tag{1.2}$$

dove la funzione F è la funzione che viene applicata all'immagine per effettuare il processo di denoising.

Le tecniche per la SCI sono di grande importanza nell'ambito dell'informatica forense. Quando le riprese digitali sono collegate alla criminalità, potrebbe essere utile identificare la telecamera sorgente, applicando tali metodologie. Numerosi casi di pedopornografia potrebbero essere risolti grazie all'applicazione di queste tecniche.
Nel prosieguo di questo elaborato parleremo di Video Camera Identification, dato che tutti gli esperimenti sono stati portati avanti su contenuti digitali quali video.
Nelle sezioni seguenti verranno elencati i metodi principali per l'estrazione di fingerprint da video.

1.1 Photo Response Non-Uniformity

Per il processo di identificazione di una sorgente è importante avere un'impronta riscontrabile in più contenuti digitali, catturati dalla stessa sorgente. Per chiarezza, le fingerprint estratte da un video V_1 e da un video V_2, se catturate dalla sorgente S, avranno un alto valore di correlazione.

Si intende anche ottenere un pattern che sia presente sia nel materiale digitale di cui si conosce l'identità, sia in quello di profilo sconosciuto. [1] Tale pattern è noto come *fixed pattern noise* (FPN) ed è causato dai sensori che compongono la fotocamera, come CCD e CMOS, adibiti per l'acquisizione degli impulsi luminosi e per la conseguente conversione del segnale, che per costruzione sono costituiti da imperfezioni, dovute a difetti di fabbricazione. In [2] è stato dimostrato che tali difetti possono essere sfruttati per l'identificazione di una sorgente, analizzando il rumore da essi introdotto.

In [1] è stato proposto un metodo migliore per il processo di source camera identification che pone l'attenzione, non più sui difetti dei sensori bensì, sulla differenza di sensibilità di ogni pixel che viene sottoposto alla luce, che risulta essere non uniforme. Questa è proprio la definizione di Photo Response Non-Uniformity (PRNU). Il PRNU descrive il rapporto tra la potenza ottica su un pixel rispetto all'uscita del segnale elettrico. Tale parametro ci informa della variazione di sensibilità dei pixel in una condizione di pari attraversamento di luce. La variazione dipende dalla dimensione non uniforme dell'area attiva dove verrà captata la luce e assorbita sotto forma di fotoni. Viene così spiegata la caratteristica moltiplicativa del PRNU: quando la dimensione dell'area attiva investita dalla luce cresce di un fattore x, anche il numero di fotoni intercettati cresce con lo stesso fattore. Ciò si può notare dal prodotto presente nell'equazione 1.1.
Come il *fixed pattern noise*, il PRNU deriva anch'esso da imperfezioni di costruzione dei sensori ed è stato dimostrato da Goljan et al in [3] essere sempre presente ed unico. Una caratteristiche molto importante del PRNU è che non solo riesce a distinguere fotocamere di differenti modelli, ma anche tra fotocamere di uno stesso brand e di uno stesso modello.
E' inoltre possibile migliorare tale tecnica. In [4] è stato studiato che le caratteristiche di intensità dei singoli pixel e i dettagli ad alta frequenza dell'immagine, come angoli ed oggetti, possono notevolmente influenzare la qualità del PRNU estratto. In particolare il PRNU può essere estratto solo in un determinato range di intensità. Tramite diversi esperimenti è stato dimostrato che tale PRNU non può essere estratto per range di intensità estremamente bassi o alti. Per valori di intensità $0, 1, 2, 254, 255$, un filtro di denoising non è capace di eliminare il rumore.
Un altro problema riguarda la presenza di dettagli in un'immagine, i quali riducono nella fingerprint estratta il numero di *false rejection rate*. Per questo motivo, immagini che non dovrebbero essere riconosciute sono erroneamente giudicate come valide al riconoscimento nel processo di SCI.
Come soluzione a questo problema Tiwari et al hanno proposto l'applicazio-

ne di una funzione di ponderazione costruita a partire dalle caratteristiche dell'immagine presenti e persistenti nel PRNU. Dopo aver studiato tali caratteristiche effettuando un'analisi sul PRNU estratto, viene applicata tale funzione di peso all'immagine facendo si che venga dato più peso a regioni dell'immagine più rappresentative , e meno peso a regioni meno rappresentative le quali devono avere meno rilevanza ai fini dell'estrazione.

Il PRNU costituisce quindi la fingerprint per una determinata sorgente e pur essendo stato scoperto non recentemente costituisce un metodo valido e affidabile per il processo di source camera identification.

1.2 Filtro di denoising V-BM4D

Il gran numero di applicazioni pratiche che coinvolgono video digitali ha motivato un interesse significativo nel risolvere problematiche strettamente legate a tali contenuti, e la letteratura contiene una grossa quantità di tali algoritmi. Al momento, l'approccio più efficace nel ripristino di immagini o video sfrutta la ridondanza data dalla somiglianza non locale tra i sotto gruppi selezionati in diverse posizioni all'interno del dato.
Sono stati proposti differenti algoritmi basati su questo approccio per la risoluzione di vari problemi di elaborazione delle immagini, ma principalmente per processi di denoising. Tra questi metodi, menzioniamo in particolare l'algoritmo BM3D, che rappresenta lo stato dell'arte per il procedimento poc'anzi citato. BM3D si basa sul cosiddetto paradigma di raggruppamento e filtraggio collaborativo: l'osservazione viene elaborata a blocchi, e i blocchi di immagini bidimensionali simili tra loro sono impilati in un *gruppo* 3-D (fase di raggruppamento), il quale viene successivamente filtrato attraverso una trasformazione in restringimento (filtro collaborativo), fornendo simultaneamente stime diverse per ogni blocco raggruppato. Queste stime vengono quindi riposte alle rispettive posizioni originali e infine eventualmente aggregate nella stima finale dell'immagine.
Il filtro BM3D è stato applicato con successo al denoising di video, creando

così la variante per i video V-BM3D. Inoltre, è stato utilizzato per molte altre applicazioni, come il raggiungimento di un certo grado di nitidezza e/o dissolvimento dell'immagine. In V-BM3D, i gruppi sono array 3-D contenenti blocchi reciprocamente simili, estratti da una serie di fotogrammi consecutivi della sequenza video. Un gruppo può includere più blocchi dello stesso frame, sfruttando naturalmente in questo modo la somiglianza non locale. Tuttavia, è tipicamente lungo la dimensione temporale che si possono trovare i blocchi più simili tra loro, cioè sul confronto di blocchi che appartengono a frame differenti.

È risaputo che i video con compensazione del movimento sono estremamente fluidi lungo l'asse temporale e questo fatto è sfruttato da quasi tutte le moderne tecniche di codifica video; la somiglianza lungo le traiettorie del movimento è molto più forte della somiglianza non locale esistente all'interno di un singolo fotogramma.

Nonostante ciò, in V-BM3D i blocchi sono raggruppati indipendentemente dal fatto che la loro somiglianza sia dovuta al tracciamento del movimento lungo il tempo o all'auto-somiglianza non locale all'interno di ciascun fotogramma. Dunque si può concludere che il filtraggio in V-BM3D non è in grado di distinguere tra somiglianza non locale temporale e spaziale. E' un limite significativo che ha portato lo studio ad un riadattamento dell'algoritmo. Difatti, è stato sviluppato un altro filtro, il filtro V-BM4D, che rappresenta lo stato dell'arte nei processi di denoising di video. V-BM4D supera le suddette debolezze riscontrate in V-BM3D, sfruttando separatamente sia la ridondanza temporale, sia quella spaziale nella sequenza video. L'elemento centrale di V-BM4D è il volume spaziotemporale, una struttura tridimensionale formata da una sequenza di blocchi estratti dal video affetto da rumore che segue una traiettoria specifica, ottenuta, ad esempio, concatenando i vettori di movimento lungo il tempo. Pertanto, V-BM4D non raggruppa blocchi, ma volumi spaziotemporali reciprocamente simili secondo una procedura di ricerca non locale. Quindi, questi gruppi sono strutture con 4 dimensioni formate a loro volta da volumi 3D e il filtraggio collaborativo

viene eseguito tramite una trasformata spazio-temporale separabile 4D. Dato il tipo di struttura che si viene a creare, la trasformazione sfrutta i seguenti tre tipi di correlazione che caratterizzano le sequenze originali dei video:

- la correlazione spaziale locale tra i pixel in ciascun blocco di un volume;
- la correlazione temporale locale tra i blocchi di ciascun volume;
- la correlazione spaziale e temporale non locale tra volumi raggruppati;

Lo spettro del gruppo 4D così ottenuto è molto sparso, il che rende il restringimento più efficace rispetto a V-BM3D e si traduce nelle prestazioni superiori di V-BM4D in termini di riduzione del rumore. Nella sezione dedicata al filtro V-BM4D 2.3.3, nel seguito dell'elaborato, verrà descritto l'algoritmo implementato.

1.3 Conditional Probability Features

Un altro medoto per implementare un processo di source camera identification è quello basato sulle *Conditional Probability Features* (CP Features). Questo metodo si basa sulla nozione di probabilità condizionata, come suggerito dal nome stesso. La probabilità condizionata di un evento A rispetto a un evento B è la probabilità che si verifichi A, sapendo che B è verificato. Tale probabilità si indica con la formula:

$$P(A|B) = \frac{P(B \cap A)}{P(B)}, \tag{1.3}$$

oppure,

$$P_A(B). \tag{1.4}$$

Le CP Features sono state proposte per la prima volta in [12] per applicazioni sulla steganografia. Tali probabilità vengono raccolte effettuando

un'analisi sui coefficienti della trasformata discreta del coseno di un'immagine in formato JPEG. Si può ottenere un'informazione probabilistica su tali coefficienti osservando l'orientamento. Così le CP Features sono generate a partire dagli orientamenti possibili per variabili ausiliarie p, q ed r; tali variabili di riferimento possono assumere qualsiasi posizione all'interno del blocco 8×8 dei coefficienti DCT dell'immagine JPEG, andando così ad effettuare un'analisi per i differenti orientamenti possibili: verticale, orizzontale e obliquo. Per ogni posizione, le variabili p, q ed r attraverseranno i coefficienti DCT, rastrellando l'intero blocco.
Le CP Features vengono estratte dai valori di luminanza prima dell'inizio del processo di classificazione.
Infine viene effettuato un processo di classificazione di tali features estratte. In [10] Yahaya et al, per condurre tale processo hanno utilizzato una SVM (Support Vector Machine), riuscendo ad ottenere un'accuratezza del 97.2%.

1.4 SCI con deep learning

Uno degli ambiti informatici che ha avuto un'evidente avanzamento è quello del machine learning, che ha portato innovazione, fornendo nuove tecniche per l'apprendimento automatico delle macchine.
Negli ultimi anni, tecniche di *deep learning* sono state applicate anche nel processo di source camera identification. In [13], Freire-Obregon et al hanno proposto l'applicazione di reti neurali convoluzionali (CNN) per il raggiungimento di tale scopo.
Questo tipo di reti, recentemente hanno riscosso successo in processi di riconoscimento di immagini, analisi di video e processamento del linguaggio naturale (NLP).
Una rete convoluzionale consiste in una serie di *layer*, ed ogni layer consiste in una serie di filtri passa alto che vengono applicati su l'intera immagine. Una rete convoluzionale ha come input l'immagine. Questa viene trasformata nel passaggio da un filtro ad un altro, ottenendo infine in output un vettore di

dimensione N, dove N è il numero di classi sulle quali è stata effettuata la fase di training.

Dopo aver creato la struttura della rete convoluzionale, che può essere composta da layer convoluzionali, layer di *pooling* e layer *fully-connected* e dopo aver settato tutti i vari iperparametri, si passa alla fase di apprendimento.

Generalmente la scelta di una determinata struttura, e il settaggio degli iperparametri avviene in maniera sperimentale, scegliendo la configurazione migliore in base al comportamento del modello.

Capitolo 2

Estrazione delle Fingerprints

In questo capitolo verrà presentato il lavoro pratico sperimentale effettuato sui video. L'obiettivo di tale analisi è di estrarre il rumore caratteristico da ogni video presente nel dataset di riferimento, fornito dall'Università di Bologna. Verrà così estratto il rumore residuo da ogni video per mezzo dell'applicazione del filtro di denoising VMB4D, ideato e sviluppato dal dipartimento di Signal Processing dell'Università di Tampere, Finlandia.
Prima di effettuare la fase vera e propria dell'estrazione, sono stati intrapresi numerosi esperimenti al fine di prendere confidenza sia con i tipi di dato da analizzare, sia per mettere a punto gli algoritmi necessari per la buona riuscita dell'estrazione.
E' stata effettuata un'analisi del data set, che è presentato in seguito, per costatarne la conformazione, e per mappare un elenco delle caratteristiche di ogni video, in relazione alla sorgente di appartenenza. Questo procedimento preventivo è stato messo in atto per evitare di incorrere ad errori futuri durante l'estrazione e per avere maggiore conoscenza dei dati grezzi che sono stati sottoposti all'analisi.
Analizzeremo sia la struttura dello spazio di lavoro creato, mostrando l'organizzazione delle directory di lavoro, elencando i file che le compongono e la strategia che è stata progettata ed applicata per la buona riuscita della sperimentazione, infine si discuterà sull'output finale.

Per portare a compimento tutti gli esperimenti, per sviluppare gli script necessari all'estrazione e alla conseguente analisi dei risultati è stato utilizzato il tool di sviluppo MATLAB, usufruendo della licenza gratuita messa a disposizione dall'Università di Bologna.

2.1 Esperimenti preliminari

Dopo aver studiato i principi teorici sul quale si basano le tecniche fondamentali per l'analisi di immagini, in particolare quelle impiegate per il procedimento di eliminazione del rumore da un contenuto digitale, tecniche che sono state oggetto di studio durante il tirocinio, sono stati intrapresi numerosi esperimenti al fine di prendere confidenza con i dati grezzi presenti nel dataset. Tale fase è stata di fondamentale importanza per la creazione e la raffinazione degli algoritmi finali utili e necessari all'estrazione.
I primi esperimenti sono stati effettuati su video di prova prelevati dal dataset di riferimento. Su tali video sono state testate le funzioni necessarie alla lettura di tali contenuti digitali, alla loro rappresentazione nel tool di sviluppo e in memoria e al processamento di denoising attraverso il filtro che andremo a descrivere in seguito.

Altri esperimenti sono stati condotti sempre su video di test, al fine di rilevare una misura del tempo di estrazione impiegato dall'algoritmo, per effettuare una stima approssimativa sulle tempistiche dell'intero processo. E' stata notata una differenza sostanziale dipendente dalle dimensioni e dalla risoluzione dei video.
Il dataset è stato costruito catturando i video per mezzo di entrambe le fotocamere, producendo così materiale con dettagli differenti: per risoluzione, dimensione del file, numero di frame per secondo etc. Pertanto, l'applicazione del solo filtro di eliminazione del rumore su un singolo video impiega un tempo non indifferente. Il processo di eliminazione del rumore da un video a colori, con risoluzione 640 x 480 e avente dimensione di circa 30 megabyte, impiega circa un minuto per ogni secondo di riproduzione. Quindi per

chiarezza, un'estrazione di questo genere, da un video avente le stesse caratteristiche descritte sopra, ma con durata di 60 secondi impiegherebbe circa un'ora di tempo. Conseguentemente, effettuare la stessa estrazione per video catturati da fotocamere esterne, le quali con ogni probabilità producono contenuti digitali con risoluzioni più alte, fa si che il procedimento di estrazione risulti proibitivo. Nonostante ciò, alcune fotocamere hanno prodotto risultati accettabili in termini di risoluzione, per la quale è stato possibile estrarre il rumore caratteristico.

Un altro esperimento preliminare è stato confrontare il tempo di estrazione e lo spazio impiegato dal conseguente output, da video a colori e da video a scala di grigi o *grayscale*. Il risultato di tale esperimento è stato quasi scontato, in quanto l'estrazione da un video a scala di grigi risulta essere molto più leggera. C'è infatti una differenza sostanziale nella quantità di dati che devono essere processati per un video a colori ed uno in grayscale, in quanto per il secondo viene processato solo un canale, il canale Y di luminanza. Per questo motivo la decisione finale è stata di analizzare solo il canale di luminanza, quindi l'estrazione è stata portata a termine su video in grayscale.
Dopo la conclusione della fase di estrazione sono stati portati avanti ulteriori esperimenti per comprendere come confrontare le fingerprint ed approcciare così alla fase di correlazione. Sono stati studiati diversi metodi di confronto tra matrici tra i quali il calcolo del *coefficiente di correlazione di Pearson* o correlazione lineare, del quale parleremo in seguito.
Uno dei problemi principali è sorto analizzando la natura dell'output prodotto dall'estrazione e capire come applicare ad esempio una correlazione tra matrici: sappiamo che la correlazione di Pearson, ad esempio, confronta due matrici bidimensionali producendo l'indice di correlazione voluto. Le fingerprint sono rappresentate in memoria come matrici tridimensionali, dove le prime due dimensioni indicano la risoluzione e la terza è equivalente al numero di frames del video dal quale è stata estratta la stessa fingerprint. Pertanto, è stato necessario trovare un metodo per ridimensionare le fingerprint da una rappresentazione tridimensionale ad una bidimensionale. Sono

stati provati, tramite esperimenti, tre metodi principali. Il primo consiste nel calcolare la media lungo la terza dimensione della fingerprint; fissati i e j, vengono mediati i valori che si trovano nella posizione (i, j, z), dove i e j sono gli indici per le prime due dimensioni (risoluzione), iterando sulla terza dimensione z, ottenendo così una matrice bidimensionale che ha nell'elemento (i_{out}, j_{out}) la media di tutti i valori che si trovano nella stessa posizione per ogni frame. Così si ottiene una matrice con valori equalizzati ma che comunque sono rappresentativi per la fingerprint estratta in origine.
Il secondo metodo consiste nel creare una nuova matrice a partire da quella che identifica la fingerprint, componendo i frame per righe, cioè rappresentando la fingerprint affiancando i frame uno dopo l'altro. Quindi nella nuova configurazione i frame vengono concatenati uno sotto l'altro, collocandosi nelle stesse colonne ma in righe successive. Pertanto, la matrice risultante ha stesso numero di colonne delle matrici in origine e numero di righe pari al prodotto della seconda dimensione e la terza.
A causa della differente risoluzione che caratterizza i video, quest'ultimo metodo non è stato utilizzato, poiché sarebbe risultato inaffidabile e macchinoso.
Il terzo ed ultimo metodo, riguarda un procedimento per la ricerca degli indici dei fotogrammi di tipo I, utilizzando *ffprobe*, uno strumento da riga di comando per la gestione delle informazioni dei contenuti digitali. Gli indici verranno utilizzati successivamente per effettuare il calcolo della media solo sui I-Frame.

Dalla conduzione degli esperimenti preliminari è sorto un ulteriore problema riguardante lo spazio necessario al processamento di un singolo video. Infatti il tool utilizzato per la manipolazione del materiale digitale e le componenti hardware messe a disposizione, non hanno permesso il caricamento di tale quantità di dati in memoria. Come abbiamo detto in precedenza un video è rappresentato da una matrice tridimensionale che occupa una grande quantità di spazio. E' stato impossibile infatti esaminare un singolo video nella sua totalità e dunque come vedremo nella sezione dedicata alla strategia adottata per l'estrazione si è scelto di optare per una scomposizione

del materiale analizzato in più parti, per evitare la saturazione della memoria disponibile. Tali esperimenti sono stati di fondamentale importanza per comprendere alcuni funzionamenti dei tool utilizzati e rendere più agevole il lavoro di estrazione e le conseguenti fasi di manipolazione ed analisi dei dati.

2.2 Dataset

Come abbiamo già accennato, il processo di estrazione dei rumori caratteristici è stato applicato su video. Il dataset *SMART_VIDEO* originario, fornito dall'Università di Bologna, è composto da circa 520 video in formato *mp4* e *mov*. La totalità dei video è stata raccolta grazie alla collaborazione degli studenti, i quali hanno catturato tali contenuti digitali con il proprio dispositivo elettronico, quindi producendo materiale avente caratteristiche differenti per risoluzione, qualità e dimensione. Nella tabella 2.1 è mostrato l'elenco dei dispositivi in possesso degli utenti, utilizzati per catturare e popolare il dataset.

Lista dei dispositivi elettronici					
ID	Brand	Modello	Sistema operativo	Front	Rear
1	Samsung	Galaxy Core Prime	Android 5.0.2	640x480	1280x720
2	Huawei P9 Lite	Huawei VNS-L31	Android 6.0	1280x720	1920x1080
3	LG	Nexus 5x	Android 6.0	1280x720	1920x1080
4	LG	Nexus 5	Android 6.0.1	1280x720	1920x1080
5	One Plus	One Plus 3	Android 6.0.1 Oxy.	1280x720	1920x1080
6	LG	X Screen	Android 6.0.1	1920x1080	1920x1080
7	LG	Nexus 5	Android 6.0.1	1280x720	1920x1080
8	Apple	iPhone 7	10.0.3	1920x1080	3840x2160
9	Nokia	635	Win 10 Mobile	absent	1280x720
10	One plus	One Plus 3	Android 6.0.1 Oxy.	1920x1080	1920x1080
11	Motorola	G2	Android 6.0	1280x720	1280x720
12	Apple	iPhone SE	iOS 10.0.1	1280x720	3840x2160
13	Meizu	MX4	Android 5.1	1920x1080	3840x2160

Tabella 2.1: Elenco dei dispositivi

Il dataset è costituito da tredici cartelle principali, che identificano gli utenti che hanno partecipato alla raccolta dei file, numerate da 1 a 13. All'interno di ogni cartella troviamo due sotto cartelle, una denominata "front" e l'altra "rear", rispettivamente in riferimento alla fotocamera anteriore e posteriore del dispositivo. Infine all'interno di ogni cartella front ed ogni cartella rear troviamo venti video, denominati con un numero da 1 a 20 e la relativa estensione (mp4 o mov).

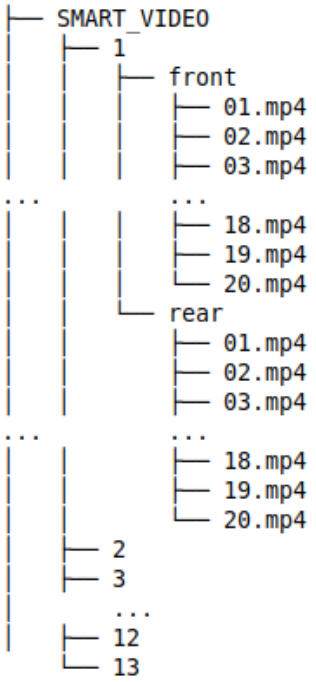

Figura 2.1: Struttura del database SMART_VIDEO

La distinzione che viene fatta, tra front e rear, è di fondamentale importanza in quanto i video sono stati catturati sia con la fotocamera frontale, sia con la fotocamera principale esterna. Tale distinzione sottolinea la presenza

di una grande differenza di risoluzione tra i video catturati con la fotocamera interna, la quale presenta nella maggior parte dei casi risoluzioni inferiori o pari a 1280×720, e la fotocamera esterna, la quale cattura video con risoluzione maggiore alla precedente. E' stata riscontrata una grande difficoltà nell'analizzare video con risoluzione maggiore a 1280×720 per via dell'elevata quantità di memoria necessaria al processamento dell'intero video. Per questo motivo è stata adottata una strategia ad hoc per il partizionamento dei dati, della quale parleremo in seguito nel paragrafo dedicato.

2.3 Procedimento di estrazione

In questo capitolo verrà presentata la strategia e gli strumenti utilizzati per estrarre il rumore caratteristico, o fingerprint, dai video presenti nel dataset.
A causa dei motivi presentati precedentemente, è stata selezionata solo una parte di materiale fornito, presente nel dataset originario. Purtroppo l'hardware messo a disposizione e le tempistiche proibitive non permettono l'analisi di video con risoluzione maggiore o pari a 1920×1080. Per questo motivo sono stati esclusi tutti i video con tale risoluzione e ovviamente quelli con risoluzione maggiore. Pertanto, i contenuti multimediali selezionati, utili ai fini dell'estrazione e della conseguente analisi, avranno risoluzione 640×480 e 1280×720.
Nonostante sia stata effettuata tale scrematura il problema non è stato risolto, ed è stato necessario adottare una strategia valida di partizionamento dei dati per evitare problemi di saturazione dello spazio in memoria. In questo capitolo verrà presentata tale strategia, verranno elencati gli algoritmi principali creati ed utilizzati per tale processo ed infine verrà analizzata l'organizzazione dell'output finale.

2.3.1 Strategia di estrazione

L'obiettivo di tale procedimento è di estratte da tutti i video selezionati i rumori caratteristici in modo tale da ottenere un processo pulito, creando l'output voluto senza nessun problema di saturazione dello spazio in memoria e garantendo tempistiche favorevoli.
Per uniformare l'output finale quanto più possibile è stato deciso di estrarre il rumore residuo solo dai primi 1500 frame di ogni video, corrispondenti approsimativamente ai primi sessanta secondi di riproduzione del file. In seguito verrà spiegata la strategia attuata che ha permesso di sviluppare gli algoritmi e script utili all'estrazione.

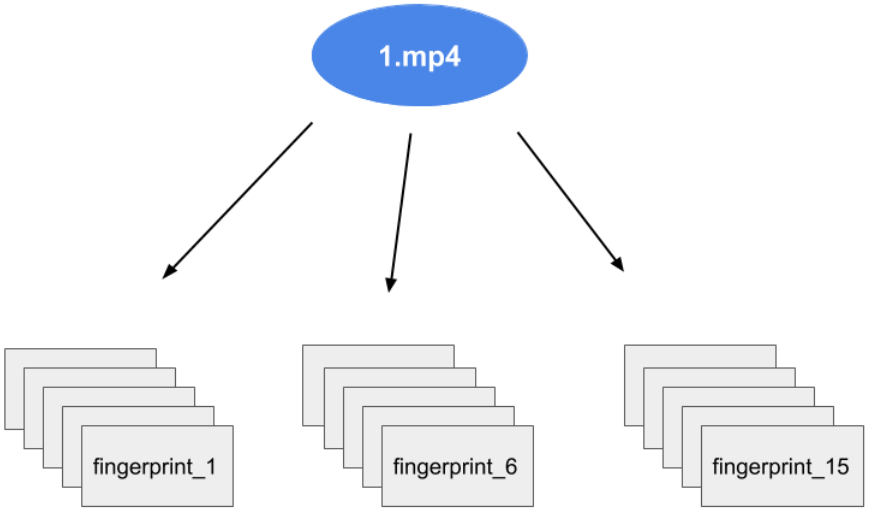

Figura 2.2: Scomposizione della fingerprint estratta da un singolo video

Come viene schematizzato in figura 2.2, il video *1.mp4* viene analizzato estraendo la fingerprint, scomposta in quindici parti, ognuna costituita da un intervallo di 100 frame del video, presi in successione. Ricordiamo che tutti

i video esaminati sono a colori.
Durante la la fase iniziale dell'estrazione vengono prelevati dal video esaminato, solo 100 frame per volta dai quali viene dapprima selezionato ed isolato il canale Y di luminanza e in una fase successiva, su quest'ultimo, viene applicato il filtro di eliminazione del rumore, o *denoising*. Così è ottenuta la fingerprint che è strettamente relativa al range di frame selezionati dal video di origine, e viene salvata nel relativo file con stesso nome ed estensione *.mat*.
L'algoritmo è stato creato in modo da eseguire un'estrazione per una singola macchina in modo da analizzare tutti i video di un particolare utente indicato, solo per video provenienti da fotocamera front o fotocamera rear, permettendo dunque l'analisi esclusiva del range di frame selezionato, così garantendo fluidità ed evitando il sovraccarico di lavoro.
Per guadagnare tempo, tramite lo sviluppo di appositi script bash, sono state avviate molteplici estrazioni eseguite contemporaneamente in parallelo. Per tale scopo l'Università di Bologna ha permesso tale strutturazione del lavoro mettendo a disposizione uno spazio disco di 100 GB e l'utilizzo da remoto delle macchine di laboratorio, in particolare quelle presenti nel laboratorio Ercolani.
Procedendo in parallelo sono stati riscontrati ulteriori problemi, in particolare nella fase di salvataggio dei dati estratti. In una fase primordiale la fase di salvataggio era stata strutturata in modo tale da salvare ogni parte di fingerprint estratta nello stesso file *.mat*, in modo da ricongiungere e quindi unificare le parti estratte, in numero di 15. L'esecuzione in parallelo di più estrazioni ha dato però problemi di accesso concorrente in lettura per tali file. Si è dunque optato per separare i dati estratti eseguendo il salvataggio delle parti estratte in file differenti. Nelle sezioni seguenti daremo una visione più chiara della struttura della cartella di output finale.

2.3.2 Cartella di progetto in remoto

Grazie allo spazio disco messo a disposizione dall'università, è stato possibile organizzare il lavoro creando una directory composta dai file necessari all'estrazione e non solo.
Nelle fasi iniziali è stato usufruito tale spazio e la potenza di calcolo per alcuni esperimenti preliminari di test. I seguenti paragrafi presentano nello specifico di quali cartelle e file si compone lo spazio di lavoro creato sulle macchine universitarie.

DB folder

La cartella *DB* contiene il database oggetto di studio. Si compone esclusivamente delle sotto cartelle, denominate con gli id degli utenti selezionati dal dataset originale. Gli utenti selezionati sono: l'utente 1, front e rear, l'utente 2 front, l'utente 4 front, il 7 solo front, il 9 solo i video rear e i video front dell'utente 11.
La stessa struttura è stata utilizzata sia per l'estrazione delle fingerprint dai video originali, sia per i video scaricati ottenuti per mezzo della routine di upload e download sulla piattaforma di messaggistica Whatsapp. La struttura della cartella DB rispecchia quella presentata nella sezione 2.2.

Fingerprint folder

La cartella *Fingerprint* è lo spazio dedicato all'output estratto. Come si può vedere in figura 2.4, la directory Fingerprint rispecchia grosso modo la struttura con la quale è organizzata la cartella DB. Infatti, contiene le sotto cartelle numerate con gli id degli utente che verranno analizzati e dal quale verrà estratta la fingerprint. Ogni utente contiene al suo interno i folder relativi al tipo di video, front o rear. E nel livello immediatamente inferiore

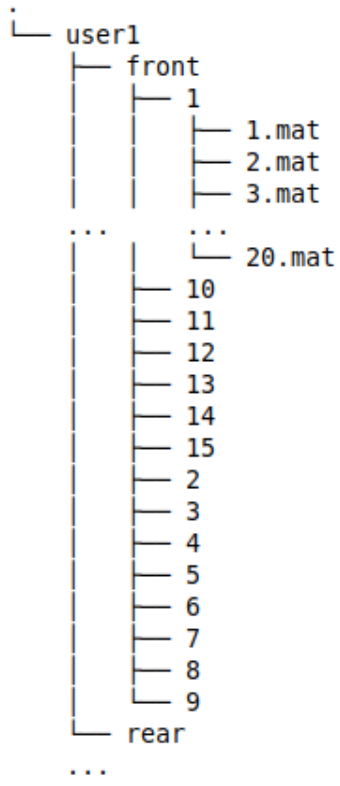

Figura 2.3: Struttura della directory Fingerprint

vi troviamo quindici sottocartelle contenenti i 20 definitivi file *.mat*, uno per ogni video dal quale è stata estratta la fingerprint parziale.
La struttura del progetto e la strategia utilizzata possono sembrare poco intuitive, ma i numerosi problemi riscontrati nelle fasi preliminari, come l'insufficienza di spazio per l'allocazione in memoria durante l'estrazione, l'insufficienza dello spazio fisico dedicato e riservato sulle macchine universitarie, ma anche le tempistiche proibitive, hanno portato alla scelta di tale strategia e conseguente struttura della cartella di lavoro, rivelandosi vincente.

Script folder

La cartella script è lo spazio adibito per tutti i programmi eseguibili necessari all'estrazione. Questo spazio è stato utilizzato anche per contenere alcuni script per esperimenti preliminari e di testing.
Al suo interno sono contenute le funzioni matlab e i file di testo di log prodotti in output alla fine di ogni esecuzione dell'algoritmo di estrazione delle fingerprint.
Vediamo adesso nello specifico il suo contenuto, analizzando gli script fondamentali.

- **Start_video_analysis.m**: Questa è la funzione matlab principale da cui ha inizio l'intero processo. Tramite l'esecuzione di questa funzione parte tutto il procedimento di estrazione delle fingerprint in base agli input che sono stati specificati per tale funzione. In particolare per eseguire la funzione *Start_video_analysis.m* bisogna specificare l'id dell'utente che si vuole processare, il tipo di video (front o rear) indicato tramite una variabile booleana, che sarà settata a 0 se si vuole attivare un'estrazione per video rear, altrimenti ad 1 se si vuole attivare un'estrazione per video front, il range di frame che si vogliono selezionare dal video dai quale estrarre la relativa fingerprint e infine un intero che sarà necessario per la formattazione dell'output.
  Di seguito viene mostrato un listato con lo sviluppo della funzione. Al suo interno si può vedere che oltre a ciò che è stato spiegato sopra, viene composto il percorso per il quale effettuare l'estrazione e viene aggiunto alla directory di lavoro anche il percorso della cartella V-BM4D, che contiene gli algoritmi e le funzioni necessarie per il procedimento di denoising, della quale cartella parleremo in seguito.

Listing 2.1: Start_video_analysis script

```
1  function [] = Start_video_analysis(user, front, ...
2                 fromFrame, toFrame, split_part)
3
4
5  path = strcat('/home/projects/videofingerprint/DB/', ...
6           num2str(user));
7
8  addpath('../VBM4D/');
9
10 Analyze_video_dataset(path, front, fromFrame, toFrame, ...
11         split_part);
12
13 end
```

- **Analyze_video_dataset.m**: Questa è la funzione più corposa del progetto e costituisce il fulcro dell'estrazione. Il file matlab *Analyze_video_dataset.m* si preoccupa di effettuare la visita della cartella DB che ospita i video, andando a selezionare i frame dei video sui quali verrà applicato il filtro di denoising. Al suo interno, oltre l'omonima funzione, ne sono contenute altre che si occupano della lettura dei video, della vera e propria fase di estrazione e conseguente salvataggio dei contenuti estratti. Viene fatta quindi una distinzione sostanziale tra video front e video rear grazie al controllo della variabile booleana relativa. La funzione *Extract_noise* si occupa di leggere il range di frame indicato dai parametri in input per mezzo della funzione *read_video_range.m*. Questa funzione è resa disponibile nel pacchetto di funzioni V-BM4D fornito dall'università Finlandese di Tampere, ma è stata modificata per adattarla agli obiettivi del progetto. I parametri in input per tale funzione sono il percorso completo del file mp4 o mov da leggere e i due numeri interi che identificano il range di frame da prelevare da tal video. Dopo aver letto il contenuto digitale, viene preso in considerazione solo il canale Y di luminanza del video e conseguentemente viene creata la

struttura dati che conterrà l'output finale. La funzione *Extract_noise* richiama per ogni video letto la funzione *Analyze_VBM4D*. A tale funzione viene dato in input, l'output della funzione *read_video_range.m*, ovvero la matrice 3D che identifica la parte di video o insieme di frame prelevati, rappresentati dal solo canale di luminanza, ed effettua la procedura di eliminazione del rumore tramite l'applicazione del filtro vbm4d. Infine, sarà sempre compito della funzione *Extract_noise* di calcolare, attraverso una differenza tra l'input iniziale e la struttura priva di rumore caratteristico, il rumore residuo finale. Nella fase finale del processo, la funzione *Save_videos* salverà i risultati estratti nei relativi file *.mat*, nella cartella Fingerprint che abbiamo sopra descritto.

- **send_mail.m**: questa funzione si occupa di notificare la terminazione del processo di estrazione che è andato a buon fine, con l'invio di una mail, riportando le informazioni più utili, quali l'id dell'utente che è stato processato, la data, il numero di contenuti digitali processati e la tipologia di video, front o rear.

VBM4D folder

Questa è la cartella originale del pacchetto **V-BM4D**, fornita e messa a disposizione dal dipartimento di Signal Processing dell'università Finlandese di Tampere, ottenibile direttamente dal sito ufficiale. Contiene le funzioni principali che identificano il filtro di denoising V-BM4D, filtro creato specificamente per l'applicazione di un processo di eliminazione di rumore caratteristico da sorgenti digitali quali video. Il pacchetto originale contiene tre file:

- **demo_denoising.m**: è uno script per l'avvio di una simulazione del processo di denoising per mezzo della funzione *vbm4d*. E' possibile modificare i parametri specificati e ottenere risultati differenti.

- **vbm4d.m**: è la funzione di denoising. Elimina il rumore caratteristico per un video, il quale può essere fornito in input con modalità differenti. Il parametro dal quale eliminare il rumore caratteristico può infatti essere una matrice 3D o una matrice 4D, rispettivamente per video in scala di grigi e video a colori.

- **read_video.m**: viene fornita anche una funzione per la lettura del file digitale sul quale applicare il filtro. Come già detto in precedenza, è stato preso spunto da questa funzione, modificando opportunamente il codice per l'adeguamento al processo sviluppato.

2.3.3 Filtro V-BM4D

Come già accennato nel paragrafo precedente il pacchetto del filtro V-BM4D si compone dei seguenti file, necessari per il procedimento di eliminazione del rumore:

- *demo_denoising.m*

- *vbm4d.m*

- *read_video.m*

Analizziamo però dal punto di vista teorico il filtro utilizzato e su quali principi e tecniche si basa. L'università Finlandese di Tampere, come presentato nell'articolo citato in bibliografia [16], propone un potente algoritmo di denoising, cioè di eliminazione di rumore da video, che sfrutta la ridondanza temporale e spaziale che caratterizza le sequenze video naturali. Infatti questi tipi di algoritmi si basano sull'idea che i pixel adiacenti di un'immagine abbiano valori simili, così anche da frame a frame in un video, i pixel che si trovano nella stessa posizione avranno valori pressoché uguali.
L'algoritmo implementa il paradigma del raggruppamento non locale e del

filtro collaborativo, in cui viene sfruttata una rappresentazione del dominio in una dimensione superiore per far si che la matrice che identifica il dato sia ancora più sparsa, così da regolarizzare maggiormente i dati. L'algoritmo proposto sfrutta la somiglianza tra i volumi spaziotemporali 3D costruiti mediante blocchi di tracciamento lungo le traiettorie definite dai vettori di movimento.

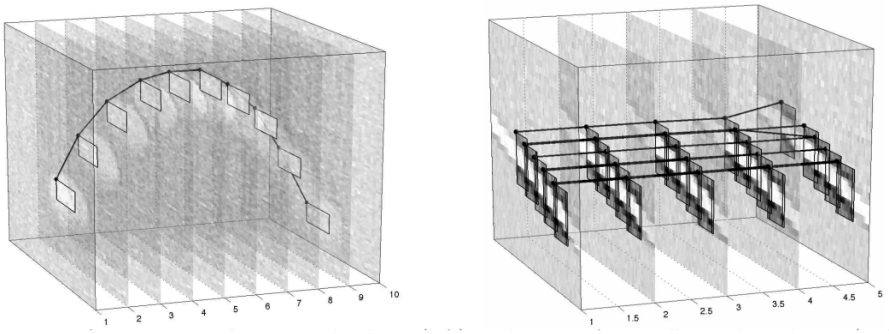

Figura 2.4: Struttura della directory Fingerprint

I volumi simili sono raggruppati concatenandoli lungo una quarta dimensione che si viene così a creare, producendo così una struttura 4D, chiamata *gruppo*. Per questo tipo di elemento esistono diversi tipi di correlazione dei dati, chiaramente identificabili lungo le dimensioni della nuova struttura:

- correlazione locale lungo le due dimensioni dei blocchi
- correlazione temporale lungo le traiettorie del movimento
- correlazione spaziale non locale (auto-correlazione) lungo la quarta dimensione

Visione generale dell'algoritmo

Considerando il fatto che l'oggetto che si va a manipolare è un video affetto da rumore, il quale può essere rappresentato come una sequenza z di

immagini rumorose, lo scopo dell'algoritmo proposto è di fornire una stima $\widehat{y}$ del video originale y attraverso il processamento dei dati osservati z. L'algoritmo V-BM4D è costituito da tre passaggi fondamentali, il raggruppamento, il filtraggio collaborativo e l'aggregazione. Questi passaggi vengono eseguiti per ogni volume spazio-temporale del video. Come descritto nell'articolo di riferimento [16], l'algoritmo è costituito da due fasi principali entrambe composte dal raggruppamento, dal filtraggio collaborativo e dalle fasi di aggregazione. Queste due fasi sono la fase Hard-thresholding e la fase Wiener filtering. Nella prima fase sono estratti i volumi dal video z affetto da rumore, e successivamente sono formati i gruppi utilizzando la misura di similarità scelta e la soglia predefinita.

Filtro Collaborativo

Questa tecnica viene utilizzata per riuscire ad ottenere predizioni su stime di cui non si è a conoscenza ed è molto utilizzata nell'ambito commerciale e nei sistemi di social networking. Nel caso dell'algoritmo V-BM4D il filtraggio collaborativo si realizza trasformando ciascun *gruppo* di cui sopra, attraverso una trasformata lineare separabile 4D decorrelante e quindi mediante una trasformazione in contrazione ed una trasformazione inversa. In questo modo, il filtro collaborativo fornisce stime per ciascun volume 3D aggregato nel gruppo, che viene quindi restituito e aggregato in modo adattativo alla posizione originale nel video. I risultati sperimentali in [16] dimostrano l'efficacia della procedura.

2.3.4 Cartella di progetto in locale

Oltre alla directory di lavoro messa a disposizione dall'Università di Bologna, è stato necessario organizzare il tutto in una directory locale. Questa directory ha ospitato il materiale necessario per gli esperimenti preliminari, per il testing delle estrazioni e per il processo di estrazione finale. I file più importanti sono gli script bash sviluppati per l'esecuzione delle estrazioni da

remoto.
Le estrazioni delle fingerprint, infatti, sono state effettuate, come già detto in precedenza, sulle diverse macchine ubicate nel laboratorio Ercolani del Dipartimento di Informatica a Bologna. Per iniziare l'intero processo di estrazione è stato effettuato l'accesso da remoto su tali macchine, tramite il protocollo *ssh*, così è stato possibile usufruire del tool di sviluppo preinstallato MATLAB sui computer universitari. L'esecuzione cosi viene inizializzata grazie allo script bash mostrato in appendice alla sezione A.1.

2.3.5 Routine sul social Whatsapp

Nonostante il processo di source camera identification sia ormai un obiettivo provato e consolidato, c'è un'altra sfida che si prefigge questo particolare ambito della ricerca. Oggigiorno oltre alla spropositata diffusione di dispositivi elettronici, in particolare gli smartphone, che ci circondano e costituiscono quasi un'appendice del nostro corpo, un altro tipo di entità digitale che ha interconnesso tutto il mondo negli ultimi anni è il social network.
I Social Network, come Facebook e Twitter, forniscono una strumento importante per lo scambio di informazioni e contenuti digitali e talvolta possono fornire un supporto per le investigazioni e per ricostruire fatti sulla base delle informazioni contenute in un profilo personale e associarlo con un account specifico. E' noto che molte attività criminali come la pornografia infantile, truffe e terrorismo proliferano mischiando tali contenuti digitali. Generalmente infatti tali attività sono svolte per mezzo di profili falsi o anonimi, sarebbe quindi molto interessante intercettare un video che è stato postato da un profilo sconosciuto, utilizzato per propositi illegali, e determinare se tale video è stato generato dalla stessa fotocamera di un altro video postato da un profilo di cui si conosce l'identità sullo stesso Social Network.
Caricare un video su Social Network altera le caratteristiche del contenuto, in particolare esso viene modificato per risoluzione, per dimensione e a volte anche ridimensionando la sua durata. La sfida che ci si propone quindi è

di capire se dopo tale pesante compressione effettuata dai Social Network è possibile determinare se due video provengono dalla stessa fotocamera.
Anche l'applicazione di messaggistica istantanea Whatsapp può essere considerata un Social Network e di grande rilievo per il nostro proposito, visto il largo utilizzo nella società e l'abnorme quantità di contenuti digitali che ogni giorno transitano per mezzo della stessa.
Ci sono più metodi di effettuare un uploading di un contenuto multimediale sulla piattaforma in questione: registrando un video direttamente all'interno dell'app tramite lo strumento di registrazione built-in che fornisce Whatsapp, da smartphone selezionando un contenuto dalla galleria, oppure effettuando un drag and drop del file tramite l'applicazione web. E' stato effettuato quindi il caricamento di tutti i video dai quali è stata estratta la fingerprint. Per semplificare la computazione è stato effettuato tale caricamento solo sul Social Network presentato in precedenza, sul quale sono stati effettuati i 140 upload per caricare i video presi in considerazione per il processo.
Da notare che il caricamento tramite applicazione web non permette l'upload di file più grandi di 64 megabyte, e proprio perché la maggior parte dei video supera tale dimensione è stato scelto di caricare l'intera collezione di video tramite smartphone. Una volta caricati tali video sono stati nuovamente scaricati dalla versione web di WhatsApp e collocati con la corretta disposizione all'interno del database in locale. Whatsapp effettua un'importante compressione su video, riportando il video ad una risoluzione 640 x 352. Queste dimensioni hanno agevoltato sia la manovra di downloading dei video, sia la successiva fase di estrazione delle fingerprint.

2.3.6 Output finale

Discutiamo in questa sezione dell'output finale prodotto dal procedimento di estrazione. Il primo tipo di file prodotto è il file testuale di log, localizzato nella cartella "script" per facilità. Com'è possibile osservare nello listato 2.2, ogni comando per l'esecuzione di un'estrazione su una determinata macchina

remota produce un output testuale. Tramite questi file di log è possibile monitorare l'esecuzione delle estrazioni e controllare, nel caso una di esse non venga portata a termine, la causa dell'errore.

$$name : user9\_front\_1101 - 1200.txt$$

Ogni file testuale di log riporta come si può osservare sopra, il nome dell'utente per il quale effettuare l'estrazione, la tipologia di video in questione (front o rear) e il range di frame selezionati.
Il file principale di output però è costituito dalle stesse fingerprint organizzate, come già spiegato, nella cartella Fingerprint.

Confronto di dimensioni - 100 frame						
ID	Utente	Tipologia	Risoluzione	Camera	Dim	N.Frame
1.mp4	1	original	640×480	front	26 MB	100
1.mp4	1	original	1280×720	rear	303 MB	100
1.mp4	9	original	1280×720	rear	290 MB	100
1.mp4	1	downloaded	640×352	front	26 MB	100
1.mp4	1	downloaded	640×352	rear	73 MB	100
1.mp4	9	downloaded	640×352	rear	71 MB	100

Tabella 2.2: Dimensione dei file di output parziali

Nella tabella di cui sopra sono schematizzate le dimensioni delle fingerprint in base alla caratteristiche dei video da cui sono state estratte. Un video è caratterizzato da diversi parametri: la risoluzione del video, quindi la tipologia della fotocamera che l'ha catturato, il numero di frame presi in considerazione, la tipologia, ovvero se esso sia originale oppure scaricato dalla piattaforma di messaggistica presa in considerazione, giocano un ruolo fondamentale nella dimensione spaziale finale del file *.mat* che contiene la fingerprint estratta. Come si può vedere dalle osservazioni riportate in tabella un video che ha risoluzione maggiore peserà sicuramente molto più di uno che ha risoluzione inferiore. Questa osservazione è strettamente legata

alla natura della camera che ha catturato il video. Le fotocamere frontali solitamente producono contenuti digitali con una qualità inferiore di quelle poste nel retro del dispositivo. Inoltre possiamo renderci conto che un video che ha subito la routine di uploading e downloading tramite l'applicazione WhatsApp subirà un'importante compressione, come spiegato nel paragrafo 2.3.5, presentando quindi dimensioni decisamente inferiori. Da notare che le fingerprint, come analizzato nel paragrafo 2.3.1, sono suddivise in quindici parti ognuna da 100 frame. Quindi, facendo un calcolo approssimativo, compattando tutte e quindici le parti che vengono estratte dai video elencati nella tabella 2.2 otterremo le dimensioni elencati nella seguente tabella.

Confronto di dimensioni - 1500 frame						
ID	Utente	Tipologia	Risoluzione	Camera	Dim	N.Frame
1.mp4	1	originali	640×480	front	390 MB	1500
1.mp4	1	original	1280×720	rear	4545 MB	1500
1.mp4	9	original	1280×720	rear	4350 MB	1500
1.mp4	1	downloaded	640×352	front	390 MB	1500
1.mp4	1	downloaded	640×352	rear	73 MB	1500
1.mp4	9	downloaded	640×352	rear	71 MB	1500

Tabella 2.3: Dimensione dei file di output totali

Come è facilmente deducibile dalle dimensioni calcolate nella tabella 2.3, risulta proibitivo confrontare, ai fini dell'analisi finale, i dati così estratti. Per questo motivo i dati verranno mediati ottenendo così i file finali sui quali effettuare i confronti e le correlazioni.

Capitolo 3

Manipolazione sui risultati estratti

Dopo la lunga fase di estrazione delle fingerprint, ci dedichiamo in questo capitolo alla descrizione della fase di manipolazione dei dati estratti. La dimensione di tali dati come mostrato nelle tabelle 2.2 e 2.3 lascia trasparire l'impossibilità di lavorare in maniera agile.
Un'altra questione da risolvere è legata al confronto finale che dovrà essere effettuato per analizzare i risultati e trarne delle conclusioni. Il metodo scelto per confrontare le fingerprint è la correlazione lineare, di cui parleremo in seguito, pertanto occorre che le due strutture confrontate siano bidimensionali e che abbiano la stessa dimensione.
Di conseguenza il vincolo imposto dal metodo di correlazione ha portato alla scelta di ridimensionare tutti i dati grezzi estratti, da una struttura tridimensionale, ad una bidimensionale. Inoltre è necessario applicare il ridimensionamento facendo in modo che l'informazione contenuta dalle fingerprint non venga alterata, ottenendo così un nuovo insieme di dati che potrà essere analizzato più facilmente, mantenendo così le proprietà caratteristiche. Nei paragrafi di cui sotto presenteremo i due metodi di ridimensionamento, basati sul calcolo della media.

3.1 Ricombinazione delle fingerprint

La strategia applicata per la fase di estrazione ha reso la computazione meno pensante riuscendo a suddividere il processo, eseguendo in parallelo diverse istanze della stessa funzione di estrazione, riducendo notevolmente i tempi dell'intero processo. Tale strategia ha però un contro: le fingerprint vengono suddivise in blocchi da 15, implicando un'obbligatoria riunificazione in un singolo dato. Le quindici parti in cui è stata suddivisa la fingerprint dovranno essere quindi impilate in un unica struttura mantenendo l'ordine prescritto. Per questo è stato necessario sviluppare un algoritmo di visita della directory che ospita le fingerprint estratte, in modo da compattare le quindici matrici tridimensionali in una, chiaramente mantenendo la numerazione dei blocchi in modo tale da rispettare l'ordine delle fingerprint estratte dal relativo video.
Durante la procedura di ricombinazione viene direttamente effettuato il calcolo di media. Vedremo nei prossimi paragrafi come questo avviene nello specifico.

3.1.1 Calcolo della media su tutti i frame

La ricombinazione delle fingerprint viene fatta in concomitanza al calcolo della media. L'algoritmo sviluppato effettua una visita della directory, per un particolare utente, per il quale si vogliono compattare le fingerprint. Vengono così immagazzinate le quindi parti in una variabile temporanea e viene quindi calcolata immediatamente la media sulla terza dimensione. Il listato di cui sotto riporta le righe che mostrano quanto spiegato precedentemente. E' possibile visionare l'intera funzione MATLAB che contiene tale codice in appendice B alla sezione B.1.1.
In riga 1 viene letta la parte di fingerprint dal file *.mat*, che ricordiamo avrà una struttura tridimensionale con terza dimensione pari a 100. Alla riga successiva notiamo il contatore *frames*, che tiene conto della somma della terza dimensione di ogni ogni blocco di fingerprint caricato. Tale calcolo

viene fatto per ottenere l'intero che costituirà il denominatore nel calcolo della media. Nella riga 3 viene effettuata la concatenazione tra la variabile temporanea *tmp_fp* e la nuova fingerprint parziale e constestualmente viene effettuta la somma di questa matrice sulla terza dimensione. Si intende che verrà sommato ogni valore che si trova nella posizione (i, j, z) iterando sul valore z, per ogni valore fissato di i e j.

Listing 3.1: calcolo media

```
1 fp = load(mat_name, fp_name);
2 frames = frames + size(fp.(fp_name), 3);
3 tmp_fp = sum(cat(3,tmp_fp, fp.(fp_name)),3 );
```

3.1.2 Calcolo della media su I-Frame

Esistono diversi algoritmi per la compressione di un file video, ciascuno con vantaggi e svantaggi nel livello di compressione finale e nella composizione dei fotogrammi. Nella maggior parte degli algoritmi ritroviamo tre tipi di fotogrammi:

- I-Frame: Intra-coded frames
- P-Frame: Forward predicted frames
- B-Frame: Bi-directional predicted frames

Un I-frame, dall'inglese 'Intra-coded frame', consiste in una immagine completamente descritta, allo stesso modo di un file immagine non in movimento. Esso è meno compresso e non richiede in fase di decompressione di altri fotogrammi per essere decodificato. I P-frame e i B-frame contengono invece solo parte dell'informazione sull'immagine e di conseguenza necessitano di meno spazio su disco rispetto i I-frame. Ciò migliora il rapporto di compressione. Nonostante tale vantaggio, una maggiore compressione significa minore accuratezza per una futura stima del PRNU, eliminando una

quantità significante di dettagli che potrebbero far ottenere valori migliori della stessa stima.
Un P-frame contiene solo la parte dell'immagine che è cambiata rispetto al fotogramma precedente. Ad esempio, in una scena in cui un oggetto si muove su uno sfondo fisso, solo il movimento di tale oggetto necessita di essere codificato. Il codificatore non ha bisogno di scrivere in memoria i pixel che riguardano lo sfondo fisso, i quali non hanno subito modifiche, risparmiando in termini di spazio. Dunque i fotogrammi di tipo P utilizzano dati provenienti dal fotogramma precedente in fase di decompressione.
Un B-frame risparmia ancora più spazio sfruttando la differenza del fotogramma corrente rispetto sia al fotogramma precedente che a quello successivo, ottenendo un livello più alto di compressione.
Da questa breve introduzione ai tipi di fotogrammi, e per come viene studiato in [18], possiamo quindi concludere che i I-Frame sono più affidabili e adatti per effettuare uno studio sull'effetto di compressione video per un processo di identificazione della telecamera sorgente basato sul PRNU. L'idea che sta alla base di questa scelta si fonda sul fatto che i I-frame possono produrre una fingerprint più affidabile rispetto a i P-frame e B-frame. Per questi motivi è stato scelto di valutare un altro approccio per la stima delle firme finali, cioè considerare nel calcolo della media la fingerprint estratta dai I-frame del video, escludendo gli altri tipi di fotogrammi. Come mostrato nel paragrafo B.1.2, ma come si può anche osservare nel listato seguente, viene effettuato un filtraggio sulla fingerprint già estratta, per mezzo degli indici che identificano la posizione dei I-frame nella successione di tutti i frame del video. Tali indici vengono preventivamente calcolati a partire dal video in questione, tramite la funzione *getIFrameIndices*, consultabile in appendice B al paragrafo B.1.3. Su tale struttura verrà effettuata la somma sui valori lungo la terza dimensione, così da ottenere una struttura bidimensionale, ed infine verrà calcolata la media dividendo per il numero di I-frame, *IF_count*.

Listing 3.2: calcolo media

```
1 IF_indices = getIFrameIndices(pathToScript, video_path);
2 IF_count = size(IF_indices,2);
3 fp_iframe = fp.(fp_name)(:,:,IF_indices{i-2}(:));
4 % calculating sum on third dimension.
5 tmp_fp = sum(cat(3,tmp_fp, fp_iframe),3 );
```

3.2 Correlazione di Pearson

L'indice di correlazione di Pearson, detto anche detto coefficiente di correlazione lineare tra due variabili statistiche è un indice che esprime un'eventuale relazione di linearità tra esse. Date due variabili statistiche X e Y, il loro indice di correlazione di Pearson è definito come la loro covarianza divisa per il prodotto delle deviazioni standard delle due variabili:

$$\rho_{X,Y} = \frac{\sigma XY}{\sigma X \sigma Y}$$

La covarianza in statistica è un numero che fornisce una misura di quanto due variabili aleatorie varino assieme, ovvero della loro dipendenza. La formula è la seguente:

$$Cov(X,Y) = \mathbb{E}\big[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\big]$$

, dove $\mathbb{E}$ è il valore atteso o media ponderata.

Effetuando qualche passaggio di sostituzione si ottiene la formula completa:

$$\rho_{X,Y} = \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{\mathbb{E}[X^2] - [\mathbb{E}[X]]^2}\sqrt{\mathbb{E}[Y^2] - [\mathbb{E}[Y]]^2}}$$

Il coefficiente ρ assume sempre valori compresi tra -1 e 1. Per cui, quanto più vicino ad 1 sarà il valore di correlazione tra due variabili statistiche tanto

più esse saranno simili tra loro. Per valori di correlazione inferiori a 0.7 si parla di *correlazione moderata o debole* se il suo valore si avvicino allo zero, altrimenti per valori superiori si dice *correlazione forte*. Ci si aspetta quindi, che i valori di correlazione che verranno calcolati e discussi nella fase di analisi si collochino in quest'ultimo range di valori, quanto più prossimi al valore 1.

3.2.1 Ridimensionamento delle fingerprint mediate

L'insieme di fingerprint ottenuto presenta una disuguaglianza in dimensione. Questo è dovuto al fatto che i contenuti digitali originali, dai quali sono state estratte le fingerprint finali, hanno una risoluzione variabile, come detto nei paragrafi precedenti, da 640×480 a 1280×720.
Inoltre il passaggio dei video sulla piattaforma di messaggistica Whatsapp e il conseguente scaricamento, ha denotato la presenza di un'altra risoluzione, prodotta dalla compressione, di dimensioni 640×352. La fase di analisi finale porrà attenzione ai confronti effettuati per mezzo della correlazione, che sappiamo essere un'operazione che implica il confronto di due variabili, nella fattispecie due matrici bidimensionali. Data questa diversità tra i dati da analizzare e visto che per correlare due matrici bidimensionali è necessario che esse abbiano la stessa dimensione, bisogna utilizzare un metodo per uniformare tali diversità. Necessità quindi un metodo di ridimensionamento delle matrici. Sono stati fatti degli esperimenti per effettuare tale ridimensionamento, utilizzando quattro criteri principali:

- effettuando un ridimensionamento, con interpolazione bicubica, della matrice bidimensionale con risoluzione inferiore riportando le dimensioni a quelle della matrice con risoluzione superiore;
- effettuando un ridimensionamento, con interpolazione bicubica, della matrice bidimensionale con risoluzione superiore riportando le dimensioni a quelle della matrice con risoluzione inferiori;

- effettuando un ridimensionamento, con interpolazione lineare, dalla dimensione superiore alla dimensione inferiore;

- effettuando un ridimensionamento, con interpolazione lineare, dalla dimensione inferiore alla dimensione superiore;

Il ridimensionamento nei primi due casi viene implementato tramite la funzione MATLAB *imresize* che utilizza di default un'interpolazione bicubica. Negli altri due casi viene creato un oggetto interpolante tramite la funzione *griddedInterpolant* tramite il quale è possibile ridimensionare le dimensioni di una matrice applicando un procedimento di interpolazione lineare. Dopo il ridimensionamento, le rispettive matrici aventi ora le stesse dimensioni sono correlate con la funzione MATLAB *corr2*. Infine, sono state confrontate le correlazioni prodotte mettendo a paragone le matrici ottenute applicando i criteri di cui sopra, al fine di stabilire quale sia il metodo migliore di ridimensionamento di una matrice.

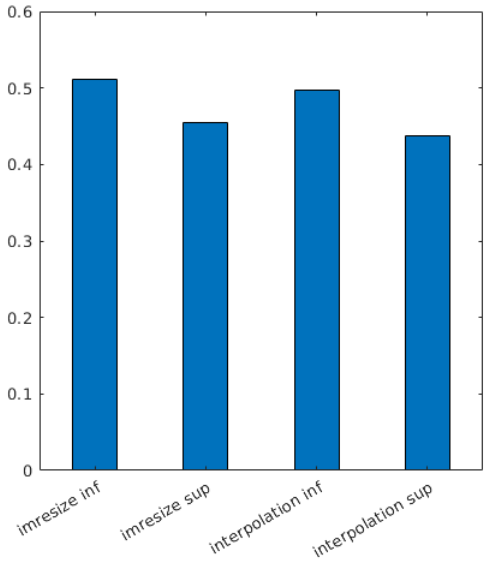

Figura 3.1: Grafico per il confronto delle metriche per il ridimensionamento delle fingerprint

Il ridimensionamento verso una dimensione maggiore utilizzando la funzione *griddedInterpolant*, non ha portato risultati soddisfacenti. Il miglior risultato è stato ottenuto utilizzando la funzione *imresize* ridimensionando la matrice ad una risoluzione inferiore. La funzione di interpolazione bicubica implementata in *imresize* è migliore della funzione di interpolazione lineare. Dunque i futuri ridimensionamenti preliminari alle correlazioni saranno effettuati per mezzo della funzione *imresize* per un ridimensionamento da una risoluzione superiore ad una inferiore.

Capitolo 4

Analisi dei risultati: correlazione tra le fingerprint

In questo capitolo verranno analizzati i dati estratti, effettuando più confronti tra diversi gruppi di fingerprint. L'obiettivo che si vuole perseguire è di ritrovare quante più possibili affinità tra le fingerprint che provengono da video catturati con la stessa fotocamera.
Il processo di estrazione ci permette di ottenere il rumore caratteristico, basandoci sul Photo Response Non-Uniformity. Come spiegato nel capitolo iniziale, tale rumore può essere considerato come la stessa fingerprint e risultare quindi identificativo per la fotocamera con la quale è stato catturato il video. Ci si aspetta che le fingerprint provenienti da video catturati con la stessa fotocamera abbiano un alto indice di similarità, mentre le fingerprint estratte da video ottenuti tramite strumenti di cattura differenti abbiano un basso indice di similarità.
Inoltre, ci si aspetta che questo rapporto di similarità sia preservato con tutta probabilità anche dopo il caricamento online su un social network o una piattaforma di messaggistica.
I confronti di similarità sono stati effettuati, dopo aver portato avanti lo studio sulle diverse tipologie di fotogrammi, anche sulle fingerprint calcolate per mezzo dei I-Frame, i quali dovrebbero apportare migliorie identificative

facendo inoltre risparmiare una grande quantità di spazio e di tempo per la rappresentazione e la precedente manipolazione dei dati finali.
Si vuole studiare in particolare la distribuzione che si ottiene mettendo in correlazione le fingerprint derivanti da una specifica fotocamera con quelle derivanti da tutti gli altri dispositivi di cattura che caratterizzano il dataset. Le immagini che verranno mostrate nelle pagine a seguire mostrano gruppi di grafici rappresentanti gli istogrammi dei valori di correlazione ottenuti come spiegato in precedenza.
Per mettere a paragone la similarità tra le diverse coppie di fingerprint è stata utilizzata, come descritto al paragrafo 3.2, la correlazione lineare o coefficiente di correlazione di Pearson.
Al paragrafo 2.2 abbiamo discusso della conformazione del dataset e del quantitativo di utenti che hanno partecipato al suo popolamento, in numero di tredici. Ogni utente ha catturato i video sia con la fotocamera anteriore, sia con la fotocamera posteriore, quindi le fotocamere effettive sono ventisei. Da questa ultima considerazione capiamo che ogni strumento fisico di cattura di contenuti digitali costituisce un'entità a parte, che ha caratteristiche differenti da qualsiasi altro strumento di cattura. Quindi, ciò vale anche per due fotocamere appartenenti allo stesso dispositivo elettronico, per le quali si noterà una dissimilarità tra la fingerprint estratta da un video catturato da una fotocamera anteriore e una estratta dalla fotocamera posteriore. Purtroppo, come già anticipato, per motivi di dimensione dei file da analizzare e per problematiche relative alle tempistiche, non è stato possibile analizzare la totalità dei video presenti nel dataset. Sono stati analizzati solo i video provenienti da sette fotocamere differenti. Nella seguente tabella vengono riportati gli utenti e le relative fotocamere, i cui video hanno preso parte attiva nella fase di estrazione.

ID	Brand	Modello	Sistema	Tipo fotocamera
1	Samsung	Galaxy Core Prime	Android 5.0.2	Anteriore
1	Samsung	Galaxy Core Prime	Android 5.0.2	Posteriore
2	Huawei P9 Lite	Huawei VNS-L31	Android 6.0	Anteriore
4	LG	Nexus 5	Android 6.0.1	Anteriore
7	LG	Nexus 5	Android 6.0.1	Anteriore
9	Nokia	635	Win 10 Mobile	Posteriore
11	Motorola	G2	Android 6.0	Anteriore

Tabella 4.1: Proiezione degli utenti e le fotocamere utilizzate

4.1 Correlazioni: istogrammi

Dalla tabella 4.1, si può osservare che la fase di analisi pone attenzione sulle fingerprint estratte dai video catturati tramite le fotocamere elencate, in numero di 7. Ricordiamo che i video catturati da ogni fotocamera sono venti, per un totale di 140 fingerprint. Per garantire l'esistenza di insiemi disgiunti sui quali effettuare analisi corrette, le fingerprint sono state suddivise in due gruppi: in ordine, le prime dieci fingerprint costituiranno le osservazioni provenienti dai video originali e il secondo gruppo di dieci fingerprint costituirà le osservazioni provenienti dai video che hanno subito la routine di upload e download sulla piattaforma WhatsApp. Otteniamo così due gruppi disgiunti di 70 fingerprint ciascuna.
Come primo obiettivo, ci si pone di correlare le fingerprint di una fotocamera con tutte le altre, ottenendo così 700 valori di correlazione. Questo procedimento viene effettuato a turno per ogni fotocamera, ottenendo così sette grafici differenti. La prima immagine 4.1, mostra tali grafici per le fingerprint estratte da video originali e ottenute tramite una media di tutti i frame del video. Lo stesso procedimento di visualizzazione è stato portato avanti per le fingerprint provenienti da video scaricati tramite WhatsApp, e mediate in entrambi i metodi presentati nei paragrafi 3.1.1 e 3.1.2, ottenendo così altre tre immagini per un totale di 28 istogrammi. Per ogni grafico, i valori sulle ascisse costituiscono i valori di correlazione, che sappiamo essere tra -1 e $+1$, invece i valori sulle ordinate rappresentano il numero di corre-

lazioni di uno specifico valore. Come sperimentato in [19], ogni istogramma dovrebbe presentare due distribuzioni ben definite e riconoscibili. E come si può notare nella maggior parte degli istogrammi nell'immagine 4.1, come ad esempio l'istogramma denominato "*User 1 Rear*", si presenta questa particolare conformazione. La distribuzione di valori che si trova sulla parte sinistra dell'istogramma rappresenta l'insieme delle correlazioni delle coppie di fingerprint che non provengono dalla stessa sorgente; la distribuzione di valori che si trova a destra, ma che si nota con maggiore difficoltà, rappresenta l'insieme delle coppie di fingerprint che provengono dalla stessa fotocamera. Si potrebbe quindi decidere di fissare un valore di soglia σ, per ogni istogramma, in modo da discriminare quale sia il giusto valore per ottenere una buona fingerprint. Quindi sottoponendo un video per un test di identificazione della fotocamera sorgente, estraendo da esso la relativa fingerprint e correlandola con un dataset costruito ad hoc, si potrebbe capire se esso viene riconosciuto come appartenente ad una sorgente già conosciuta o meno, confrontando i valori di correlazione con la soglia precedentemente stabilità per la specifica fotocamera. Se i valori di correlazione superano la soglia determinata in precedenza, allora il video viene identificato come proveniente da una fotocamera conosciuta. Come si può osservare nei diversi istogrammi, le due distribuzioni non si comportano sempre allo stesso modo, proprio per questo motivo per ogni utente deve essere scelto un valore di soglia differente, effettuando prima tale studio sulla curva dell'istogramma.

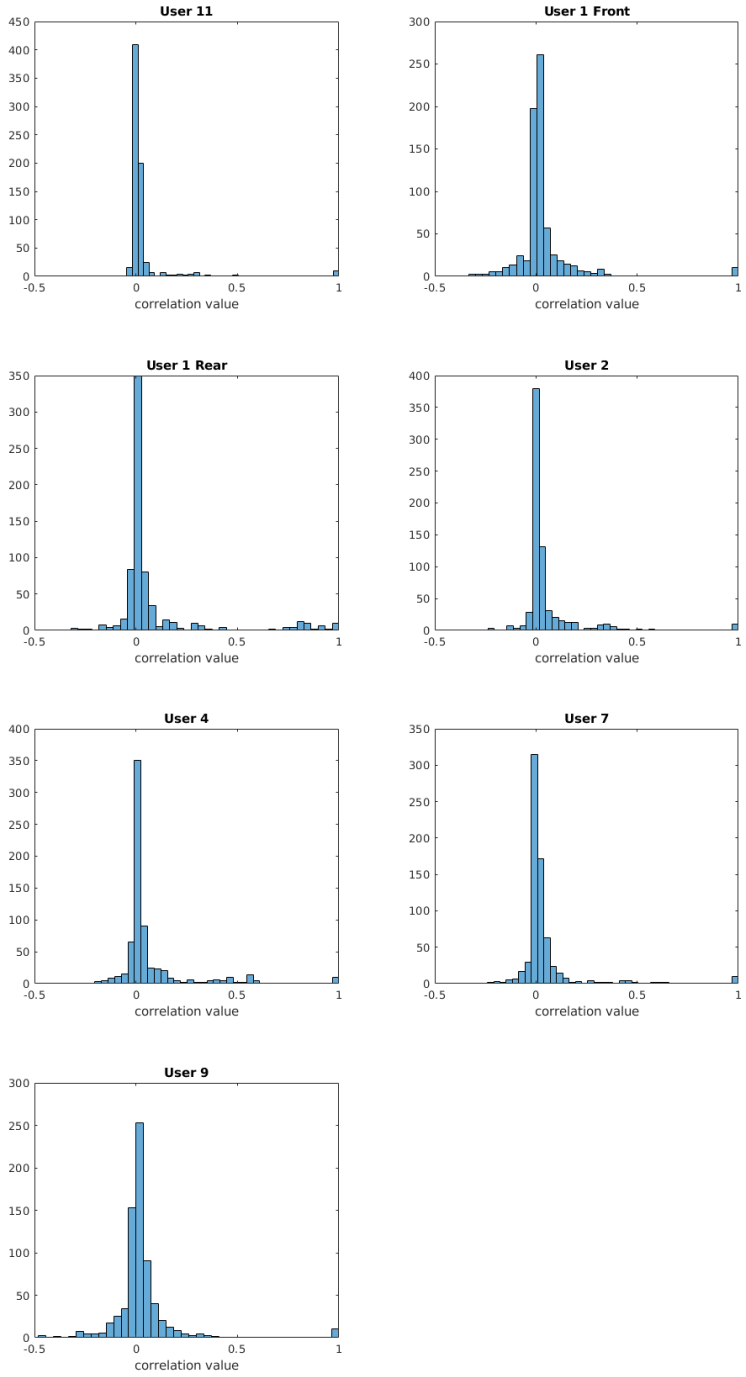

Figura 4.1: Istogrammi - correlazioni fingerprint originali

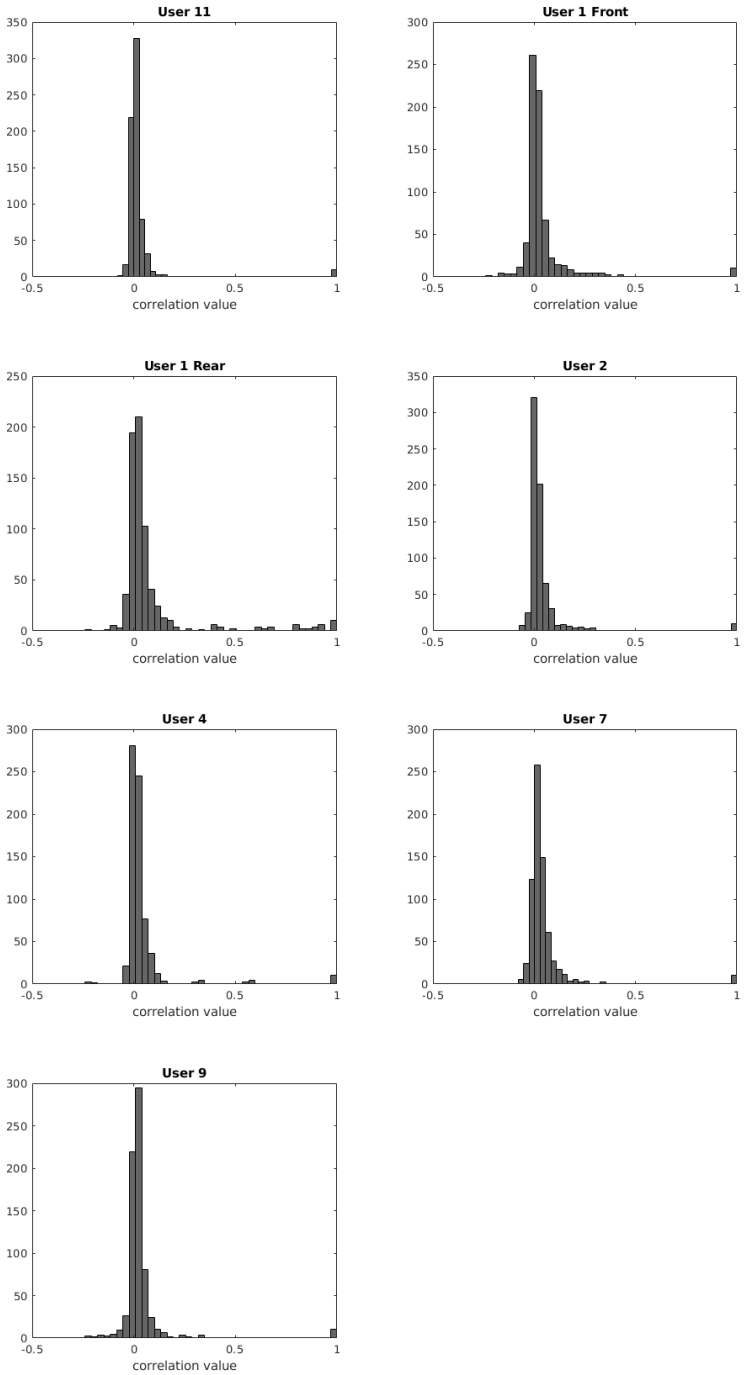

Figura 4.2: Istogrammi - correlazioni fingerprint scaricate

Per quanto riguarda i confronti di correlazione tra fingerprint ottenute mediando solo i frame di tipo I del video, riassunti negli istogrammi delle immagini 4.3 e 4.4, non hanno portato buoni risultati.
Risulta che le due distribuzioni sono poco evidenti e i valori di correlazione che si ottengono negli istogrammi delle prime due immagini, vengono a mancare in questa nuova configurazione, non rispecchiando il valore atteso. Gli I-Frame costituiscono i fotogrammi che contengono più informazione e ci si aspettava che applicare un'analisi scartando tutto il resto delle informazioni, bastasse per ottenere risultati migliori o accettabili. Nonostante ciò, essi non bastano per la costruzione di una buona fingerprint. La spiegazione può risiedere, appunto, nel quantitativo di informazione che viene apportato dall'intero insieme di frame di un video analizzato, rendendo la firma più solida e identificativa.

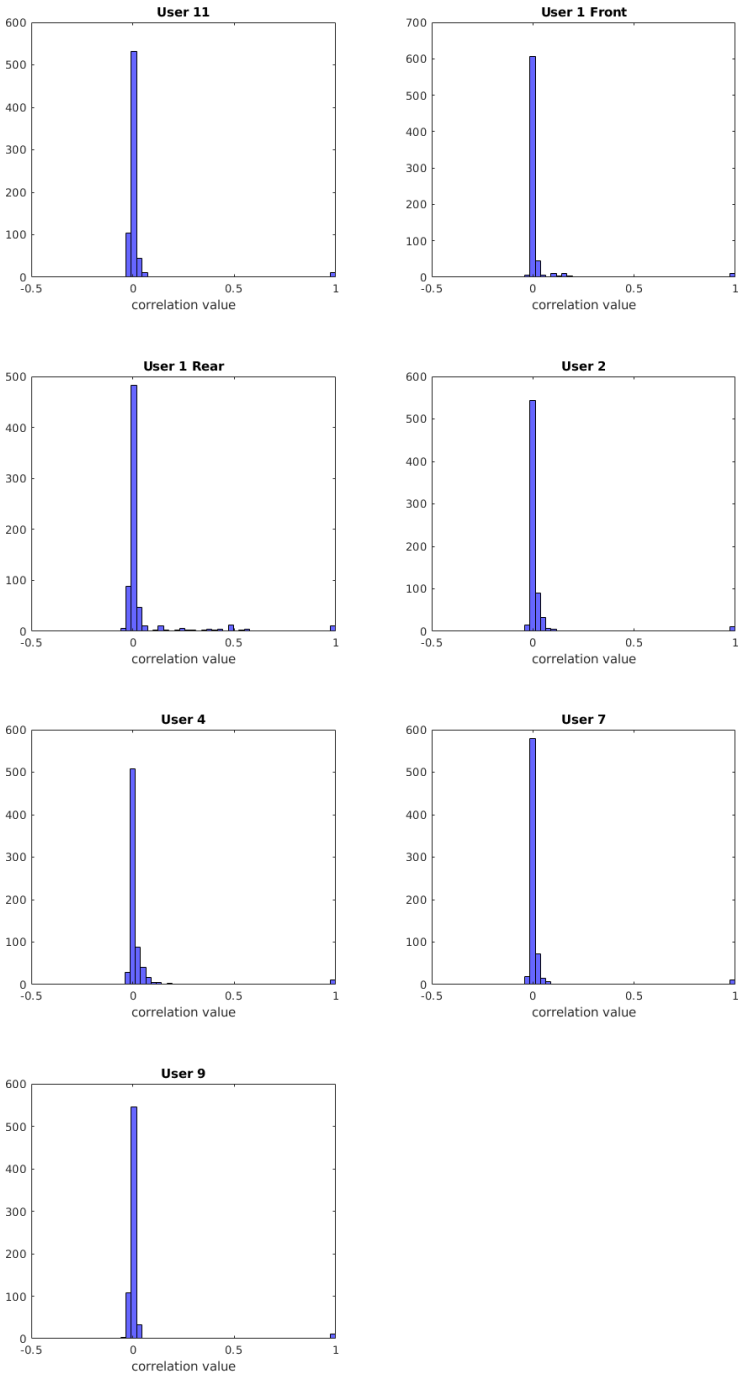

Figura 4.3: Istogrammi - correlazioni fingerprint originali solo I-Frame

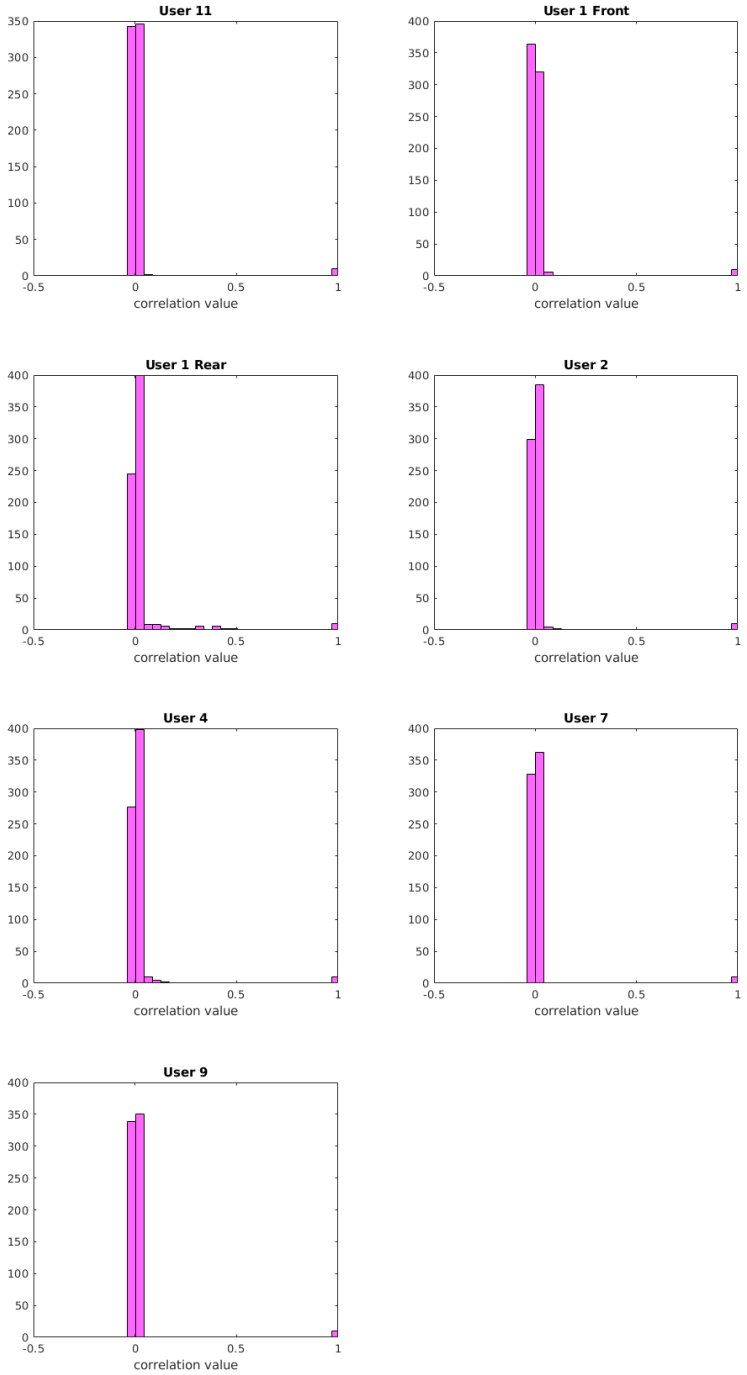

Figura 4.4: Istogrammi - correlazioni fingerprint scaricate solo I-Frame

4.2 Correlazioni: confronto totale tra le fingerprint

Per avere una visione più generale dei confronti tra le coppie di fingerprint, in questo paragrafo verranno mostrati alcuni grafici che mostrano una visione più ampia e totale dell'intero gruppo di correlazioni.
Le immagini che si trovano in questa sezione, mostrano quattro grafici rappresentanti rispettivamente quattro insiemi di correlazioni ben determinati; i grafici denominati *"Correlazioni originali - normal mean"* e *"Correlazioni scaricati - normal mean"* per le fingerprint ottenute da video originali e scaricati, calcolate con la media su tutti i frame; i grafici denominati *"Correlazioni originali - I-Frame"* e *"Correlazioni scaricati - I-Frame"* per le fingerprint ottenute da video originali e scaricati, calcolando la media solo sui fotogrammi di tipo I.
Si possono così visionare le correlazioni di tutte le possibili coppie tra le 70 fingerprint provenienti da video originali e le 70 fingerprint provenienti da video scaricati. Come si può osservare, i valori presenti nelle ascisse e nelle ordinate corrispondono alle stesse fingerprint. In particolare in figura 4.7, sono prese in esame le 70 fingerprint provenienti da video originali, calcolate effettuando una media su tutti i frame, le quali vengono correlate a coppie, producendo i risultati dispiegati nel grafico, ottenendo quindi un'insieme di 4900 correlazioni.
Dunque, il grafico rappresenta una matrice quadrata , dove l'elemento (i, j) è il coefficiente di correlazione di Pearson tra la fingerprint in riga i e la fingerprint in colonna j. La correlazione in posizione (i, j), è uguale in questo caso alla correlazione in posizione (j, i), mentre sulla diagonale principale ritroviamo i valori di correlazione delle fingerprint con se stesse, con valori pari a 1.
Pertanto, in ogni riquadro del grafico, ricadranno un totale di 100 correlazioni, ottenute dal confronto tra le 10 fingerprint relative ad una particolare fotocamera, con altre 10 appartenenti ad un'altra sorgente.

Nella legenda sono segnalati i valori di correlazione per mezzo di una colorazione che rende più semplice la lettura del grafico. Valori di correlazione compresi tra 0.7 e 1 saranno segnalati con un colore caldo, tra l'arancione e il giallo, valori medi saranno identificati con colori più chiari tendenti al verde, ed infine valori bassi di correlazione potranno essere visibili con un colore più cupo, tendente al blu. Ci si attente che i valori di correlazione più alti, quelli determinati da un colore caldo, si distribuiscano attorno alla diagonale principale, proprio perché le correlazioni tra coppie di fingerprint che appartengono alla stessa sorgente avranno un valore alto di similarità.

Dall'esperienza ottenuta tramite lo studio degli istogrammi precedenti (4.1), si vuole quindi ricercare un filtro che riesca a scremare i risultati di correlazione in modo da risaltare solo quelli significativi. Ciò potrebbe essere possibile utilizzando il valore di soglia σ descritto al paragrafo 4.1.

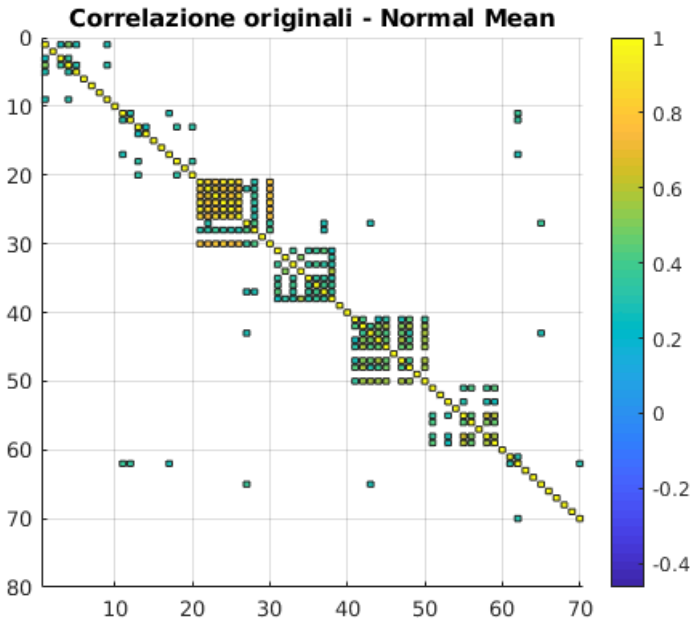

Figura 4.5: Totalità delle correlazioni tra le medie delle fingerprint originali

La soglia gioca un ruolo fondamentale nel filtraggio dei risultati. Tramite

la determinazione di tale valore, si riescono a discriminare i valori di correlazione più alti, cioè i valori che indicano un'alta similarità tra le fingerprint. In questo modo appariranno nel grafico solo le correlazioni che superano la soglia σ. Queste sono proprio le correlazioni tra le coppie di fingerprint che che provengono dalla stessa sorgente.

Ricordiamo che tale valore di soglia è dipendente dallo studio dell'istogramma relativo ad una particolare fotocamera, sarebbe quindi opportuno determinare tale valore per ogni strumento di cattura differente e filtrare i risultati di correlazione delle coppie di fingerprint con la soglia adatta.
Nella fattispecie si dovrebbero ottenere 7 differenti valori di soglia, e filtrare le diverse regioni del grafico, che identificano i confronti spiegati in precedenza, con i rispettivi valori σ. Per semplicità è stato determinato manualmente un unico valore di soglia con la quale è stato filtrato l'intero insieme di correlazioni per tutti i grafici mostrati in questa sezione.
Nel grafico in figura 4.6, vengono riportate le correlazioni tra tutte le coppie delle 70 fingerprint provenienti da video scaricati. In questo caso i valori di correlazione ottenuti sono nella media, la maggior parte in un intervallo di valori tra 0.50 e 0.96, il quale si può ritenere ancora un intervallo accettabile ai fini dell'identificazione di una buona fingerprint. Mettendo a paragone i due grafici rappresentanti i confronti tra le fingerprint ottenute da video originali e scaricati, si nota che molte correlazioni che nel primo superano il valore di soglia e vengono quindi visualizzate, nel secondo non compaiono più. Questa rilevante perdita di informazioni può essere dovuta alla compressione introdotta dalla piattaforma di messaggistica Whatsapp, rendendo le fingerprint relative meno significati e più deboli. Nonostante ciò, molti confronti danno un esito di correlazione soddisfacente, confermando l'ipotesi dell'unicità della fingerprint, anche nel caso in cui il file sorgente, dal quale viene estratta, abbia subito una netta compressione.

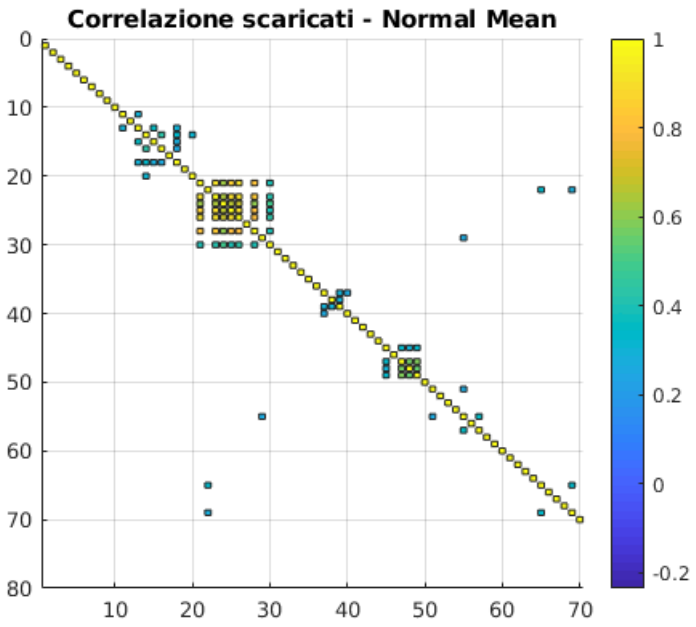

Figura 4.6: Totalità delle correlazioni tra le medie delle fingerprint estratte da video scaricati

Come accennato al paragrafo 4.1, considerare nel calcolo della media solo i fotogrammi di tipo I, peggiora i valori di correlazione ottenuti, non apportando nessun vantaggio o migliorie nella costruzione della fingerprint.
A prima vista, si nota come pervengano meno informazioni e i valori di correlazioni che riescono a superare il valore di soglia sono in numero inferiore, i quali non bastano per confermare la solidità della firma univoca.
I valori di correlazione tra le coppie di fingerprint in questione hanno valori che si aggirano tra e 0.26 e 0.58, un intervallo di valori comunque troppo basso non soddisfacente.

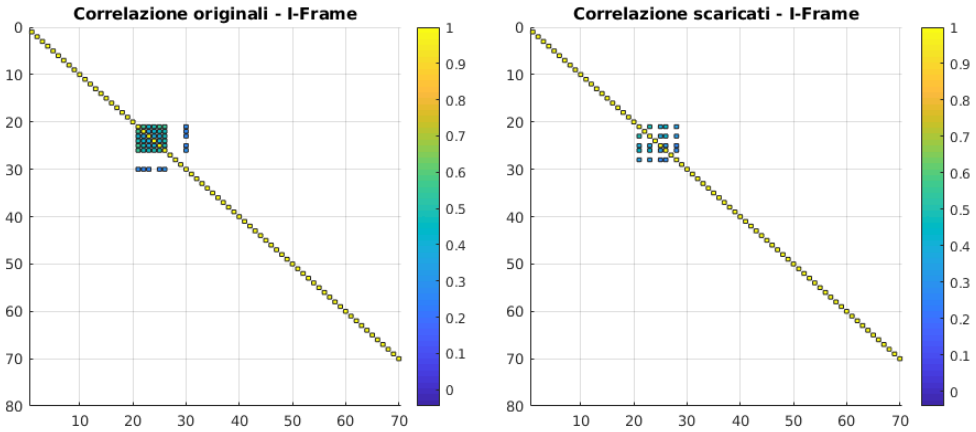

Figura 4.7: Correlazioni originali e scaricati su I-Frame

Conclusioni

Questa tesi ha avuto come scopo principale l'analisi di un processo di identificazione di una fotocamera sorgente, più specificamente è stata effettuata un'analisi sui dati provenienti da un processo di estrazione di fingerprint identificative da un insieme di contenuti digitali tramite l'applicazione di tecniche di eliminazione del rumore da video.
In particolare l'applicazione del filtro di denoising per i video **V-BM4D**, sviluppato e messo a disposizione dall'Università Finlandese di Tampere, si è rivelato uno strumento adatto per la creazione di fingerprint che possano essere identificative per una fotocamera sorgente.
Un secondo scopo, ma di eguale importanza, ha portato ad un'analisi dello stesso processo di identificazione di una sorgente digitale, sottoponendo però in una fase preliminare l'insieme dei video alla routine di passaggio su una piattaforma online, per il quale scopo è stato scelto di utilizzare l'applicazione di messaggistica istantanea Whatsapp.
La conseguente attività di estrazione su tali dati, che hanno subito una significativa compressione dovuta alla routine di upload e download, ha comunque portato a risultati soddisfacenti, nonostante si noti una grave perdita di informazioni, dovuta esattamente alla fase di elaborazione, che si traduce in perdita di specificità statistica.
E' stata messa a punto una strategia, che ha messo in atto una divisione del carico di lavoro eseguito in parallelo su differenti macchine fornite dal dipartimento di Informatica dell'Università di Bologna, rendendo più fluido l'intero processo, limitando le tempistiche e gestendo al meglio lo spazio di-

sponibile in memoria, evitando possibili intoppi durante la fase di estrazione. Il confronto tra i due metodi di calcolo della media, messi a punto per ottenere le fingerprint di referenza, hanno dato risultati che si discostano nettamente l'un l'altro. Le fingerprint calcolate per mezzo di una media sulla totalità dei frame, selezionati dai relativi video, riportano risultati ottimali, ottenendo nella fase di correlazione valori significativi. Effettuare il calcolo della media considerando solamente i I-Frame, ha abbassato notevolmente i coefficienti di correlazione dimostrando che tale costruzione delle fingerprint risulta inaffidabile per una possibile identificazione della sorgente.
E' stato dimostrato inoltre come si possa determinare un valore di **soglia** σ, per la discriminazione dei valori di correlazione significativi, utile alla costruzione di una buona fingerprint.
Questo progetto di tesi è da considerarsi come un punto di partenza per molteplici sviluppi futuri. L'analisi presentata in questo elaborato si limita al confronto di più correlazioni tra le fingerprint, verificando le sorgenti che manifestano un'alta similarità. Difatti, sarebbe stato necessario applicare una tecnica di clustering per eseguire effettivamente un procedimento di identificazione di una fotocamera sorgente, testando tale processo su alcuni video dei quali non si conosce la provenienza. L'identificazione della soglia σ è stato un passo di fondamentale importanza, che ha fornito uno strumento di classificazione dei risultati correlati, seppur essa sia stata determinata manualmente. Come lavoro futuro, da integrarsi in concomitanza con un processo di clustering, sarebbe interessante calcolare in maniera automatica tale valore effettuando uno studio appropriato sulle sorgenti conosciute. Infine, avendo a disposizione strumenti più potenti e spazi di memoria illimitati, si potrebbe ampliare tale lavoro considerando un'insieme di dati più grande, integrando il lavoro non solo sui video a scala di grigi ma anche a colori e cercando di costruire fingerprint più accurate che possano essere maggiormente identificative.

Appendice A

Codice sorgente per l'esecuzione dell'estrazione

In questa sezione verranno elencati per esteso tutti i codici sorgenti che sono stati sviluppati ed eseguiti ai fini dell'estrazione. Verranno in seguito elencati gli script necessari alla compattazione delle fingerprint e al calcolo delle fingerprint così mediate.

A.1 Bash Script

Il primo codice sorgente che viene mostrato è quello contenuto nello script bash *bash_script_parallel.sh*. Questo è lo script iniziale sviluppato per permettere l'inizio dell'estrazione eseguendo sulle macchine universitarie il comando MATLAB indicato tra apici. Viene chiesta la password in input tramite il comando *read* e viene eseguito in remoto per mezzo del protocollo ssh la funzione MATLAB *Start_video_analysis.m*, di cui parleremo in seguito. Nel codice riportato di seguito si può notare che vengono eseguite quattro estrazioni per l'utente 1 su un diverso range di frame. Da questa esecuzione ci si aspettano in output le fingerprint estratte dai video catturati dalla fotocamera frontale dell'utente 1.

Listing A.1: bash_script_parallel.sh

```
1  ##!/bin/bash
2
3  #Chiedo la password per ssh
4  echo -n Password:
5  read -s password
6  echo
7
8  #=============================================================
9  # UTENTE 1 Front
10
11 sshpass -p $password \
12 ssh -o StrictHostKeyChecking=no \
13 enrico.valguarnera@zuniga.cs.unibo.it \
14 " cd /home/projects/videofingerprint/script/ ; ...
15 nohup matlab -nodisplay -nosplash -r ...
16 \"Start_video_analysis(1,1,1,100,1); ...
17 exit;\" > user1_front_1-100.txt 2>&1 " &
18 echo Avviato 1 Grayscale Prima Parte Front
19
20 sshpass -p $password \
21 ssh -o StrictHostKeyChecking=no \
22 enrico.valguarnera@doncurzio.cs.unibo.it \
23 " cd /home/projects/videofingerprint/script/ ; ...
24 nohup matlab -nodisplay -nosplash -r ...
25 \"Start_video_analysis(1,1,101,200,2); ...
26 exit;\" > user1_front_101-200.txt 2>&1 " &
27 echo Avviato 1 Grayscale Seconda Parte Front
28
29 sshpass -p $password \
30 ssh -o StrictHostKeyChecking=no \
31 enrico.valguarnera@giovanna.cs.unibo.it \
32 " cd /home/projects/videofingerprint/script/ ; ...
33 nohup matlab -nodisplay -nosplash -r ...
34 \"Start_video_analysis(1,1,201,300,3); ...
35 exit;\" > user1_front_201-300.txt 2>&1 " &
36 echo Avviato 1 Grayscale Terza Parte Front
37
```

```
38 sshpass -p $password \
39 ssh -o StrictHostKeyChecking=no \
40 enrico.valguarnera@cettina.cs.unibo.it \
41 " cd /home/projects/videofingerprint/script/ ; ...
42 nohup matlab -nodisplay -nosplash -r ...
43 \"Start_video_analysis(1,1,301,400,1); ...
44 exit;\" > user1_front_301-400.txt 2>&1 " &
45 echo Avviato 1 Grayscale Quarta Parte Front
```

A.1.1 Funzione iframe_indices_extraction.sh

La seguente funzione contiene il comando *ffprobe*. E' un componente della suite software *FFmpeg*, in particolare è uno strumento da riga di comando per la gestione delle informazioni dei contenuti digitali. Il comando ricerca per il video in input tutti gli indici dei frame di tipo I e scrive l'output in un file di testo *indices.txt*. Questa funzione è utilizzata dalla funzione presentata nella sezione B.1.2.

Listing A.2: iframe_indices_extraction.sh

```
1 #!/bin/bash
2
3 ffprobe -select_streams v -loglevel panic -show_frames -
    show_entries frame=pict_type -of csv $1 | grep -n I | cut -d
    ':' -f 1 > indices.txt
```

A.2 MATLAB Script

In questa sezione dell'appendice elencheremo le funzioni sviluppate in MATLAB.

A.2.1 Funzione Start_video_analysis.m

La funzione MATLAB *Start_video_analysis.m* prende in input più parametri, necessari per selezionare dal dataset i dati dai quale estratte le fin-

gerprint: l'utente, la tipologia della fotocamera (front o rear), due parametri per indicare il range di frame che devono essere considerati per una determinata estrazione e infine un intero che necessità ai fini della formattazione dell'output.

Listing A.3: Start_video_analysis.m

```
1 function [] = Start_video_analysis(user, front, fromFrame, ...
2 toFrame, split_part)
3
4 path = strcat('/home/projects/videofingerprint/DB/',num2str(user
      ));
5 addpath('../VBM4D/');
6 Analyze_video_dataset(path, front, fromFrame, toFrame,
      split_part);
7
8 end
```

A.2.2 Funzione Analyze_video_dataset.m

L'esecuzione della funzione precedentemente mostrata fa si che venga richiamata un'altra funzione MATLAB, *Analyze_video_dataset.m*. Questo file contiene al suo interno numerose funzioni, tutte necessarie ai fini dell'estrazione. La funzione omonima costituisce la funzione principale dell'intero file. Si occupa, in base ai parametri che le vengono associati, della selezione e lettura dei video target per una determinata estrazione. Inoltre ha il compito di effettuare la conseguente applicazione del filtro di denoising e del salvataggio dell'output estratto, come viene spiegato a paragrafo 2.3.2.

Listing A.4: Analyze_video_dataset.m

```
1 function [] = Analyze_video_dataset(video_path, typeVideos, ...
2 fromFrame, toFrame, split_part)
3
4 scriptFolder = pwd;
```

```
5  % videos path
6  cd(video_path);
7  % gain user name to process
8  userName = split(video_path,'/');
9  userName = userName{end};
10 videoTypeStr = '';
11
12 Noise = struct('front', {}, 'rear', {});
13 Noise(1).user_name = userName;
14
15 % Start analysis
16 analyzedToday = 0;
17
18 % Check if is front
19 if(typeVideos == 1)
20     fprintf('Start to analyze front videos\n')
21     Noise(1).front = {};
22     cd('./front');
23     fprintf('User: %s\tType Folder: front\n', userName);
24     videoTypeStr = 'front';
25
26     % Start extraction for front video
27     [Noise.front,last_video] = Extract_noise(userName, ...
28     videoTypeStr, fromFrame, toFrame, split_part);
29     analyzedToday = analyzedToday + last_video;
30     % return to user dir
31     cd('..');
32 end
33
34 if(typeVideos == 0)
35     fprintf('Start to analyze rear videos\n')
36     Noise(1).rear = {};
37     cd('./rear');
38     fprintf('User: %s\tType Folder: rear\n', userName);
39     videoTypeStr = 'rear';
40     % Start extraction for rear video
41     [Noise.rear, last_video] = Extract_noise(userName, ...
42     videoTypeStr, fromFrame, toFrame, split_part);
```

```
43      analyzedToday = analyzedToday + last_video;
44
45  end
46
47  fprintf('Analysis finished %s\n',userName)
48  cd(scriptFolder);
49
50  % Send analysis report via email.
51  sendmail_cbd(['Utente: ' num2str(userName) 10 'Type Videos: '
        ...
52  num2str(videoTypeStr) 10 'Video Part: ' num2str(split_part) ...
53  10 'Analyzed:' num2str(analyzedToday) 10]);
54  end
55
56
57  function [tmp_videos,last_video] = Extract_noise(userName, ...
58  video_type, fromFrame, toFrame, split_part)
59  %  userName       : user index
60  %  video_type     : video type. Front or Rear.
61  %                   Can Assume value 0 or 1.
62  %                   If assume value 1 front user video
63  %                   will be analyzed.
64  %                   If 0 rear user video will be analyzed.
65  %
66  %                  RGB  : video is analysed in RGB
67  %
68  %                   Gray : video is analyzed in grayscale.
69  % fromFrame          : start frame
70  % toFrame            : final frame
71  % split_part         : integer that need to format output
72
73  fprintf('Start extracting noise... \n')
74  video_list = dir
75  tmp_videos = cell(1,length(video_list));
76  last_video = 0;
77
78  for i=3:length(video_list)
79      current_video = read_video_range(video_list(i).name, ...
```

```
80                  fromFrame, toFrame);
81      currentTime = clock;
82      currentTime = currentTime(4:end);
83
84      mat_name = str2num(video_list(i).name(1:end-4));
85
86      tmp_videos{i} = Analyze_VBM4D(current_video);
87
88      % difference to obtain fingerprint.
89      noise_final = single(current_video) - tmp_videos{i};
90      % Saving fingerprint in relative .mat file.
91      Save_videos(userName, mat_name, noise_final, video_type, ...
92                  split_part);
93
94      fprintf('Video %d analysed \n', num2str(i-2));
95      last_video = i;
96  end
97  fprintf('Noise extraction finished\n');
98  end
99
100
101 function[y_est] = Analyze_VBM4D(y)
102 % y          : estimation. Extracted fingerprint.
103
104
105 sigma      = 25;       % Noise standard deviation. it should be
106                        %    in the same
107                        % intensity range of the video
108 profile = 'lc';        % V-BM4D parameter profile
109                        %  'lc' --> low complexity
110                        %  'np' --> normal profile
111 do_wiener = 1;         % Wiener filtering
112                        %   1 --> enable Wiener filtering
113                        %   0 --> disable Wiener filtering
114 sharpen = 1;           % Sharpening
115                        %   1 --> disable sharpening
116                        %  >1 --> enable sharpening
117 deflicker = 1;         % Deflickering
```

```
118                         %   1 --> disable deflickering
119                         %  <1 --> enable deflickering
120 verbose = 1;            % Verbose mode
121
122
123 % V-BM4D filtering
124 y_est = vbm4d( y, sigma, profile, do_wiener, sharpen, ...
125             deflicker, verbose );
126
127
128 end
129
130
131
132 function [] = Save_videos(user, index, video_fingerprint, ...
133             video_type, split_part)
134
135 currentFolder = pwd;
136
137 % fingerprint path in machines file system.
138 fingerprint_folder='/home/projects/videofingerprint/Fingerprint'
    ;
139 pathName = strcat(fingerprint_folder,'/',user,'/', ...
140             video_type, '/', num2str(split_part),'/');
141
142 cd(char(pathName));
143 % fileName for saving video fingerprint
144 fileName = strcat(num2str(index),'.mat');
145 % variable to save renaming
146 newName = strcat('fingerprint', '_', num2str(split_part));
147 S.(newName) = video_fingerprint;
148
149 save(fileName, '-struct', 'S')
150
151 cd(currentFolder);
152 end
```

A.2.3 Funzione read_video_range.m

La funzione *read_video_range.m* si occupa della lettura del range di frame di un video, ovvero del gruppo di frame per il quale si vuole effettuare un'estrazione. Ricordiamo che la selezione di una porzione di frame, quindi la frammentazione dell'estrazione in parti di fingerprint, è necessaria per motivi di mancanza di risorse e inoltre è attuata per sfruttare l'esecuzione in parallelo su più macchine. La funzione necessita in input dei due valori interi, che identificano l'intervallo di frame, e il nome del video da leggere. Dunque, vengono letti 100 frame da un particolare video, viene estratto il relativo canale Y di luminanza e viene assegnato alla variabile di output y. Nel caso in cui non vengano forniti in input gli estremi per l'intervallo di frame, allora verranno considerati tutti i frame del video nella lettura.

Listing A.5: read_video_range.m

```
1
2  function y = read_video_range(file_name, fromFrame, toFrame)
3
4  fprintf('Loading test sequence "%s" \n', file_name);
5  mov        = VideoReader(file_name);
6  if ~exist('fromFrame','var') || ~exist('toFrame','var') || ...
7          isempty(fromFrame) || isempty(toFrame)
8      fromFrame = 1;
9      toFrame = mov.NumberOfFrames;
10 end
11 nFrames    = min(toFrame,mov.NumberOfFrames);
12 vidHeight = mov.Height;
13 vidWidth   = mov.Width;
14 isRGB = strcmpi(mov.VideoFormat,'RGB24');
15
16 %frames = read(mov);
17 mov = VideoReader(file_name);
18 ind = 0;
19 for i=fromFrame:toFrame
20  ind = ind + 1;
21    frames(:,:,:,ind) = read(mov, i);
```

```
22 end
23
24
25 if isRGB
26  diff = toFrame - fromFrame + 1;
27  for i=1:diff
28              y(:,:,i) = rgb2gray(frames(:,:,:,i));
29          end
30 end
```

A.2.4 Funzione sendmail_cbd.m

La funzione *send_mail.m* è stata sviluppata per l'invio di una notifica tramite mail alla fine di un'estrazione, nel caso essa vada a buon fine. La mail prende in input un messaggio che viene preventivamente formattato dalla funzione chiamante, fornendo informazioni rilevanti, come l'utente estratto, la tipologia di fotocamera, la data e il numero di video analizzati. La mail utilizzata per inviare tali notifiche è stata fornita dal Dipartimento di Informatica dell'Università di Bologna.

Listing A.6: Start_video_analysis.m

```
1  function [] = sendmail_cbd(message_text)
2
3  %mail and password
4  mail = 'progetto.unibo.cbd@gmail.com';
5  password = 'progettocbd';
6  %date formatting
7  t = datetime('now');
8  [Y,M,D,H,MI] = datevec(t);
9  Y = num2str(Y);
10 M = num2str(M);
11 D = num2str(D);
12 H = num2str(H);
13 MI = num2str(MI);
```

```
14
15 %sender group mail
16 % remember: complete with Prof and Flavio's mails
17 recipient = {'valguarneraenrico@gmail.com'};
18 %setting pref
19 setpref('Internet','SMTP_Server','smtp.gmail.com');
20 setpref('Internet','E_mail',mail);
21 setpref('Internet','SMTP_Username',mail);
22 setpref('Internet','SMTP_Password',password);
23 %setting preperties
24 props = java.lang.System.getProperties;
25 props.setProperty('mail.smtp.auth','true');
26 props.setProperty('mail.smtp.socketFactory.class', ...
27 'javax.net.ssl.SSLSocketFactory');
28 props.setProperty('mail.smtp.socketFactory.port','465');
29 fprintf('Sendind mail...');
30 %To format in line use '10' between to text message in sendmail.
31 %finally send a mail from progetto.unibo.cbd mail to recipients
       mail.
32 sendmail(recipient, 'Analysis Report', ...
33         [message_text 10 'Date: ',Y,'-',M,'-',D,'  ',H,':',MI]);
34
35 end
```

Appendice B

Codice sorgente per la fase di manipolazione sulle fingerprint

B.1 Calcolo della media delle fingerprint

In questa sezione verranno presentate le funzioni e gli script sviluppati per il calcolo delle medie delle fingerprint.
Le matrici bidimensionali finali, necessarie per la fase di correlazione, sono state calcolate effettuando una media sulla totalità della terza dimensione della struttura tridimensionale che rappresenta la fingerprint, come spiegato al paragrafo 3.1.1.
Successivamente, è effettuato lo stesso calcolo di media ma considerando per la terza dimensione solo gli indici corrispondenti agli I-Frame del video sorgente dal quale è stata estratta la fingerprint. Vedremo in seguito il codice sorgente sviluppato per implementare tale procedimento.

B.1.1 Funzione fingerprint_normal_avarage.m

Nel seguente listato viene mostrato il codice sorgente per calcolare la media delle fingerprint. Come spiegato al capitolo 3, le fingerprint vengono mediate sulla terza dimensione, cioè viene effettuata una media dei valori

sulla lungo i frame presi in considerazione. Questa funzione MATLAB crea, per ogni fingerprint dell'utente indicato in input, una matrice bidimensionale effettuando la media sulla terza dimensione della matrice tridimensionale che identifica la suddetta fingerprint, sfruttando inoltre la tecnica di ricombinazione spiegata al paragrafo 3.1. In output si otterrà l'insieme delle 20 matrici bidimensionali, una per ogni fingerprint dell'insieme selezionato in input.

Listing B.1: fingerprint_normal_avarage.m

```
1  function [] = fingerprint_normal_avarage(path, user)
2
3  start_time = tic;
4  addpath('path_to_project_folder');
5  initial_path = pwd;
6  % print read fingerprint part
7  fprintf('path to compat: %s\n', path)
8
9  destination_path = 'path_to_destination_folder';
10 cd(path);
11 % take part list folder in the relative directory of the input
       path
12 part_list = dir;
13
14 for vid=1:20
15     tmp_fp = [];
16     frames = 0;
17     mat_name = strcat(num2str(vid), '.mat');
18
19     for i=3:length(part_list)
20
21         % fingerprint name inside mat, example: fingerprint_5
22         fp_name=strcat('fingerprint_',num2str(part_list(i).name)
               );
23
24         if(exist(['./' part_list(i).name],'dir') == 7)
25             % enter in the part directory
26             cd(part_list(i).name);
```

```
27                fprintf('Enter in part %s\n', part_list(i).name)
28
29                if(exist(['./' mat_name],'file') == 2)
30                      fprintf('Loading mat %s ...\n', mat_name)
31                      fp = load(mat_name, fp_name);
32                      % calculating frame number for division
33                      frames = frames + size(fp.(fp_name), 3);
34                      fprintf('Number of frames: %s', num2str(
                            frames))
35
36                       % calculating sum
37                      tmp_fp = sum(cat(3,tmp_fp, fp.(fp_name)),3 );
38                      clear fp
39                end
40          end
41          cd('..');
42      end
43
44      fprintf('Calculating mean for video %d\n', vid)
45      % calculating final mean
46      fp_mean = tmp_fp / frames;
47      fp_name = strcat(destination_path, 'fp_mean_', ...
48                num2str(user), '_', num2str(vid), '.mat');
49
50      fprintf('Saving mean final matrix in mat %s\n', fp_name)
51      save(fp_name,'fp_mean');
52  end
53  cd(initial_path);
54  end
```

B.1.2 Funzione fingerprint_iframe_avarage.m

Come viene introdotto nel paragrafo 3.1.2, mostriamo il codice sorgente della funzione *fingerprint_iframe_avarage.m*. Essa effettua una media sulla terza dimensione della fingerprint, selezionando solo gli indici che identificano i I-Frame che componono il video originale. Ricordiamo che la fingerprint

ha la stessa struttura del video dal quale è stata estratta. Si procede con la ricerca degli indici degli I-Frame, i quali vengono salvati temporaneamente in un file di testo e immagazzinati in un vettore. Questo compito spetta alla funzione *getIFramesIndices*; successivamente viene fatta una selezione sulla terza dimensione.

Listing B.2: fingerprint_iframe_avarage.m

```
1
2  function [] = fingerprint_iframe_avarage(fp_path, db_path, user)
3  start_time = tic;
4  addpath('aggiungo_la_directory di lavoro');
5  initial_path = pwd;
6  destination_path = 'percorso_di_destinazione';
7
8  % composizione del percorso per lo script bash necessario
9  pathToScript = fullfile(pwd, 'iframe_index_extraction.sh');
10 cd(fp_path);
11 part_list = dir;
12
13 for vid=1:20
14     tmp_fp = [];
15     cd(db_path);
16     video_list = dir;
17     video_path = fullfile(pwd, video_list(vid+2).name);
18
19     IF_indices = getIFrameIndices(pathToScript, video_path);
20     IF_count = size(IF_indices,2);
21
22     % composizione del nome del file .mat
23     mat_name = strcat(num2str(vid), '.mat');
24     cd(fp_path);
25
26     for i=3:length(part_list)
27         % composizione del nome della variabile contenuta dentro
               il file .mat
28         fp_name = strcat('fingerprint_', num2str(part_list(i).
               name));
```

```
29
30          if(exist(['./' part_list(i).name],'dir') == 7)
31              cd(part_list(i).name);
32              if(exist(['./' mat_name],'file') == 2)
33                  fp = load(mat_name, fp_name);
34                  fp_iframe = fp.(fp_name)(:,:,IF_indices{i
                        -2}(:));
35                  % calcolo della somma dei frame selezionati
                        sulla terza dimensione
36                  tmp_fp = sum(cat(3,tmp_fp, fp_iframe),3 );
37              end
38          end
39          cd('..');
40      end
41      % divisione per il calcolo della media
42      fp_mean = tmp_fp / IF_count;
43      fp_name = strcat(destination_path, 'fp_mean_', num2str(user)
            , '_', num2str(vid), '.mat');
44
45      fprintf('Saving mean final matrix in mat %s\n', fp_name)
46      save(fp_name,'fp_mean');
47  end
48
49  % return to original directory
50  cd(initial_path);
51  fprintf('Final Time For Compatting: %.1fs\n', toc(start_time));
52
53  end
```

B.1.3 Funzione getIFramesIndices

Tale funzione viene chiamata dalla funzione fingerprint_iframe_avarage.m. Richiede in input il nome del percorso dello script bash iframe_indices_extraction.sh, e il video per il quale si vogliono ricercare gli indici dei I-Frame.
Restituisce in output una struttura di tipo *cell array*, contenente i suddetti indici, suddivisi per range di frame. Infatti, nelle righe 13 e 14 si può notare

che viene fatto un controllo sul range di frame. Vista la struttura particolare che si è deciso di adottare per il processo di estrazione, è stato necessario dividere gli indici in 15 celle differenti.

Listing B.3: getIFramesIndices.m

```
1  function [B] = getIFrameIndices(pathToScript, video_path)
2  cmdStr = ['.' pathToScript ' ' video_path];
3  % run system command
4  system(cmdStr);
5
6  file = fopen('indices.txt', 'r');
7  A = fscanf(file, '%d');
8  B = {};
9  range = 100;
10 for part= 1:15
11     z = 1;
12     for y=1:length(A)
13         if((A(y) > (range-100)) && (A(y) < range))
14             B{part}(z) = A(y) - (100 * (part - 1));
15             z = z + 1;
16         end
17     end
18     range = range + 100;
19 end
20 end
```

Bibliografia

[1] J. Lukàs, J. Fridrich, and M. Goljan, *Digital camera identification from sensor pattern noise*, IEEE Trans. Inf. Forensics Secur.1 (June (2)) (2006) 205-214.

[2] Z. Geradts, T. Gloe, *D6.8b: Identification of images*. 2009.

[3] M. Goljan, J. Fridrich, and T. Filler, *Large scale test of sensor fingerprint camera identification*, Media Forensics and Security. (Feb (4)) (2009).

[4] M. Tiwari, B. Gupta, *Image features dependant correlation-weighting function for efficient PRNU based source camera identification*, Forensic Sci Int. 285 (2018) 111-120.

[5] S. Milani, L. Cuccovillo, M. Tagliasacchi, S. Tubaro, P. Aichroth, *Video camera identification using audio-visual features*, IEEE EUVIP, Dec. 10-12 (2014).

[6] Y. Scheelen, J. van der Lelie, *Camera identification on Youtube*, (feb (12)) (2012).

[7] Van Houten W. , Geradts Z. , *Source video camera identification for multply compressed videos originating from Youtube*, Elsevier, Digital Investigation 6 (May (5)) (2009) 48-60.

[8] Q. Rao, J. J. Wang, L. M. Zhang, *Enhancing source camera identification based on multiplicative denoising filter*, IEEE TrustCom-BigDataSE-ISPA (2016).

[9] W. H. Chuang, H. Su, M. Wu, *Exploring compression effects for improved source camera identification using strongly compressed video*, IEEE 18th Int. Conf. on Imange Processing, (2011)

[10] S. Yahaya, A. TS Ho, A. Wahab, *Advanced video camera identification using conditional probability features*, IET Conference on Image Processing (2012).

[11] D. Shullani, M. Fontani, M. Iuliani, O. Al Shaya, and A. Piva, *VISION: a video and image dataset for source identification*, EURASIP Journal on Information Security (2017).

[12] A. W. Wahab, J. A. Briffa, H. G. Schaathun, and A. TS Ho, *Conditional probability based steganalysis for JPEG steganography*, IEEE International Conference on Signal Processing Systems (2009)

[13] D. Freire-Obregòn, F. Narducci, S. Barra, M. Castrillon-Santana , *Deep learning for source camera identification on mobile devices*, Elsevier ,Pattern Recognition Letters 000 (2018) 1-6.

[14] S. K. Kuanar , R. Panda, A. S. Chowdhury, *Video key frame extraction through dynamic Delaunay clustering with a structural constraint*, Elsevier, J.Vis. Commun. Image R. (2013) 1212-1227.

[15] I. Amerini, R. Caldelli, A. Del Mastio, A. Di Fuccia, C. Molinari, A. P. Rizzo, *Dealing with video source identification in social networks*, Elsevier, Signal Processing: Image Communication 57 (2017) 1-7.

[16] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," IEEE Trans. Image Process., vol. 16, no. 8, August 2007.

[17] M. Maggioni, G. BoracchiG. , A. Foi, K. Egiazarian, *Video denoising using separable 4-D nonlocal spatiotemporal transforms*, Proc. SPIE Electronic Imaging 2011, Image Processing: Algorithms and Systems IX, 7870-2, San Francisco (CA), USA, January 2011. doi:10.1117/12.872569

[18] W.H. Chuang, H. Su, M. Wu, *Exploring compression effects for improved source camera identification using strongly compressed video*, in: 2011 18th IEEE International Conference on Image Processing, 2011, pp. 1953–1956. doi:10.1109/ICIP.2011. 6115855.

[19] G. J. Bloy, *Blind Camera Fingerprint and Imae Clustering*, IEEE Transaction On Pattern Analysis and Machine Intelligence, Marzo 2008, Vol 30, No. 3, doi:10.1109/TPAMI.2007.1183

Ringraziamenti

Il percorso accademico magistrale è stata l'esperienza più importante della mia vita, ho conosciuto persone nuove, ho rafforzato le mie capacità e ho ampliato enormemente le mie conoscenze informatiche, ma non solo...
Ho capito che le persone che mi hanno permesso tutto ciò, mia madre, mio padre, tutti i membri della famiglia, che mi sono stati accanto e mi hanno dato conforto "da giù"... mentre io ero "sopra", tutti i miei amici, che seppur non sentendoci ne vedendoci ogni giorno, mi hanno sempre fatto sentire ad ogni mio ritorno, come se non fossi mai partito, sono ciò di cui io ho bisogno e sono stata la vera forza che mi ha permesso di finire questo percorso con massima serenità.

Ringrazio i miei coinquilini che mi hanno accompagnato in questo percorso, in particolare il mio amico Luca, che nelle "tiepide" giornate bolognesi mi ha intrattenuto con la sua amicizia e il suo immancabile "senso dell'umorismo"! Non potevo trovare di meglio *comb*!

Ringrazio tutti gli amici che ho conosciuto a Bologna e spero che la nostra amicizia continui anche se gli obblighi del lavoro, molto probabilmente, ci porteranno a vivere in città diverse..ma chi lo sà!

Ringrazio inoltre il chia.mo prof. Montesi e il Dott. Flavio Bertini per avermi seguito durante il lavoro di tesi.

Beh penso di aver ringraziato tutti...ahhh si, ma che stupido! Potevo mai dimenticarti !?
Sei stata tu che mi hai sostenuto più di tutti, aspettando con ansia ogni mio ritorno. La sfrenata speranza di ritornare ogni mese per vedere i tuoi splendidi occhi azzurri mi ha accompagnato in questo lungo percorso, ma abbiamo resistito e siamo arrivati fin qua! Grazie.

Adesso comincia una nuova vita e sicuramente nuove esperienze, sperando sempre nel meglio.