Università degli Studi di Firenze
Facoltà di Ingegneria - Dipartimento di Sistemi e Informatica

Tesi di Laurea Magistrale in Ingegneria Informatica

Analisi della similarità visuale per la ricerca di video annotati in archivi di grande dimensione

Candidato
Salvatore Smeraldo

Relatori
Prof. Alberto Del Bimbo

Ing. Marco Bertini

Correlatori
Ing. Lamberto Ballan

Ing. Giuseppe Serra

Anno Accademico 2009-2010

Ringraziamenti

Desidero porgere i miei ringraziamenti alle persone grazie alle quali si è reso possibile lo svolgimento di questo lavoro di tesi. Un ringraziamento particolare va al Prof. Alberto Del Bimbo per avermi permesso di affrontare un'esperienza di studio molto stimolante e sicuramente formativa su temi di ricerca molto interessanti e attuali. Un forte ringraziamento va anche all'Ing. Marco Bertini, all'Ing. Lamberto Ballan e all'Ing. Giuseppe Serra. Grazie alla loro disponibilità costante ed al forte interesse dimostrato, hanno contribuito a risolvere le problematiche incontrate durante questo lavoro. Un ringraziamento speciale lo dedico ai miei genitori e a mio fratello Vincenzo per avermi permesso di completare gli studi e per essermi stati sempre di supporto. Desidero inoltre dedicare un forte ringraziamento a Isabella che mi è stata particolarmente vicina durante tutto il periodo di svolgimento della tesi. Infine un ringraziamento particolare va agli amici Mirco, Alessandro e Tommaso che hanno condiviso con me i momenti di gioia e quelli più bui incontrati durante tutto il percorso universitario.

Indice

Capitolo 1

Introduzione

In questa tesi sono stati affrontati temi legati all'analisi del contenuto informativo di video e immagini, concentrandosi, in particolare, sulla risoluzione dei seguenti problemi:

1. Ritrovamento di video simili, cioè appartenenti alla stessa categoria semantica, in database di dimensioni molto estese;

2. Suggerimento di tag ai singoli shot di un video, grazie all'analisi della similarità visuale tra frame del video ed un insieme di immagini annotate.

Analizziamo brevemente gli ambiti in cui vanno a collocarsi le due parti realizzate, cercando di capire le principali caratteristiche e le maggiori funzionalità che questi sistemi devono possedere per poter risultare funzionali.

Negli ultimi anni la quantità di informazione multimediale generata sottoforma di video presente in rete è cresciuta tremendamente grazie alla popolarità acquisita dai siti web basati sulla condivisione di filmati quali *YouTube* o *Google*

Video. Ogni giorno vengono aggiunti molti video a questi siti (si pensi che secondo una ricerca del Marzo 2010 su *YouTube* vengono caricate 24 ore di video in ogni minuto), per cui c'è un bisogno crescente di sistemi di gestione di contenuti multimediali. Un problema complesso, che si sta affrontando negli ultimi tempi, è quello del confronto tra video basato sul loro contenuto per identificare video distinti che rappresentano lo stesso concetto semantico. Una sfida importante, affrontata dalla prima parte di questo lavoro di tesi, è la realizzazione automatica di una ricerca accurata e affidabile dei video maggiormente simili ad un nuovo video caricato in un sito web. A tal proposito, c'è anche la necessità che il meccanismo di ritrovamento sia rapido, date le dimensioni dei database in gioco. In letteratura non sono presenti molti approcci che si occupino esattamente di questo tipo di problema. Esistono, infatti, sistemi per l'identificazione dei *near duplicate* di un video, oppure approcci per l'identificazione di eventi in un video e questi ultimi, molto spesso, utilizzano tecniche di apprendimento supervisionato per l'identificazione dell'evento a partire da un insieme di eventi predefiniti. Il sistema realizzato in questa tesi, invece, si propone di identificare i video che rappresentino lo stesso concetto semantico di un video di partenza, senza l'utilizzo di tecniche di apprendimento.

Per quanto riguarda la parte di tesi che riguarda il suggerimento dei tag, al giorno d'oggi tutti i siti web che consentono la condivisione di contenuto multimediale, quali *Flickr, Facebook, YouTube*, offrono all'utente anche la possibilità di annotare (assegnare dei tag) il materiale che egli condivide. I tag vengono poi utilizzati per recuperare i contenuti presenti in tali siti e per facilitarne la navigazione. Ad esempio, *Flickr* e *Facebook* non solo consentono di etichettare un'immagine intera, ma anche di associare tag a singole porzioni di immagine. Viceversa, etichettare le singole sequenze video all'interno di un filmato è un lavoro maggiormente oneroso, così gli utenti tendono a taggare l'intero contenu-

to di un video. Inoltre, visto che il processo di etichettatura è completamente manuale, gli utenti cercano di impiegare meno tempo possibile nel processo di assegnazione dei tag, così da ottenere annotazioni sparse e scarsamente uniformi del contenuto visuale. In definitiva, la lista dei tag associati ad un video è solo una descrizione generale del video nel suo complesso, senza indicazioni temporali di dove i tag siano realmente presenti. Per affrontare questo tipo di problema, in questa tesi si propone un sistema per il suggerimento di nuovi tag e per la localizzazione temporale degli stessi all'interno della sequenza video.

Per risolvere i due problemi suddetti sono stati utilizzati due approcci distinti che però condividono un'idea comune. L'idea alla base dei due metodi, riscontrabile nella quasi totalità degli approcci a questi tipi di problema, è la modellazione di un oggetto, o comunque di una scena, attraverso un insieme di punti di interesse locali. I suddetti punti di interesse devono essere invarianti a trasformazioni geometriche come traslazioni, rotazioni e scalature e a trasformazioni affini. Per questo motivo lo standard è divenuto l'utilizzo dei punti **SIFT** di Lowe [Lowe04] per le loro buone prestazioni e per il loro costo computazionale relativamente basso. Un'alternativa molto valida è l'utilizzo dei punti **SURF** [SURF] che presentano costi computazionali più bassi in termini di tempi di esecuzione.

Nell'ambito dell'utilizzo di feature robuste, una soluzione che è divenuta molto popolare negli ultimi tempi è l'approccio **Bag-of-Words**, originariamente proposto per la processazione di linguaggi naturali e per il recupero di informazioni. In tale ambito, il metodo Bag-of-Words è utilizzato per la categorizzazione dei documenti, dove ogni documento viene rappresentato dalla frequenza delle parole che vi sono contenute. Nel dominio visuale, invece, un'immagine o il frame di un video è l'analogo visuale di un documento e può essere rappresentato da un insieme (bag) di descrittori locali invarianti (SIFT o SURF), detti *parole visuali*.

Partendo da questa base comune, i due metodi proposti si sviluppano secondo strade distinte. La prima parte del lavoro deve riconoscere video appartenenti allo stesso gruppo semantico, per esempio riguardanti calcio, basket, oppure una specifica trasmissione televisiva. In questa fase, si deve lavorare con database di dimensioni molto vaste, per cui c'è la necessità di avere un approccio dalle buone prestazioni, ma anche rapido da realizzare. Per questi motivi è stato pensato un sistema basato sui Bag-of-Visual-Words di punti SURF. Più specificamente, viene proposto un approccio basato su una **foresta di vocabolari visuali a più livelli ottenuti tramite clustering gerarchico**. Inoltre sono state utilizzate informazioni di colore, più precisamente la componente H della rappresentazione di colore HSV. Seguendo questo approccio ogni frame viene rappresentato con un descrittore di poche dimensioni, che racchiude informazioni globali e locali. Per questo motivo un video è rappresentabile come una successione di descrittori. In questo ambito si deve lavorare con quantità di video molto estese, per cui è stata proposta anche una struttura di indicizzazione che consenta di recuperare le relazioni tra filmati. La struttura utilizzata è il CFP-Tree, dove gli indici sono formati proprio dai descrittori di frame suddetti.

Per la parte di localizzazione e suggerimento di tag, invece, è stato creato un sistema che vada a confrontare i frame rappresentanti gli shot di un video con una serie di immagini annotate. In questo modo si ricavano le immagini maggiormente simili da un punto di vista visuale per poter applicare un meccanismo di valutazione dei tag, mirato al suggerimento degli stessi ai singoli shot del video. Il set di immagini con cui confrontare gli shot del video viene fornito dal lavoro di tesi di Nencioni [Nencioni10] applicato al video all'interno del quale si vogliono localizzare e suggerire tag. Questo algoritmo realizza un processo di eliminazione dei tag non rilevanti dal video di partenza, quindi un'espansione semantica di quelli rimanenti, in modo che al temine della procedura il video

di partenza possieda un insieme di tag più completo. Con tali tag vengono realizzate interrogazioni al sito *Flickr* per recuperare un insieme di immagini annotate.

L'intero insieme di immagini annotate viene confrontato con i frame che costituiscono il video di partenza. La seconda parte del mio lavoro di tesi si occupa di realizzare questo aspetto del problema. Per il confronto tra immagini, è stato proposto un descrittore basato su una **foresta di vocabolari visuali costruita sui BoW di punti SIFT**. Più specificamente, viene utilizzato un approccio che può essere chiamato **TOP-SIFT** perchè sfrutta solo le componenti SIFT più presenti nelle immagini e consente di ridurre le dimensioni del descrittore e rendere il confronto tra immagini più veloce. Al TOP-SIFT, infine, vengono aggiunte informazioni sulla distibuzione degli *edge*, grazie all'uso del descrittore **Edge Histogram Descriptor** dello standard *MPEG-7*, e informazioni di colore tramite un **correlogramma di colore** basato sulla componente H della rapresentazione di colore HSV.

Capitolo 2

Stato dell'arte

In questo capitolo viene realizzata una descrizione generale degli articoli presenti in letteratura legati ai lavori svolti in questa tesi. In particolar modo, vengono descritti gli articoli che hanno fornito idee importanti o che hanno mostrato degli approcci particolarmente innovativi, utili durante lo svolgimento di alcune fasi di progettazione della tesi.

In 2.1 è presentato un articolo che descrive l'approccio Bag-of-Visual Words e la sua applicazione al problema della classificazione visuale. La sezione 2.2 è riservata alla descrizione di un articolo in cui viene presentato un metodo per la rilevazione di eventi nei video, che è stato utile per comprendere ancora meglio la modalità di funzionamento dei Bag-of-Visual Words. In 2.3 viene offerta la descrizione di un sistema scalabile per il riconoscimento di immagini che consente di lavorare efficientemente con un vasto numero di tipi di oggetti da riconoscere. Lo schema prevede l'utilizzo di descrittori locali quantizzati gerarchicamente in un vocabolario ad albero. Questo articolo ha fornito l'ispirazione per la realizzazione di un vocabolario di parole visuali ad albero. La sezione successiva, 2.4, descrive un metodo che sviluppa un insieme di vocabolari ad al-

bero (foresta) che vengono calcolati con una modalità ad incremento e quindi si adattano all'aggiunta di nuove immagini al database. Da questo articolo è stata estrapolata l'idea di utilizzare una foresta di vocabolari ad albero per offrire una rappresentazione più estesa della conoscenza. Gli articoli descritti nelle sezioni 2.5 e 2.6 presentano due approcci distinti per la rilevazione di eventi all'interno dei video e sono stati utili per ottenere una visione più nitida su come affrontare l'argomento e su quali potessero essere le maggiori problematiche incontrabili. Gli articoli descritti in 2.7 e 2.8 sono stati importanti per capire come affrontare il problema della localizzazione temporale dei tag all'interno dei video, fornendo una visione complessiva sulle modalità con cui realizzare il sistema. L'articolo in 2.9, invece, presenta il descrittore TOP-SURF, da cui è stata tratta ispirazione per l'implementazione del TOP-SIFT, utilizzato come descrittore di immagini nel programma per il suggerimento dei tag. Infine, l'articolo 2.10 descrive un approccio per la diffusione e il suggerimento dei tag a partire dalla similarità visuale. Inoltre viene proposto l'algoritmo di classificazione dei tag $Vote^{+}$. Queste informazioni sono state utilizzate per la progettazione della parte di codice mirata a suggerire i tag, una volta identificata le immagini maggiormente simili ad un frame del video.

2.1 Bag-of-Visual-Words Expansion Using Visual Relatedness for Video Indexing

In questo articolo [Jiang08] viene presentato un metodo di espansione della tecnica Bag-of-Visual-Words (BoW) per poterla rendere più funzionale al problema della classificazione visuale.

Prima di entrare nello specifico del metodo, viene data una spiegazione sui BoW e del loro utilizzo nell'ambito della classificazione visuale. Da un'immagine si

possono estrarre dei Keypoint locali (SURF, SIFT, etc...). Questi keypoint vengono poi raggruppati in un certo numero di cluster ed ogni cluster viene trattato come una parola visuale. Con i keypoint mappati nelle parole visuali, un'immagine viene rappresentata come un vettore di feature che tenga conto della presenza (o del numero di occorrenze) delle parola visuale. Questa rappresentazione del BoW è analoga alla rappresentazione bag-of-words per i documenti testuali.

Con il modello BoW testuale, un documento viene rintracciato se esso contiene i temini di ricerca di una query. Nel caso della classificazione di immagini o di video, e quindi nel caso di Bag-of-Words visuali, il problema è più complicato. Infatti le parole visuali sono l'output di algoritmi di clusterizzazione e possono essere correlate l'un l'altra per motivi di quantizzazione. Per alleviare il problema dei sinonimi, gli autori propongono un nuovo approccio per l'espansione dei BoW che vada a rimuovere gli effetti della correlazione tra parole visuali.

Basandosi su un vocabolario di parole visuali, viene costruita un'*ontologia visuale* per modellare la relazione *is-a* tra parole visuali. All'interno dell'ontologia visuale, viene definita rigorosamente la *relazione visuale* tra parole visuali, in modo simile a come viene stimata la relazione semantica tra parole testuali [Jiang97].

2.1.1 Relazione tra parole visuali

Dato un insieme di keypoint, per prima cosa viene costruito un vocabolario visuale attraverso la clusterizzazione dei keypoint tramite un algoritmo *k-means*. Da questo vocabolario visuale viene generata un'ontologia visuale applicando un cluster agglomerativo per raggruppare gerarchicamente due parole visuali alla volta in modalità *bottom-up*. Di conseguenza, le parole visuali nel vocabolario vengono rappresentate in un albero gerarchico, detto ontologia visuale. In questo

albero le foglie sono le parole visuali, mentre i nodi interni sono i pro-genitori che modellano la relazione *is-a* tra le parole visuali. Nell'ontologia visuale, ogni nodo è un'ipersfera nello spazio delle feature dei keypoint. La dimensione (il numero di keypoints) dell'ipersfera cresce attraversando l'albero dalle foglie verso la radice. Per stimare la relazione tra le parole visuali viene utilizzata l'ontologia JCN. Denotando v_ie v_j come due parole visuali, JCN considera l'IC (information content) dei pro-genitori comuni e delle due parole confrontate:

$$JCN(v_i, v_j) = \frac{1}{IC(v_i) + IC(v_j) - 2 \cdot IC(LCA(v_i, v_j))} \tag{2.1}$$

dove LCA è il pro-genitore comune con la posizione più bassa nell'albero dell'ontologia visuale delle parole visuali v_i e v_j. IC viene quantificato come la componente negativa della log verosimiglianza della probabilità della parola. Tale probabilità viene stimata come la percentuale dei keypoint in un'ipersfera visuale.

2.1.2 Espansione del BoW utilizzando la relazione visuale

Sia V un vocabolario di n parole visuali: $V = (v_1, v_2, ..., v_n)$. Dato il vocabolario, un'immagine I può essere rappresentata come un vettore di feature $F_I = (w_{v_1}, w_{v_2}, ..., w_{v_n})$, dove w_{v_i} denota il peso della parola v_i nell'immagine. Basandosi sulla relazione visuale calcolata con JCN, viene realizzata un'espansione delle parole visuali diffondendo i pesi w_{v_i} della parola v_i a un'altra parola v_j:

$$w_{v_j} = w_{v_i} - w_{v_i} \times JCN(v_i, v_j) \times \alpha \tag{2.2}$$

dove α è un parametro per controlare il grado di influenza della relazione JCN tra le parole visuali. Lo scopo di questa espansione è la limitazione del problema

della correlazione tra parole visuali. Per completezza, si può affermare che il peso di una parola dipende dall'influenza di altre parole ontologicamente correlate.

2.2 Video event classification using string kernels

In questo articolo [MICC09] viene presentato un metodo per introdurre informazioni temporali nel riconoscimento di eventi nei video, utilizzando un approccio BoW. Gli eventi vengono modellati utilizzando una sequenza composta di istogrammi di feature visuali, calcolati per ogni frame con il tradizionale approccio BoW. Le sequenze vengono trattate come stringhe e ogni istrogramma viene considerato come un carattere. La classificazione di queste sequenze di lunghezza variabile, la cui lunghezza dipende dalla durata del filmato, viene realizzata tramite classificatori SVM con uno *string kernel* che utilizza la distanza *Needlemann-Wunsch*.

2.2.1 Rappresentazione e classificazione degli eventi

Dato un insieme di video annotati, lo scopo è l'apprendimento del modello degli eventi per categorizzare in maniera corretta nuovi video. Strutturalmente un evento viene rappresentato da una sequenza di frame, che potrebbe avere una lunghezza distinta a seconda dell'evento. Un evento viene modellato da una sequenza di vettori di frequenza delle parole visuali, calcolati a partire dai frame che costituiscono la sequenza contenente l'evento. Considerando ogni vettore di frequenza come un *carattere*, la sequenza di vettori viene detta *frase*. In aggiunta, viene definito un *kernel*, basato sull'*edit distance*, utilizzato con le SVM per gestire dati di ingresso di lunghezza variabile, a seconda del tipo di evento considerato.

I frame del video vengono rappresentati utilizzando i bag-of-words. Per prima cosa viene costruito un vocabolario visuale attraverso la quantizzazione di un grande insieme di descrittori di feature locali estratti da una raccolta di video. Sono stati utilizzati i *DoG* come rilevatori di keypoint e i *SIFT* come descrittori. Il vocabolario visuale (o *codebook*) viene generato clusterizzando i punti rilevati nello spazio delle feature. Per la clusterizzazione viene utilizzato il *k-mean clustering,* mentre la metrica di clustering è la *distanza Euclidea.* Il centro di ogni cluster risultante è detto *parola visuale.* Una volta che il vocabolario è stato definito, ogni keypoint rilevato nel frame viene assegnato ad un unico cluster. In questo modo, un frame viene rappresentato da un vettore di frequenza delle parole visuali che serve per memorizzare l'occorrenza di ciascuna parola visuale all'interno del frame.

Uno shot di un video viene rappresentato come una *frase* (stringa) formata dalla concatenazione della rappresentazione bag-of-words di *caratteri* (frame) consecutivi. Per confrontare queste *frasi*, e conseguentemente le azioni e gli eventi, possono essere adattate metriche definite nella teoria dell'informazione. Nello specifico, viene utilizzata la *edit distance* di *Needleman-Wunsch* [Needleman70].

2.2.2 Risultati sperimentali

Gli esperimenti sono stati condotti su video di calcio e sul dataset TRECVID 2005 che contiene video di notiziari. I risultati mostrano che la SVM con lo *string kernel* riesce a superare le prestazioni ottenibili con un classificatore KNN o con l'approccio standard BoW.

2.3 Scalable Recognition with a Vocabulary Tree

L'articolo [Nister06] presenta uno schema di riconoscimento di immagini scalabile che consente di lavorare efficientemente con un vasto numero di tipi di oggetti da riconoscere. Lo schema prevede l'utilizzo di descrittori locali quantizzati gerarchicamente in un vocabolario ad albero. I descrittori estratti da una raccolta di immagini vengono quantizzati in parole visuali, definite grazie ad un algoritmo *k-means* applicato al vettore di descrittori. L'insieme delle parole visuali viene utilizzato grazie ad uno schema *Term Frequency Inverse Document Frequency (TF-IDF)*. Gli autori propongono un sistema di assegnazione dei punteggi di tipo *TF-IDF* gerarchico utilizzando parole visuali definite gerarchicamente, in modo da formare un vocabolario ad albero.

2.3.1 Costruzione ed utilizzo del Vocabulary Tree

Il vocabolario ad albero definisce un meccanismo di quantizzazione gerarchica costruito tramite un clustering *k-means* gerarchico. In questo caso, k non indica il numero finale di cluster, bensì è il fattore di ramificazione (numero di figli di ogni nodo) dell'albero. La prima operazione è l'esecuzione di un processo di clusterizzazione *k-means*, in modo da definire k centri di cluster. Lo stesso processo viene applicato ricorsivamente ad ogni gruppo di vettori di descrittori, andando a suddividere tale gruppo in k nuove parti. L'albero viene determinato livello per livello, fino ad arrivare al massimo numero di livelli L.

Nella fase online, ogni vettore di descrittori viene semplicemente propagato attraverso l'albero con la seguente procedura: ad ogni livello il vettore di descrittori viene confrontato con i k centri di cluster candidati (rappresentati dai k figli nell'albero) e tra questi viene scelto quello più vicino. Questa è una modalità

molto semplice di realizzare k prodotti scalari ad ogni livello, per un totale di kL prodotti scalari.

In questo modo l'albero definisce direttamente sia il vocabolario visuale che la procedura di ricerca all'interno dello stesso.

2.3.2 Definizione del punteggio

Una volta definita la quantizzazione, vediamo come determinare la rilevanza di un database di immagini rispetto a un'immagine query. Questa procedura viene computata valutando la similarità tra i cammini nel vocabolario ad albero dei descrittori delle immagini contenute nel database e il cammino del descrittore dell'immagine query. Un problema da risolvere è l'assegnazione di un peso w_i ad ogni nodo i nel vocabolario ad albero. Questo peso deve essere basato sull'entropia e sia il vettore dell'immagine query q_i che il vettore dell'immagine del database d_i devono essere definiti in base ai pesi appena descritti:

$$q_i = n_i w_i \tag{2.3}$$

$$d_i = m_i w_i \tag{2.4}$$

dove n_i e m_i sono il numero di vettori di descrittori dell'immagine query e dell'immagine del database, rispettivamente, con un cammino attraverso un nodo i. Il punteggio di rilevanza s che valuta la similarità tra l'immagine query e quella del database, viene calcolato basandosi sulla differenza normalizzata tra i vettori delle stesse:

$$s(q, d) = ||\frac{q}{||q||} - \frac{d}{||d||}|| \tag{2.5}$$

La normalizzazione può essere eseguita in qualsiasi norma. Nel caso più semplice, i pesi w_i vengono settati a valori costanti, ma le prestazioni vengono

migliorate da un sistema di pesatura dell'entropia quale:

$$w_i = \ln \frac{N}{N_i} \tag{2.6}$$

dove N è il numero di immagini nel database e N_i è il numero di immagini nel database che possiedono almeno un vettore di descrittori con un cammino che passi attraverso il nodo i.

Attraverso prove sperimentali, gli autori hanno notato che per avere i migliori risultati di ritrovamento è importante avere un vocabolario vasto (un gran numero di nodi foglia) e non assegnare pesi particolarmente forti ai nodi interni del vocabolario ad albero.

2.3.3 Risultati sperimentali

L'approccio è stato testato realizzando query su un database con ground truth di 6376 immagini appartenenti a gruppi distinti costituiti da 4 immagini ciascuno. Ogni immagine del database viene utilizzata come immagine di query e le tre rimanenti immagini del suo gruppo dovrebbero essere idealmente collocate nelle posizioni migliori dei risultati della query. Gli autori mostrano i risultati per diverse configurazioni del sistema in cui valutano la percentuale di immagini realmente simili presenti tra le prime 3 dei risultati delle singole query. Le prestazioni migliori (90.6%) sono state ottenute con un numero di livelli della struttura pari a 6 e con un fattore di ramificazione di 10, per un totale di 1 milione di nodi foglia. Questi risultati sono stati ottenuti con un sotto insieme del database originale, composto da sole 1400 immagini. Se venissero aumentate le dimensioni del database le prestazioni decrescerebbero, visto che ci sarebbero più immagini che quindi porterebbero a maggior confusione.

2.4 Adaptive Vocabulary Forests for Dynamic Indexing and Category Learning

In questo paper [Yeh] viene descritto un algoritmo che sviluppa un insieme di vocabolari ad albero (foresta) calcolati con una modalità ad incremento e che si adattano all'aggiunta di nuove immagini al database. Questo tipo di struttura viene utilizzata per risolvere i problemi di indicizzazione e di riconoscimento di immagini.

2.4.1 Vocabolario a foresta adattivo

Gli autori hanno progettato un sistema adattivo che accresce incrementalmente la conoscenza immagazzinata in un singolo vocabolario ad albero, al crescere del numero di immagini inserite nel database. Per questo, è stato creato un sistema che accresca il vocabolario ad albero, elimini i nodi obsoleti, mantenga la rappresentazione piramidale e combini più alberi in una foresta per migliorare le prestazioni di riconoscimento delle immagini.

Prima di continuare nell'illustrazione del metodo creato, è necessario spendere alcune parole sulla rappresentazione piramidale. Il confronto piramidale tra vocabolari fu introdotto da [Grauman06]: è un metodo rapido per approssimare il confronto tra due insiemi di vettori di grandi dimensioni. Viene calcolato un punteggio di similarità tra due piramidi confrontando i punti in modalità *bottom-up* e pesando i match ad ogni livello della piramide in modo inversamente proporzionale, rispetto alla dimensione del bin. Per generare una piramide a partire da un vocabolario e da una nuova immagine, i punti dell'immagine vengono inseriti nell'albero in modo che ogni punto sia rappresentato con un cammino lungo il vocabolario ad albero. Per ognuno dei cammini nell'albero, viene creato un percorso parallelo nella piramide, in modo che la piramide

mantenga una rappresentazione sparsa e il confronto tra due piramidi viene realizzato con una scansione di ognuna delle piramidi. Seguendo questa procedura, due vettori di grandi dimensioni vengono confrontati in tempo lineare.

Accrescimento dei vocabolari ad albero Il metodo di accrescimento delle dimensioni dell'albero è costituito da tre passi:

1. Inserimento di un nuovo punto all'interno del vocabolario ad albero;
2. Rimozione di eventuali nodi sovraffolati in preparazione della ristrutturazione dell'albero;
3. Clusterizzazione dei punti eliminati al passo precedente e re-inserimento all'interno del vocabolario ad albero.

Eliminazione dei nodi obsoleti All'interno del vocabolario ad albero vengono continuamente inseriti nuovi punti che rappresentano feature estratte dalle immagini. Alcuni nodi dell'albero, però, possono rimanere inutilizzati per lunghi periodi di tempo. Visto che l'ambiente progettato è dinamico, questo indica che i nodi in questione sono obsoleti rispetto all'obiettivo corrente del problema. Esiste un limite superiore per il numero di foglie che un albero può possedere; se le dimensioni dell'albero crescono in modo che questo limite debba essere superato, allora vengono eliminate le foglie utilizzate meno recentemente.

Mantenimento della piramide Ogni bin in una piramide corrisponde ad un nodo nel vocabolario ad albero. Per questo, ogni nodo contiene una lista di puntatori ai bin delle piramidi a cui tale nodo corrisponde. Le modifiche al vocabolario, che corrispondono a variazioni nel sistema di rappresentazione piramidale, sono le seguenti:

1. Inserimento di un nuovo punto all'interno del vocabolario ad albero;

2. Rilascio o ristrutturazione di punti nel vocabolario ad albero.

Combinazione di Alberi La combinazione di singoli vocabolari ad albero porta due vantaggi importanti. In prima istanza, una foresta riduce gli effetti di quantizzazione in prossimità dei contorni dei nodi dei singoli alberi. Inoltre, si possono migliorare le prestazioni durante l'esplorazione di una nuova area dello spazio delle feature.

2.4.2 Risultati sperimentali

Per l'indicizzazione è stata utilizzata la stessa rappresentazione delle immagini e lo stesso metodo di assegnare i punteggi di [Nister06] (SIFT a 128 dimensioni estratti dalle regioni estreme massimalmente stabili).

Le prestazioni dell'algoritmo sono state analizzate nel contesto di problemi dinamici del mondo reale, come nel caso dei database online ai quali vengono aggiunte immagini, che gli utenti vogliono trovare, molto frequentemente. Per simulare un ambiente dinamico, ad ogni istante di tempo i, viene aggiunta un'immagine X_i al database, in ordine casuale. Quindi viene interrogato il database con l'immagine X_{i+1} e vengono recuperate le immagini maggiormente simili tra le i immagini già indicizzate dal sistema. I risultati di questa operazioni vengono considerati accurati quando le immagini che occupano le più alte posizioni di classifica appartengono allo stesso gruppo dell'immagine X_{i+1}(il dataset consiste di gruppi di immagini). A questo punto, l'immagine X_{i+1}viene inserita nel database e X_{i+2} è l'immagine query successiva.

Il sistema è stato testato anche per l'apprendimento di categorie utilizzando un classificatore SVM *one-vs-all* e misurando le prestazioni tramite la *mean*

recognition rate come in [Darrell06]. Anche il descrittore delle immagini è stato implementato similarmente a [Darrell06]: feature *10-d PCA-SIFT* concatenate con coordinate 2-d, campionate uniformemente su intervalli di 8 pixel.

In definitiva, l'utilizzo delle foreste rispetto ai singoli alberi migliora le prestazioni sia nel caso di ritrovamento delle immagini che nel caso di apprendimento delle categorie. Inoltre, l'adattamento dell'albero migliora ulteriormente i risultati rispetto all'utilizzo di un albero statico.

2.5 Video Event Detection Using Motion Relativity and Visual Relatedness

In questo articolo [Wang08] viene proposto un approccio per la rilevazione di eventi all'interno dei video. Un evento viene descritto da 2 aspetti:

What: cosa è coinvolto nell'evento. Viene descritto con feature globali (momento del colore, edge histogram) e feature locali (BoW).

How: come l'evento evolve nel dominio temporale. Viene descritto con feature semantiche (istogrammi di moto e mappe di vettori di moto).

Gli autori propongono una nuova feature di moto, detta **Expanded Relative Motion Histogram of Bag-of-Visual-Words (ERMH-BoW)**, per impiegare la relatività del moto e la correlazione visuale nella rilevazione di eventi. Il sistema si articola secondo le seguenti fasi:

1. Generazione del BoW per descrivere l'aspetto *What* dell'evento;
2. Calcolo di un istogramma di moto delle parole visuali che consente di sfruttare il moto locale, più discriminante per la rilevazione di eventi;

3. Calcolo della relazione tra gli istogrammi di moto delle parole visuali, per poter raffigurare l'attività di un evento;

4. Espansione del modello creato per considerare la correlazione tra parole visuali.

2.5.1 Relative motion histogram of BoW(RMH-BoW)

Il primo passo da realizzare è la costruzione del Bag-of-Visual-Word (BoW). Viene costruito un vocabolario visuale attraverso i vettori di quantizzazione calcolati a partire da un vasto insieme di descrittori di feature locali, estratti da un insieme di video. I descrittori utilizzati sono i DOG come *keypoint detector* e i SIFT come *keypoint descriptor*. Il vocabolario visuale viene generato clusterizzando i keypoint rilevati nello spazio delle feature utilizzando l'algoritmo *k-mean* e la *distanza euclidea* come metrica di clustering. Il centro di ogni cluster risultante è detto *parola visuale*. Una volta definito il vocabolario, ogni keypoint rilevato in un frame viene assegnato ad un unico membro del cluster, cioè ad una precisa parola visuale, in modo che ogni frame venga rapresentato da un istogramma di frequenza (un vettore) delle parole visuali.

Dopo aver generato il BoW, si passa alla creazione del Motion Histogram of Visual Words (MH-BoW). Il primo passo consiste nell'estrazione delle feature per ogni coppia di frame successivi. Quindi, vengono rintracciate le corrispondenze tra keypoint nei frame successivi e viene calcolato il vettore di moto m_p per ogni coppia di frame consecutivi. L'istogramma di moto viene generato sommando il vettore di moto di tutti i keypoint mappati nella stessa parola visuale. Per ogni parola visuale, quindi, viene costruito un istogramma a 4 dimensioni, considerando che il vettore m_p viene decomposto in 4 componenti corrispondenti alle 4 direzioni. Per una parola visuale v, l'istogramma viene costruito come:

$$H_v(i) = \sum_{p \in N_v} D_i(m_p)$$

$i = 1, 2, 3, 4$. D_i proietta m_p lungo l'i-esima direzione. Al termine di questa procedura si ha un vettore di feature a N dimensioni, detto Motion Histogram of BoW (MH-BoW) dove N è il numero di parole visuali e ogni elemento è un istogramma di moto di 4 dimensioni. In questo modo sono stati integrati gli aspetti *What* e *How* di un evento.

Il passo successivo consiste nella generazione dell'istogramma di moto relativo tra parole visuali (RMH-BoW) che risolve il problema della distorsione di moto, causato da movimenti variabili della camera. Per compiere questa operazione, viene analizzata la relatività di moto tra oggetti e scene distinte. Per ogni coppia di parole visuali, viene calcolato l'istogramma di moto relativo:

$$RH_i(v_a, v_b) = \sum_{p \in N_{v_a}, q \in N_{v_b}} D_i(m_p, m_q)$$

ove p e q sono due punti di interesse mappati, rispettivamente, nelle parole visuali v_a e v_b. Utilizzando questa notazione, le informazioni di moto di un video vengono rappresentate da una matrice R di dimensione $N \times N$ dove $R(i, j)$ è un istogramma di moto relativo tra due parole visuali i e j. Questa matrice è detta **Relative Motion Histogram of Bow (RMH-BoW)** e raffigura l'intensità e i pattern del moto relativo tra parole visuali distinte e può essere utilizzata per descrivere le attività e le interazioni tra oggetti e scene distinte di un evento. Intuitivamente, eventi distinti vengono presentati come oggetti distinti, in termini di pattern e di intensità di moto. RMH-BoW può essere quindi utilizzato come un sistema di apprendimento supervisionato per scoprire i pattern comuni in video distinti che contengono lo stesso evento.

2.5.2 Espansione di RMH-BoW con la relazione tra parole visuali (ERMH-BoW)

Gli autori dell'articolo propongono di cercare la correlazione tra parole visuali di RMH-BoW realizzando un'espansione che consenta di diffondere gli istogrammi di moto tra parole visuali correlate. La correlazione visuale è una misura della similarità tra parole visuali stimabile alla stregua della modalità con cui viene stimata la correlazione semantica tra parole testuali, usando l'ontologia generale di WordNet. Per realizzare questa operazione viene utilizzato l'algoritmo [Jiang08] di cui ho discusso nel paragrafo 2.1.

2.5.3 Modalità di rilevazione degli eventi e esperimenti realizzati

Dato un video clip, viene calcolato l'ERMH-BoW tra ogni coppia di frame consecutivi, in modo che la signature di un video sia una sequenza di ERMH-BoW. Per misurare la similarità tra video viene computata la *Earth Mover's Distance* (EMD) [Rubner00]. Durante il calcolo della EMD per la misura della similarità tra video, la distanza tra una coppia di frame appartenenti a due video distinti viene computata grazie alla *distanza euclidea* tra l'ERMH-BoW corrispondente ai due frame:

$$d(R_1^E, R_2^E) = \sqrt{\frac{1}{N^2} \sum_{1 \leq i,j \leq N} (R_1^E(i,j) - R_2^E(i,j))^2} \tag{2.7}$$

ove R_i^E è l'ERMH-BoW del frame i-esimo.

Dati due video clip

$$A = \left\{ (R_{A1}^E, w_{A1}), ... (R_{Am}^E, w_{Am}) \right\}$$

e

$$B = \left\{(R^E_{B1}, w_{B1}), ...(R^E_{Bn}, w_{Bn})\right\}$$

con m e n che sono le signature delle matrici, e $w_{Ai} = 1/m$ e $w_{Bj} = 1/n$ sono i pesi delle singole matrici ERMH-BoW, la distanza EMD tra A e B viene calcolata come:

$$D(A, B) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d(R^E_{Ai}, R^E_{Bj})}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}} \tag{2.8}$$

dove f_{ij}è il match ottimale tra due sequenze di matrici ERMH-BoW A e B. I dettagli del calcolo di f_{ij} possono essere trovati in [Xu07].

Con la distanza EMD tra video clip calcolata come nell'equazione 2.8, viene adottato l'algoritmo in [Xu07] per addestrare le SVM per la rilevazione di eventi. La distanza EMD tra video clip viene incorporata nella funzione di *kernel* del framework SVM, utilizzando la seguente funzione Gaussiana:

$$K(A, B) = \exp(-\frac{1}{kM} D(A, B)) \tag{2.9}$$

ove il fattore di normalizzazione M è la media della distanza EMD tra tutti i video clip di training, mentre k è un fattore di scala deciso empiricamente dalla *cross-validation*.

Per la fase di sperimentazione è stata utilizzata una parte del dataset TRECVID2005 [TRECVID] costituita da 14 dei 24 eventi che compongono l'intero dataset. Questi eventi sono rappresentati in circa 40000 video, dei quali circa il 50% è stato utilizzato per la fase di training, mentre il rimanente 50% è stato utilizzato per la fase di testing. Per la valutazione delle prestazioni è stata utilizzata l'*Average Precision (AP)* [TRECVID] che è risultata essere del 25.96%.

2.6 Content based video matching using spatio-temporal volumes

Questo articolo [Basharat08] presenta un approccio per il confronto tra sequenze video utilizzando la segmentazione spazio-temporale dei video. Viene utilizzata la traiettoria dei punti di interesse per generare i volumi dei video. La similarità tra video viene analizzata rilevando regioni importanti (*foreground* e *background*) nella scena, estraendo le feature meno sensibili a variazioni e infine sviluppando una tecnica di corrispondenza tra volumi che gestisca il confronto parziale tra video. Le maggiori fasi di progettazione sono le seguenti:

- **Estrazione del volume del video**: utilizzando le corrispondenze tra punti SIFT in frame successivi, vengono identificate le traiettorie, poi raggruppate in cluster in base alla similarità e alla prossimità spaziale. Al termine del procedimento, ogni volume comprende regioni di movimento indipendenti tra loro.
- **Confronto tra video utilizzando le feature del volume**: da ogni volume viene estratto un set di feature che comprende il colore, la tessitura, il moto e descrittori SIFT. La similarità tra le feature viene calcolata sulla base della *Earth Mover's Distance* (EMD) [Rubner00]. Due video vengono confrontati costruendo un grafo bipartito, dove i volumi sono rappresentati dai vertici e la loro similarità è il peso assegnato agli archi.

2.6.1 Estrazione dei volumi spazio-temporali

Generazione della traiettoria Per prima cosa il video viene suddiviso in shot. Quindi, per ognuno dei frame dello shot, vengono estratti i punti SIFT a 128 dimensioni (in questo modo ogni punto di interesse è rappresentato da un

vettore a 128 dimensioni). Il passo successivo consiste nell'identificazione delle corrispondenze tra punti di interesse in ogni coppia di frame consecutivi, tramite la procedura presente in [Lowe04]. I punti corrispondenti vengono poi uniti per generare le traiettorie; eliminando però le traiettorie con troppo spiazzamento rispetto alla media.

Assumendo un modello di velocità uniforme, vengono fuse traiettorie incomplete per poterne generare di più potenti. Per fare ciò, ogni traiettoria viene proiettata in avanti (cioè nell'immagine successiva) e viene realizzata una nuova ricerca di punti SIFT applicando una finestra spazio-temporale. Se viene trovato l'inizio di una nuova traiettoria, allora le singole traiettorie vengono fuse.

Rilevazione delle regioni Per comprendere il moto di un oggetto da un frame al successivo viene utilizzata un'omografia basata su RANSAC. In base a questo procedimento viene scelto il segmento con il moto dominante per generare una graduatoria dei segmenti. Per ogni regione rilevata vengono estratte le feature utilizzando informazioni sparse (descrittori SIFT e di moto) e dense (colore e tessitura).

Tracking delle regioni A questo punto, ogni frame possiede un set di segmenti, ciascuno con i suoi descrittori. Per risolvere il problema della corrispondenza tra frame, viene utilizzata la traiettoria comune massima tra le regioni, basandosi sul fatto che le traiettorie multiple presenti che partono da una regione forniscono molti vincoli per tracciare la regione nei frame successivi. Partendo da un set di regioni etichettate del primo frame si effettua una propagazione delle etichette attraverso frame successivi, utilizzando una finestra temporale. La propagazione delle etichette viene effettuata anche all'indietro, partendo dall'ultimo frame (procedura *split-and-merge*).

2.6.2 Estrazione delle feature del volume

Una volta che i volumi sono disponibili, vengono estratte le feature poi utilizzate per realizzare il confronto tra i volumi stessi. Le feature utilizzate sono le seguenti:

- punti SIFT-128d,
- colore (HSV)-3d,
- tessitura (Canny Edge Detector)-8 direzioni nel range [0,2Π],
- moto (utilizzo delle traiettorie dei punti interni al volume)-direzione del moto quantizzata in 8 dimensioni nel range [0,2Π].

2.6.3 Confronto tra video basato sul volume e risultati sperimentali

Le parti distinte della scena vengono catturate dai volumi i cui contenuti sono rappresentati dai corrispondenti set di features; per questo un video viene rappresentato come un insieme di volumi. Per calcolare la similarità tra due video vengono calcolate le similarità tra ogni coppia di volumi, per ogni tipo di feature utilizzata. Tali valori di similarità vengono poi combinati per costruire un singolo valore di similarità tra due volumi. Quindi viene costruito un grafo bipartito con i volumi come vertici e tali valori di similarità come peso degli archi. A questo punto, la corrispondenza tra volumi viene valutata selezionando i valori massimi presenti nel grafo bipartito (algoritmo di Kuhn Munkres [Plummer86]).

Gli esperimenti sono stati realizzati su un dataset di 337 video ottenuti da TRECVID 2005 [TRECVID], video online scaricati da Google Video[GoogleVideo] e

da BBC Motion Gallery[BBC]. Ci sono quattro categorie principali di oggetti presenti in questi video, cioè barche, automobili, aeroplani e carri armati. In definitiva viene ottenuto un valore di *average precision* di circa 70% dove:

$$Precision = \frac{\{SimilarVideos\} \cap \{\ RetrievedVideos\}}{\{RetrievedVideos\}}$$

$$AveragePrecision = \frac{\sum Precision(r)\delta(r)}{\{SimilarVideos\}}$$

dove δ è la funzione binaria sulla rilevanza della graduatoria r. I risultati vengono valutati sui 10 video con la miglior posizione in graduatoria, cioè sui 10 video suggeriti come simili per ognuno dei video di ingresso.

2.7 ShotTagger: Locating Tags in Video

Questo articolo [ShotTagger] presenta un metodo per assegnare automaticamente i tag di livello video al livello dei segmenti. L'approccio si sviluppa secondo due passi; per prima cosa viene analizzata la distribuzione dei tag all'interno del video, basandosi su un framework di apprendimento ad istanze multiple. Per risolvere i problemi di ambiguità e di imperfezione dei tag associati al video viene analizzata la relazione semantica tra tag co-occorrenti. La seconda fase di lavorazione prevede uno smoothing temporale tra shot adiacenti, per raffinare i risultati del processo di *tagging* interno al video.

2.7.1 Annotazione dei tag all'interno del video

Per modellare la relazione tra il video e gli shot viene utilizzato un algoritmo di regressione logistica ad istanze multiple (MILR) [Jiang09]. L'obiettivo è quello di stimare la probabilità condizionale di uno shot di essere associabile ad un

tag t, dato il contenuto dello shot. Uno shot viene analizzato estraendo un key frame dallo shot stesso. Per ogni key frame è estratta una feature SIFT a 128 dimensioni [Lowe04]. Questa feature viene poi quantizzata in un bag of words (BoW) a 1111 dimensioni utilizzando metodi di clustering gerarchico [Nister06]. Infine, tramite l'applicazione di una tecnica PCA viene ridotta la dimensione della feature BoW a sole 200 dimensioni.

Nel framework MILR, viene utilizzata la regressione logistica per modellare la probabilità condizionale di uno shot rispetto ad un tag t:

$$V_{ij} = P_t(y_{ij} = 1|f_{ij}) = \frac{1}{1 + \exp(-(\sum w_n \cdot f_{ij,n} + b))} \tag{2.10}$$

dove la probabilità condizionale da stimare è indicata con $P_t(y_{ij} = 1|f_{ij})$. f_{ij} rappresenta il vettore di feature dello shot j-esimo all'interno del video i-esimo. $f_{ij,n}$ è l'n-esima dimensione del vettore delle feature e w_nè l'n-esima dimensione del vettore dei pesi W associato al vettore delle feature. Il parametro b è un termine di bias. La coppia di parametri (W, b) è sconosciuta e deve essere calcolata. Tali parametri vengono determinati tramite una procedura di ottimizzazione che consiste nella minimizzazione di una funzione obiettivo, che assume la forma della funzione errore quadratica $E_t(W, b) = \sum(Y_i - V_i)^2$, dove $E_t(W, b)$ è il valore che deve essere minimizzato e Y_i indica l'occorrenza del tag t per il video i, e quindi vale 0 o 1. Dopo aver stimato la coppia di parametri, $P_t(y_{ij} = 1|f_{ij})$ può essere calcolata basandosi sull'equazione (2.10). La stima della coppia di parametri viene eseguita utilizzando metodi di ottimizzazione basati sul gradiente.

Il passo successivo consiste nell'applicazione di una procedura che valuti la conoscenza esterna dei tag co-occorrenti applicando dei vincoli di ottimizzazione con lo scopo che i video non risultino positivi rispetto a tutti i tag originali che

possiedono. La premessa su cui si basa questa operazione è che se molti tag di un video sono semanticamente correlati, allora si incrementa la possibilità di questo video di avere degli shot rilevanti. Per calcolare la similarità dei tag di un video, viene utilizzato il motore di ricerca *Google* realizzando una query per ognuno dei tag e valutando la *cosine similarity* tra i migliori n frammenti di articolo risultanti da ciascuna query.

2.7.2 Smoothing temporale delle annotazioni dei tag a livello shot

Dato che shot adiacenti solitamente hanno un contenuto semantico simile e quindi potrebbero essere annotati con gli stessi tag, viene utilizzato lo smoothing temporale per raffinare i risultati della procedura suddetta. In questo approccio viene utilizzato il metodo dei *nearest neighbor* (vengono considerati gli shot adiacenti) per modificare il punteggio di rilevanza dello shot corrente.

2.7.3 Valutazione delle prestazioni

Visto che non esiste un dataset di video *YouTube* che contenga i video e gli shot corrispondenti già annotati, gli autori hanno creato un ampio dataset di video realizzando diverse query da *YouTube*. Le query selezionate per il download sono basate sugli eventi annotati LSCOM [LSCOM] e su alcuni concetti di alto livello che si trovano all'interno delle categorie. Gli autori hanno effettuato il controllo manuale per identificare gli eventi considerati all'interno degli shot. In definitiva sono stati isolati 23 tag e sono stati utilizzati 1224 video per un totale di 49154 shot. La percentuale di eventi riconosciuti è del 39%, risultato ottenuto con un metodo di apprendimento a istanze multiple basato sul contesto con regressione logistica e smoothing temporale.

2.8 Tag Suggestion and localization in user-generated videos based on social knowledge

Da questo articolo [MICC10] sono state tratte idee molto interessanti per lo svolgimento del mio progetto di tesi. Viene presentato un sistema per il suggerimento dei tag di un video e per la localizzazione temporale degli stessi all'interno delle sequenze video. L'algoritmo suggerisce nuovi tag da associare ad un keyframe analizzando le annotazioni di video e di immagini che si possono trovare in siti come *YouTube* e *Flickr*.

2.8.1 Studio della rilevanza dei tag per l'annotazione di video

L'approccio proposto in questo articolo ha due scopi: l'estensione del numero di tag associati ad ogni video e l'associazione di tag agli shot rilevanti che compongono il video.

L'annotazione di video viene realizzata in due passi. Nella prima fase viene calcolata la rilevanza dei tag per ogni shot, andando però ad eliminare i tag non rilevanti, successivamente ad ogni shot vengono aggiunti nuovi tag. Dopo aver estratto gli shot dal video, vengono estratti tre keyframe per ogni shot. I tag associati al video vengono utilizzati per selezionare e scaricare da *Flickr* un set di immagini che sono state annotate con i tag di partenza del video. L'unione di tutti i tag delle immagini scaricate da *Flickr* costituisce il dizionario di tag con cui possono essere annotati gli shot del video. Dato che le immagini di *Flickr* sono state annotate da principianti, è fondamentale valutare la rilevanza dei termini che compongono il lessico, per evitare che siano aggiunte annotazioni non corrette. A questo scopo è stato adattato l'algoritmo per la valutazione

della rilevanza dei tag presente in [Li09] per poter funzionare con l'annotazione degli shot dei video. Il calcolo della rilevanza di un tag si basa sul numero di volte in cui il tag compare nei *k-nearest neighbours* dell'immagine sottratto della frequenza a priori del tag stesso. Questo richiede un calcolo efficiente dei *k-nearest neighbours*, dal punto di vista visuale, per il keyframe. Le immagini di cui viene valutata la similitudine rispetto al keyframe sono quelle scaricate da *Flickr* in base ai tag del video di partenza.

Per rappresentare l'informazione visuale dei keyframe e delle immagini, viene calcolata una feature visuale a 72 dimensioni che contiene informazioni globali di colore e tessitura. Il vettore è composto da un correlogramma di colore a 48 dimensioni calcolato nello spazio di colore HSV, da 6 dimensioni date dal momento di colore calcolato nello spazio di colore RGB e da un vettore a 18 dimensioni per tre feature di Tamura che raccolgono informazioni sulla tessitura (le feature considerate sono la granularità, il contrasto e la direzionalità). Alle feature delle immagini scaricate da *Flickr* viene applicato un algoritmo di *k-mean* clustering. Per ciascuno dei keyframe viene identificato il cluster rappresentato dalla parola visuale più vicina e le immagini che appartengono a quel cluster vengono selezionate come vicine.

Per ottenere risultati migliori, un tag viene mantenuto nella lista associata ad un'immagine solo se presente tra i tag delle immagini più vicine. Inoltre viene condotta un'espansione tramite *WordNet* mirata ad espandere i tag da associare a ciascuno shot. Il modo di calcolare la rilevanza di un tag è quello dato dall'algoritmo $Vote^{+}$proposto in [Li09]. Questo punteggio viene utilizzato per ordinare i tag da associare ad uno shot; i cinque tag più rilevanti vengono associati allo shot.

2.8.2 Risultati sperimentali

Le prestazioni sono state valutate utilizzando un dataset progettato per rappresentare la varietà dei contenuti di *YouTube*. Il dataset è stato creato scegliendo 4 video di *YouTube* da ognuna delle 14 categorie di *YouTube*. In questo modo il numero di shot rilevati è 1135, per un totale di 3405 keyframe analizzati. Per ogni tag associato ad un video di *YouTube*, sono state scaricate le prime 15 immagini da *Flickr* in base al criterio della rilevanza fornito dalle API di *Flickr*. Inoltre, il sistema scarica da Flickr 5 immagini aggiuntive per ognuno dei sinonimi aggiunti grazie all'espansione realizzata con *WordNet*. La valutazione delle prestazioni è stata realizzata considerando l'*accuracy*; calcolata come la proporzione dei veri positivi rispetto al numero totale di veri e falsi positivi. Negli esperimenti le prestazioni sono state valutate in termini di:

STL (Shot level tag identification): valutazione delle prestazioni della localizzazione dei tag a livello shot. Questa misura mostra l'accuratezza della localizzazione dei tag negli shot considerando solo i tag iniziali del video di *YouTube*. Il valor medio di *accuracy* ottenuto è 0.63.

STSL (Shot level tag suggestion and localization): valutazione delle prestazioni della localizzazione dei tag a livello shot sia per i tag generati dagli utenti (originali) che per quelli suggeriti. In questo caso il valor medio di *accuracy* è 0.35.

STSL-WN (STSL with WordNet query expansion): misura delle prestazioni di STSL con l'espansione dei tag grazie ai sinonimi ottenuti da *WordNet*. Il valor medio di *accuracy* è 0.36.

2.9 TOP-SURF: a visual words toolkit

In questo articolo [LIACS10] viene presentato un descrittore di immagini che combina i punti di interesse con le parole visuali. L'obiettivo degli autori di questo articolo è l'identificazione delle immagini simili all'interno di una raccolta di milioni di immagini. Su un insieme di dati così vasto non risulta affidabile, in termini di tempi di esecuzione e di capacità di memoria, l'applicazione di un metodo SURF che vada ad estrarre i punti SURF da tutte le immagini e poi imposti un confronto tra tutti i punti SURF. Questo è uno dei motivi che ha spinto gli autori a creare questo tipo di descrittore, che riesce a ridurre significativamente la dimensione del descrittore. Per calcolare il descrittore TOP-SURF di un'immagine sono necessari diversi passi di esecuzione.

2.9.1 Punti di interesse rappresentativi

Gli autori hanno utilizzato un set ampio di distinte immagini di training composto da un milione di immagini scaricate da Internet, un milione di immagini scaricate da Flickr e 3000 immagini di città scattate dagli stessi autori. Per ognuna di queste immagini vengono estratti i punti di interesse SURF e vengono scelti casualmente 25 punti.

2.9.2 Clusterizzazione in parole visuali

Per raggruppare l'intera collezione di punti di interesse rappresentativi in un numero limitato di cluster viene utilizzato un approccio basato sulla tecnica dei bag-of-words presentata in [Philbin07]. Per identificare la locazione iniziale di ciascun cluster, viene scelto casualmente uno dei punti di interesse e questi viene assegnato al cluster. A questo punto, per ogni cluster vengono determinati

i suoi 100 *nearest neighbours*, cioè i 100 punti di intersse a lui più vicini. Se un punto SURF risulta tra i vicini di più di un cluster, questo viene assegnato solo al cluster a cui è più vicino. Infine vengono modificati tutti i centri dei cluster assegnando loro il valore della media tra la locazione corrente e i suoi 100 vicini. Per assicurare stabilità ai cluster questo processo viene ripetuto 1000 volte.

I cluster finali sono il *dizionario di parole visuali*. Lo scopo degli autori è l'enfatizzazione delle parole visuali non presenti molto frequentemente, dato che queste possono essere considerate più descrittive se trovate in un'immagine. Per assegnare un peso alle parole visuali identificate viene utilizzata una tecnica di tipo *tf-idf weighting* [Salton83].

2.9.3 Selezione delle parole visuali più descrittive

Dopo aver presentato la tecnica di costruzione delle parole visuali, può essere presentato il metodo di calcolo del descrittore TOP-SURF di un'immagine. Per prima cosa vengono estratti i punti SURF da un'immagine; questi punti vengono poi convertiti in un istogramma di frequenza delle parole visuali presenti nell'immagine, determinando la parola visuale più simile ad ogni punto di interesse. Quindi viene applicato un metodo *tf-idf weighting* per assegnare un punteggio a tutte le parole visuali nell'istogramma. Per formare il descrittore di immagine proposto nell'articolo, vengono infine selezionate le parole visuali con punteggio più alto. Dato che vengono utilizzate solo le N parole visuali più presenti, il descrittore viene chiamato TOP-SURF.

2.9.4 Confronto tra immagini

Per confrontare i descrittori TOP-SURF di due immagini viene calcolata la *cosine similarity* normalizzata d_{cos} tra i loro istogrammi *td-idf* T_A e T_B:

$$d_{cos} = 1 - \frac{T_A \cdot T_B}{|T_A| \cdot |T_B|} \tag{2.11}$$

Una distanza di 0 significa che i descrittori sono identici, mentre una distanza di 1 indica che essi sono completamente differenti. Il vantaggio di questo approccio è proprio la velocità del confronto tra due immagini che viene realizzato semplicemente comparando un piccolo numero di parole visuali. Di contro, con un semplice metodo SURF, per confrontare due immagini, ogni punto di interesse di un'immagine deve essere ricercato all'interno dell'altra immagine realizzando un confronto con tutti i punti SURF della seconda immagine. Questo procedimento richiede tempi di esecuzione nettamente superiori a quelli necessari al confronto tra parole visuali.

2.10 Learning Social Tag Relevance by Neighbor Voting

In questo articolo [Li09], a partire da un'immagine, viene proposto un algoritmo di votazione che apprende la rilevanza dei tag accumulando voti dai vicini visuali dell'immagine di partenza.

L'idea di base di questo metodo suggerisce che data un'immagine, la rilevanza di ognuno dei suoi tag può essere inferita verificando il numero di volte che il tag in questione compare all'interno del set di immagini simili. In base a questo, maggiore è la frequenza di un tag all'interno del set, maggiore sarà la sua rilevanza. Non sempre i tag che hanno una frequenza molto elevata sono buoni; il rischio è quello di attribuire una rilevanza alta a quei tag che in realtà sono troppo generali. Una buona misura della rilevanza deve perciò considerare

sia la distribuzione all'interno del set di immagini simili che quella all'interno dell'intera collezione.

2.10.1 L'apprendimento della rilevanza dei tag

E' necessario introdurre una notazione per trattare il caso in esame. Si denota con Φ una collezione di immagini dotate di *user-tag* e con W un vocabolario di tag utilizzato in Φ. Per un'immagine $I \in \Phi$ ed un tag $w \in W$, sia $r^*(w, I) : \{W, \Phi\} \longmapsto \mathbb{R}$ una misura della rilevanza del tag. Tale misura di rilevanza soddisfa i seguenti 2 criteri:

- **Image Ranking**: date due immagini $I_1, I_2 \in \Phi$ ed un tag $w \in W$, se w è rilevante rispetto a I_1 ma irrilevante rispetto a I_2, allora

$$r^*(w, I_1) > r^*(w, I_2) \tag{2.12}$$

- **Tag Ranking**: dati due tag $w_1, w_2 \in W$ ed un immagine $I \in \Phi$, se I è rilevante rispetto a w_1 ma irrilevante rispetto a w_2, allora

$$r^*(w_1, I) > r^*(w_2, I) \tag{2.13}$$

Lo scopo dell'algoritmo è trovare una misura di rilevanza dei tag che soddisfi i due criteri.

2.10.2 L'utilizzo dei visual neighbors per il calcolo della rilevanza

Si deve studiare come le immagini rilevanti ed irrilevanti per un tag sono etichettate con quel tag. In un database di immagini dotate di *user-tag*, è possibile che

per un tag specifico w, il numero di immagini irrilevanti rispetto a w sia significativamente maggiore del numero di immagini rilevanti, ovvero $\mid R_w^c \mid \gg \mid R_w \mid$, dove $\mid \bullet \mid$ è l'operatore di cardinalità sul set di immagini.

Sia $L_w \subset \Phi$ il set di immagini della collezione etichettate col tag w. Può essere fatta un'assunzione riguardo al comportamento del tagging: "in un database di grandi dimensioni, la probabilità di tagging corretto è superiore rispetto a quella di tagging errato". Si analizza la distribuzione di immagini rilevanti ed irrilevanti rispetto a w all'interno del set dei *k-nearest neighbors*. Nel primo scenario, la ricerca visuale si comporta come il campionamento casuale ed il numero di immagini rilevanti nel neighbor set coincide con quello delle immagini rilevanti all'interno del set estratto casualmente. Nel caso in cui la ricerca visuale sia migliore rispetto al campionamento casuale, date due immagini I_1 rilevante per w ed I_2 irrilevante per w, ci si aspetta di ottenere:

$$\mid N_f\left(I_1, k\right) \cap R_w \mid > \mid N_{rand}\left(k\right) \cap R_w \mid > \mid N_f\left(I_2, k\right) \cap R_w \mid \tag{2.14}$$

dove f rappresenta una funzione di similarità tra due immagini, basata su feature di basso livello. Il numero di immagini rilevanti per il tag w all'interno del neighbor set è espresso come:

$$\mid N_f\left(I, k\right) \cap R_w \mid = k \cdot \left(P\left(R_w\right) + \epsilon_{I,w}\right) \tag{2.15}$$

dove $\left(P\left(R_w\right) + \epsilon_{I,w}\right)$ è la probabilità che un immagine selezionata casualmente dal neighbor set sia rilevante rispetto a w. A questo punto è necessario fare una seconda assunzione: "una ricerca visuale basata sul contenuto è migliore rispetto ad un campionamento casuale". Dividendo il neighbor set in due distinti sottoset $N_f\left(I, k\right) \cap R_w$ e $N_f\left(I, k\right) \cap R_w^c$, si conta il numero di occorenze di w all'interno

dei due sottoset:

$$n_w\left[N_f\left(I,k\right)\right] = n_w\left[N_f\left(I,k\right) \cap R_w\right] + n_w\left[N_f\left(I,k\right) \cap R_w^c\right] =$$

$$= k \cdot \left(P\left(R_w\right) + \epsilon_{I,w}\right) P\left(w \mid R_w\right) + k \cdot \left(P\left(R_w^c\right) - \epsilon_{I,w}\right) P\left(w \mid R_w^c\right) \tag{2.16}$$

In modo simile si può derivare:

$$n_w\left[N_{rand}\left(k\right)\right] = k \cdot \left(P\left(R_w\right) P\left(w \mid R_w\right) + P\left(R_w^c\right) P\left(w \mid R_w^c\right)\right) \tag{2.17}$$

La quantità $n_w\left[N_{rand}\left(k\right)\right]$ riflette la frequenza di occorrenza di w nell'intera collezione di immagini, e può quindi essere denotata con $Prior\left(w,k\right)$. Sostituendo (2.17) in (2.16) si ottiene:

$$n_w\left[N_f\left(I,k\right)\right] - Prior\left(w,k\right) := k \cdot \left(P\left(w \mid R_w\right) - P\left(w \mid R_w^c\right)\right) \epsilon_{I,w} \tag{2.18}$$

In definitiva, la rilevanza del tag w rispetto all'immagine I ed al set dei suoi k neighbors si definisce come:

$$tagRelevance\left(w,I,k\right) := n_w\left[N_f\left(I,k\right)\right] - Prior\left(w,k\right) \tag{2.19}$$

2.10.3 Neighbor Voting Algorithm

L'espressione (2.19) esprime come la rilevanza del tag w dipenda dal numero delle sue occorrenze nei k *nearest neighbors* dell'immagine I e dalla frequenza a priori del tag stesso. Considerando che ogni *nearest neighbor* vota circa la presenza/assenza del tag w, $n_w\left[N_f\left(I,k\right)\right]$ indica il numero di voti dei "vicini" sul tag w. Si introduce quindi un *neighbor voting algorithm*: in primo luogo viene effettuata una ricerca visuale per identificare i vicini di un'immagine corredata

di tag, quindi vengono usati i tag di ogni vicino per votare sui tag dell'immagine di riferimento. Si approssima la frequenza a priori del tag w come:

$$Prior\,(w,k) \approx k\frac{\mid L_w \mid}{\mid \Phi \mid} \tag{2.20}$$

dove k è il numero di *visual neighbors*. $\mid L_w \mid$ indica il numero di immagini etichettate col tag w, mentre $\mid \Phi \mid$ è la dimensione dell'intera collezione. Si può notare come l'espressione (2.19) non restituisce necessariamente valori di rilevanza positivi; per questa ragione il valore minimo di $tagRelevance\,(w,I,k)$ è impostato a 1. In altre parole, se il valore di rilevanza di un user-tag è minore della sua frequenza originale all'interno dell'immagine alla quale appartiene, si "rigetta" tale quantità.

In aggiunta, è importante notare come i risultati della procedura fin qui descritta siano influenzati dal numero delle immagini di ogni singolo utente all'interno del neighbor set. Per rendere più oggettivo il *neighbor voting algorithm*, quindi, si impone il vincolo di *unique-user*, in modo che un utente abbia al massimo 1 immagine all'interno del neighbor set. I risultati sperimentali mostrano come tale vincolo sia capace di ridurre il bias dell'algoritmo di voting.

2.10.4 Gli esperimenti

L'algoritmo proposto è stato valutato nel caso di *Image Ranking* e *Tag Ranking*. Nel primo caso, vengono comparati 3 metodi di image retrieval basati sui tag nel caso di presenza/assenza dell'algoritmo basato sulla rilevanza. Nel secondo caso, vengono valutate le potenzialità dell'algoritmo rispetto al task di fornire un ausilio al "tagging" svolto dagli utenti. Facciamo un cenno al dataset attraverso il quale gli autori hanno valutato il metodo fin qui descritto.

Nel caso di *Image Ranking*, vengono selezionati 20 concetti visuali come query:

per ogni query sono selezionati casualmente 1000 esempi (immagini etichettate) all'interno di una collezione composta da 3.5 milioni di foto estratte da *Flickr*. Tali esempi vengono ri-etichettati secondo il criterio di rilevanza descritto nelle precedenti sezioni, dopo di che vengono classificate le 1000 immagini di test con 2 metodi baseline e con l'algoritmo proposto. Nel caso di *Tag Ranking*, invece, viene adottato il dataset proposto da [vanZwol08]. Il set di foto consiste di 331 immagini estratte da *Flickr*, senza sovrapposizioni all'interno della collezione di 3.5 milioni di foto. Tutte le 331 immagini vengono utilizzate come test.

In definitiva, per gli esperimenti *Image Ranking* gli autori hanno riscontrato un miglioramento del 24.3% rispetto ai metodi baseline basati sui tag. Per il *Tag Ranking*, invece, sono stati considerati due tipi di esperimento: suggerimento di tag per immagini già annotate e suggerimento di tag per immagini non ancora taggate. In entrambi i casi, i risultati mostrano che l'utilizzo dei tag delle immagini visualmente simili favorisce la qualità dei suggerimenti.

Capitolo 3

Ricerca ed indicizzazione di video simili

In questo Capitolo viene descritto il sistema realizzato per il **ritrovamento e l'indicizzazione** di video simili. Per prima cosa, il concetto di similitudine tra due video viene inteso come appartenenza degli stessi alla medesima categoria semantica. Per questo due video vengono ritenuti simili, ad esempio, quando rappresentano lo stesso avvenimento sportivo (partita di calcio, partita di basket, ecc..), la stessa trasmissione televisiva o, più in generale, lo stesso evento.

Il primo obiettivo del progetto è la realizzazione di un sistema per la classificazione di filmati tramite un descrittore (signature) per ogni frame. In questo modo un filmato è composto da una successione di descrittori che devono essere utilizzati per la rilevazione della similitudine. Per realizzare il secondo obiettivo, invece, è stata adattata una struttura di indicizzazione, già progettata per l'elaborato di Database Multimediali [DBMul09], per un accesso rapido a vaste

quantità di dati. Applicazioni di questo genere rientrano nell'ambito del **video matching** e tipicamente si sviluppano in due passi:

- La prima fase viene realizzata offline e consiste (eventualmente) nel segmentare i video e nell'estrarre una signature (descrittore) per ogni segmento. Queste informazioni vengono poi memorizzate in un database;
- La fase successiva consiste nel memorizzare i vari descrittori creati per i video e stabilire quali siano i filmati simili tramite un confronto tra descrittori.

Il problema della rilevazione dei video simili rimane una sfida aperta. In letteratura sono presenti approcci basati sul contenuto e molti di questi si occupano di identificare i video tramite signature globali. I metodi basati sui punti locali si sono dimostrati particolarmente affidabili per l'identificazione di video contenenti lo stesso evento, anche con variazioni piuttosto complesse. Purtroppo, le potenzialità di queste tecniche si vanno a scontrare con problemi di confronto e scalabilità, data la necessità di modellare gli eventi da ricercare. Altri lavori hanno portato a buoni risultati combinando le signature globali derivate da istogrammi di colore e i punti locali (tipicamente SIFT o SURF) che vengono elaborati confrontando i frame a coppie. Sfortunatamente, questo tipo di punti locali rimane proibitivo da considerare, dato il gran numero di coppie di frame all'interno di un video e dato il gran numero di punti locali che vengono rintracciati all'interno di ogni singolo frame. Per queste ragioni si ottengono rilevazioni accurate a discapito di costi elevati in termini di tempi di esecuzione, soprattutto per database molto estesi come quelli di video nel web. Per questo tipo di applicazioni, il tempo di computazione diviene una limitazione piuttosto importante nel perseguire l'obiettivo della ricerca di video simili in tempo reale. Per questo si rende necessaria, oltre all'estrazione di punti locali per rappresen-

tare un frame, anche la creazione di strutture per l'indicizzazione rapida di vaste quantità di video, in modo da non dover confrontare tutti i filmati presenti a coppie, che porterebbe a un'esplosione dei tempi di esecuzione di un sistema che debba funzionare su vaste quantità di dati.

Tenendo in considerazione tutte le informazioni suddette, il sistema proposto prevede la creazione di una **foresta di vocabolari visuali** realizzati secondo la tecnica Bag-of-Visual Words (BoW), utilizzando il descrittore SURF come feature. Quindi, per ogni frame di un video, si procede al calcolo del descrittore. Il descrittore di frame proposto è formato da un numero intero di 6 cifre, di cui 5 rappresentano informazioni rigurdanti la distribuzione dei punti SURF secondo l'approccio BoW, mentre 1 cifra tiene traccia della distribuzione di colore dell'immagine, valutata secondo la componente H della rappresentazione di colore HSV. Tramite la procedura di calcolo del descrittore, ogni frame di un video, quindi, viene rappresento da un numero intero di 6 cifre. Ne consegue che l'intero video è sintetizzato da una successione di interi a 6 cifre che sono i descrittori distinti dei singoli frame che compongono il video. Dopo aver calcolato i descrittori per tutti i video all'interno di un database dove devono essere rintracciati i video simili, viene costruita la struttura di indicizzazione CFP-Tree grazie alla quale si recuperano le corrispondenze tra video, basandosi sulla percentuale di frame etichettati con lo stesso valore tra video distinti.

Il sistema creato riceve in ingresso un database di video, all'interno del quale devono essere identificati i gruppi di video che rappresentano il medesimo ambito semantico. Schematizziamo le fasi principali che compongono il sistema creato:

1. Creazione della foresta di vocabolari di parole visuali basati sul descrittore SURF;

2. Per ogni frame di un video, calcolo del descrittore, formato da un intero

di 6 cifre;

3. Una volta calcolati i descrittori di frame di tutti i video che compongono il database, viene utilizzata la struttura di indicizzazione CFP-Tree per recuperare le relazioni tra video.

Entriamo ora nel dettaglio di ciascuna fase di progettazione realizzata.

3.1 Creazione della foresta di vocabolari

Prima di entrare nel dettaglio delle operazioni realizzate, chiariamo le modalità di utlizzo della tecnica Bag-of-Visual Words nell'ambito dei contenuti visuali. Da un insieme di video o di immagini, si possono estrarre dei keypoint locali (SURF, SIFT, etc...). Dopo aver raccolto tutti i keypoint, viene applicato un algoritmo di clustering in modo che i keypoint siano raggruppati in un certo numero di cluster. Ogni cluster viene trattato come una **parola visuale** che va a far parte del vocabolario di parole visuali.

Dopo aver creato il vocabolario, si passa all'analisi dei video estraendo i medesimi keypoint da ogni frame del video in esame. Tali keypoint vengono mappati nelle parole visuali, cioè ogni singolo keypoint viene associato alla parola visuale più simile, in modo che un'immagine possa essere rappresentata come un vettore di feature che tenga conto della presenza, oppure conteggi il numero di occorrenze, di ogni parola visuale. Questa rappresentazione del BoW è analoga alla rappresentazione bag-of-words per i documenti testuali. Con questo tipo di rappresentazione, ogni singola immagine è sintetizzata da un vettore che descrive l'immagine tramite l'occorrenza della parole visuali del vocabolario all'interno dell'immagine stessa.

In questo progetto i punti salienti utilizzati sono stati i SURF alla cui descrizione viene dedicato ampio spazio nella sezione A.2. Più precisamente è stata utilizzata una **foresta di vocabolari ad albero**. Un singolo vocabolario ad albero è originato da un processo di quantizzazione gerarchica, ottenuto applicando gerarchicamente il *k-means clustering* sull'insieme di punti SURF estratti da un database di video utilizzato appositamente per raccogliere punti SURF. Nel caso di un vocabolario ad albero, il valore k (cioè il numero di cluster in cui viene suddiviso l'intero database tramite l'applicazione dell'algoritmo k-means) non è il numero finale di cluster, bensì definisce il fattore di ramificazione dell'albero. Per prima cosa, quindi, viene applicato l'algoritmo k-means sull'intero insieme di dati di training, definendo k cluster, rappresentati da altrettante parole visuali. A questo punto, i dati di training vengono partizionati in k gruppi, dove ciascun gruppo contiene i punti SURF la cui parola visuale più simile tra le k create, è quella identificativa del gruppo stesso. Lo stesso procedimento suddetto viene applicato ricorsivamente ad ogni gruppo di vettori di descrittori, dividendo ricorsivamente ogni gruppo in k nuove parti. Tramite questa procedura, l'albero viene determinato livello per livello, fino al massimo numero di livelli L, ed ogni divisione in k nuove parti viene definita solo dalla distribuzione del vettore di descrittori (punti SURF) che appartiene al genitore della cella di quantizzazione.

Nel progetto realizzato è stato fissato il fattore di ramificazione $k = 10$ e il numero di livelli $L = 3$, come si può osservare in Figura 3.1.

Nel sistema realizzato sono stati utilizzati tre alberi distinti, costruiti con la tecnica appena descritta, in modo da formare una foresta di vocabolari ad albero. I tre alberi sono stati costruiti con tre insiemi diversi di video. Il motivo che ha spinto a realizzare una foresta è che in questo modo si poteva variegare maggiormente l'informazione, altrimenti contenuta in un solo albero. In questo modo,

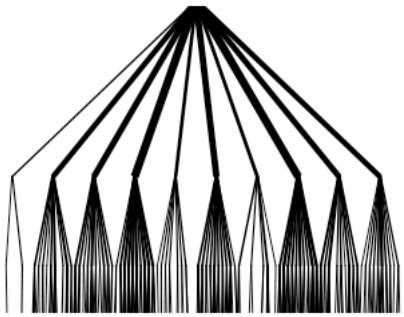

Figura 3.1: Tre livelli di un vocabolario ad albero con fattore di ramificazione $k = 10$ popolato per rappresentare un'immagine con 400 feature. Lo spessore della ramificazione corrisponde al numero di feature all'interno dell'immagine che hanno seguito quella ramificazione. Qualche ramo è assente ed indica che è stato intrapreso da nessuna feature.

infatti, se il descrittore di due frame risulta lo stesso per tutti e tre i vocabolari, le possibilità che quei due frame siano realmente simili aumenta rispetto al caso in cui i due siano simili rispetto al descrittore calcolato su un singolo albero. L'idea è che data un'immagine, ogni vocabolario ad albero possa suggerire una lista di immagini candidate. Tramite la foresta, vengono fuse queste liste in una singola lista in modo che la combinazione di più alberi riesca a migliorare le prestazioni di riconoscimento dei singoli vocabolari ad albero.

3.1.1 Software realizzato

Dopo aver descritto il metodo applicato, vediamo come sono state realizzate le operazioni suddette. Per completare questa fase, sono stati realizzati due distinti progetti in linguaggio C++. Il primo di questi riceve in ingresso un insieme di video, per ciascuno video cattura tutti i frame che lo compongono e da questi vengono estratti i punti SURF a 64 dimensioni. I dettagli implementatitvi dell'estrazione dei punti SURF si possono trovare nella Sezione 3.2.3. Al termine della procedura, viene creato un file che contiene tutti i punti SURF a

64 dimensioni trovati. Questo algoritmo è stato lanciato su tre distinti insiemi di dati, uno per ciascun albero che deve essere creato.

Un'osservazione interessante riguarda le modalità di creazione dei database. Per questa operazione non sono necessarie particolari tecniche, basta racogliere un certo numero di video per avere abbastanza punti SURF su cui costruire il vocabolario.

Con il primo progetto C++, quindi, sono stati realizzati i tre insiemi di punti SURF da cui ricavare i tre distinti vocabolari ad albero che vanno a formare la finestra. Con un secondo progetto C++, invece, si realizza il vocabolario ad albero vero e proprio. Questo algoritmo viene lanciato una volta per ciascun insieme di punti SURF di ingresso, quindi un totale di 3 volte.

Ricevuto in ingresso l'insieme di vettore di descrittori (ogni punto SURF è un vettore a 64 dimensioni), il programma applica l'algoritmo *k-means cluster* per suddividere l'intero insieme di dati in $k = 10$ gruppi, rappresentati da altrettante parole visuali. A questo punto, si associa ad ogni vettore di SURF la parola visuale a cui tale punto è più simile. La similitudine viene valutata tramite la *distanza Euclidea*. Il passo successivo è l'applicazione ricorsiva dell'algoritmo k-means sui gruppi di dati isolati al passo precedente. In questo modo si ottengono altre $k \times k$ parole visuali, collegate in gruppi di k al proprio genitore, per formare il vocabolario finale.

La funzione utilizzata per il k-means appartiene alla libreria *opencv* ed è la seguente:

```
double kmeans(const Mat& samples, int clusterCount, Mat& labels, TermCriteria
termcrit, int attempts, int flags, Mat* centers)
```

Tale funzione consente di trovare i centri dei cluster e raggruppare i dati di ingresso intorno ai cluster. Si può passare all'analisi dei singoli parametri:

samples: è una matrice di floating-point dei dati di ingresso. Contiene una riga, di 64 elementi, per ogni dato di ingresso.

clusterCount: è il numero di cluster in cui suddividere l'insieme di dati.

labels: è il vettore di interi che memorizza gli indici dei cluster a cui appartengono i singoli dati di ingresso.

termcrit: specifica il massimo numero di iterazioni e/o l'accuratezza, cioè la distanza di cui si possono muovere i centri dei cluster tra un'iterazione e la successiva.

attempts: quante volte l'algoritmo viene eseguito utilizzando differenti etichettature iniziali. L'algoritmo ritorna le etichette che garantiscono il miglior valore di compactness.

flags: può assumere i seguenti valori:

- KMEANS_RANDOM_CENTERS: ad ogni iterazione i valori dei centri dei cluster vengono selezionati casualmente;
- KMEANS_PP_CENTERS: per l'inizializzazione dei centri dei cluster viene utilizzato l'algoritmo *kmeans++* di Arthur e Vassilvitskii;
- KMEANS_USE_INITIAL_LABELS: durante il primo (e possibilmente unico) tentativo, la funzione utilizza le etichette fornite dall'utente invece di calcolarle a partire dai centri iniziali. Per il secondo e per i successivi tentativi, la funzione utilizza centri scelti casualmente o semi-casualmente.

centers: è la matrice di output dei centri dei cluster, una riga per ogni cluster.

La funzione kmeans implementa un algoritmo di k-means che trova i centri di clusterCount cluster e raggruppa i dati di ingresso intorno ai cluster. In uscita, la

funzione fornisce labels$_i$ che è l'indice del cluster a cui è stata assegnata l'i-esima riga della matrice di ingresso.

La funzione restituisce un parametro che è la misura di *compactness* ed è calcolata come:

$$\sum_i ||samples_i - centers_{labels_i}||^2$$

Questo valore viene calcolato dopo ogni tentativo; viene scelto il miglior valore (quello minimo) e vengono ritornate le corrispondenti etichette e il corrispondente valore di compattezza.

In questo progetto la funzione è stata utilizzata con il parametro flag settato a KMEANS_PP_CENTERS, in modo da utilizzare l'algoritmo di Arthur e Vassilvitskii. Inoltre, il valore di attempts è stato settato a 3.

3.2 Calcolo del descrittore

Si ricordi che ogni frame di un video viene rappresentato con un numero intero di sei cifre. L'intero video è quindi identificabile dalla successione di valori assunti dai propri frame. Delle sei cifre che compongono il descrittore di frame, cinque contengono informazioni riguardanti la distribuzione dei punti SURF nell'immagine, mentre una tiene conto della distribuzione della componente H della rappresentazione di colore HSV all'interno dell'immagine. Entriamo ora nel dettaglio delle operazioni effettuate per il calcolo delle singole componenti del descrittore.

3.2.1 Distribuzione dei punti SURF

Come già scritto in precedenza, cinque delle sei cifre che compongono il descrittore di frame, provengono dalla distribuzione dei punti SURF all'interno del frame stesso. Più precisamente, due cifre provengono dall'analisi della distribuzione dei punti SURF nel primo vocabolario ad albero, altre due cifre dal secondo vocabolario, mentre dall'ultimo vocabolario deriva un solo valore. Viene ora descritta la procedura per l'analisi dei primi 2 vocabolari visuali; essendo analoga per tutti e due i vocabolari viene presentata facendo riferimento ad uno solo di essi.

Dato un frame che compone il video, vengono estratti tutti i punti SURF a 64 dimensioni rintracciabili nel frame stesso. Quindi, per ognuno dei frame estratti viene ripetuta la seguente procedura. Il vettore di descrittori a 64 dimensioni che rappresenta un singolo punto SURF viene propagato attraverso il vocabolario visuale ad albero. Il descrittore SURF viene confrontato con le k parole visuali che rappresentano le k ramificazioni in cui è suddiviso il primo livello del vocabolario ad albero. Tra queste k parole visuali, viene scelta quella più vicina al descrittore SURF, in termini di *distanza Euclidea*. A questo punto, la procedura viene ripetuta sulle k parole visuali che discendono dal ramo prescelto al passo precedente.

Per ogni frame del video è stato creato un array di $k \times k$ elementi per ognuno dei primi due vocabolari ad albero. Questo array viene utilizzato per memorizzare la foglia in cui vanno a cadere i punti SURF nell'immagine. Infatti, per ogni punto SURF, viene incrementato di un'unità il valore della posizione dell'array che corrisponde alla foglia in cui termina il punto SURF. La procedura utilizzata per convertire ciascuna foglia in un indice numerico è molto semplice e viene illustrata di seguito. Le parole visuali sono k per cui possono essere rappresentate con indici numerici che vanno da 0 a $k - 1$. Per cui, valendo $k = 10$, il

valore numerico che rappresenta una foglia si ottiene con $l_1 \times 10 + l_2$ dove l_1 è il valore dell'indice di primo livello corrispondente alla parola visuale più simile al descrittore del punto SURF, mentre l_2 è l'analogo per la parola visuale di secondo livello, scelta tra le k parole visuali che discendono dalla parola prescelta al primo livello. In questo modo, essendo i valori di l_1 e l_2 compresi tra 0 e $k - 1 = 9$, si ha che ogni foglia viene rappresentata con un indice univoco che varia da 0 a 99.

Dopo aver ripetuto la procedura su tutti i punti SURF del frame, l'array in questione contiene la distibuzione dei punti SURF rispetto al vocabolario visuale ad albero. A questo punto, la scelta è stata quella di rappresentare l'intero frame con la parola visuale più presente e quindi più rappresentativa del contenuto informativo dell'immagine. Entrando nel dettaglio, il frame viene rappresentato con un numero intero a 2 cifre, che però non è il valore della foglia più presente. Infatti, la prima cifra è data dalla parola visuale di primo livello che ha avuto più riscontri, e la seconda è data dalla parola visuale di secondo livello più presente, tra le k parole che discendono dalla parola visuale di primo livello selezionata al passo precedente. Molte volte, il valore calcolato con questa procedura è diverso rispetto a quello della foglia più presente in assoluto, ma con questa modalità viene rappresentata meglio la distribuzione dei punti SURF rispetto ad un vocabolario ad albero.

L'intera procedura viene effettuata parallelamente anche per il secondo vocabolario visuale, mentre per il terzo vocabolario i descrittori SURF vengono confrontati solo con le k parole visuali del primo livello in modo che alla fine il frame venga rappresentato con un numero intero che sintetizza la parola visuale col maggior numero di riscontri.

Come si può notare dalla descrizione fornita, ogni descrittore SURF viene semplicemente propagato attraverso gli alberi che rappresentano i vocabolari. Per

due dei tre alberi, ad ogni livello dell'albero la propagazione avviene confrontando il vettore dei descrittori con i k centri dei cluster (rappresentati da k figli nell'albero) e scegliendo quello più vicino in base alla *distanza Euclidea*. Questa è un modo semplice di realizzare k prodotti vettoriali ad ogni livello, risultando in un totale di $k(L-1)$ prodotti vettoriali. Questa procedura si dimostra efficiente, soprattutto se il valore di k non è molto alto. In questo progetto, avendo settato $k = 10$, si ottengono tempi di calcolo abbastanza ristretti ed inoltre il cammino lungo l'albero può essere codificato da un singolo valore intero, quindi facilmente utilizzabile in un sistema di indicizzazione dei risultati.

Può essere interessante osservare che l'albero definisce direttamente sia il vocabolario visuale che una procedura di ricerca efficiente. Questo non sarebbe accaduto nel caso in cui non si fosse realizzato il vocabolario visuale gerarchicamente, caso in cui si sarebbero dovuti realizzare più confronti per determinare la similitudine tra il descrittore SURF e tutte le parole visuali che avrebbero rappresentato i cluster in cui sarebbe stata suddivisa l'immagine. Inoltre, l'approccio gerarchico offre maggiore flessibilità nell'assegnazione dei punteggi per le parole visuali più presenti e consente di realizzzare la procedura descritta precedentemente, tramite la quale viene premiata la parola visuale di primo livello più presente in combinazione con quella più presente di secondo livello scelta tra i figli della parola di primo livello stessa.

3.2.2 Utilizzo della componente H della rappresentazione HSV

Come già scritto in precedenza, una delle 6 cifre che compongono il descrittore, si ottiene dalla distribuzione della componente H all'interno dell'immagine. Si è utilizzato questo valore per dare anche informazioni di colore al descrittore di frame, senza però che queste fossero troppo dettagliate, visto che l'obiettivo

non è la realizzazione di un approccio per l'identificazione dei *nearest-neighbour*, bensì devono essere trovati i video appartenenti alla stessa categoria semantica.

Il calcolo per l'identificazione della distribuzione dell'immagine viene effettuato solo sulla parte centrale dell'immagine. Suddividendo l'immagine in quattro parti uguali su entrambi i lati della stessa, si ottiene l'area da analizzare andando a prendere i pixel contenuti solo all'interno dell'incrocio tra le due parti centrali, lungo i due lati dell'immagine, in cui è stata suddivisa l'immagine stessa. Ad ogni pixel contenuto in questa area, viene applicata la procedura descritta di seguito. Avendo i punti dell'immagine in rappresentazione RGB, la procedura serve per passare allo spazio di colore HSV. Siano r, g e $b \in [0,1]$ le coordinate rossa, verde e blu del pixel in uno spazio di colore RGB. Sia max il valore massimo tra r, g e b e min il minimo tra questi. Per il calcolo di $h \in [0,360]$ dello spazio HSV la procedura è la seguente:

$$h = \begin{cases} 0 & \text{if.} max = min \\ (60^{\text{o}} \times \frac{g-b}{max-min} + 360^{\text{o}}) \text{mod} 360^{\text{o}} & \text{if.} max = r \\ 60^{\text{o}} \times \frac{b-r}{max-min} + 120^{\text{o}} & \text{if.} max = g \\ 60^{\text{o}} \times \frac{r-g}{max-min} + 240^{\text{o}} & \text{if.} max = b \end{cases}$$

Per il calcolo di s e v, si procede nel seguente modo:

$$\begin{cases} 0 & \text{if.} max = 0 \\ \frac{max-min}{max} = 1 - \frac{min}{max} & \text{altrimenti} \end{cases}$$

$$v = max$$

Dopo aver descritto lo spazio HSV, analizziamo le operazioni effettuate per rappresentare con un solo numero il valore di H di un singolo pixel. I 360^{o} gradi

del dominio di H sono divisi in sei fasce di colore, come si può osservare nella parte sinistra di Figura A.8. La fascia del colore rosso sta a cavallo degli 0 gradi, cioè va da 0^o a 30^0 e da 330^o a 360^o. Per fare in modo che questa fascia sia continua, al valore di H che si ottiene viene sommato 30 in modulo 360, cioè:

$$H = (H + 30) \mathrm{mod} 360$$

A questo punto si esegue la seguente operazione:

$$(H/360.001) \times 6$$

Con la divisione si porta il valore di H nell'intervallo [0,1]. Moltiplicando il valore ottenuto per 6 si ottiene un valore compreso tra 0 e 5 che corrisponde ad una delle 6 fasce di colore. Per distinguere anche i casi di bianco, grigio e nero si effettuano dei controlli sui valori R, G e B e si assegna il valore 6 nel caso di bianco, 7 per il nero e 8 se il pixel ha un colore corrispondente ad una tonalità di grigio.

Il procedimento appena descritto viene ripetuto per tutti i pixel contenuti nell'area di interesse. Il valore intero che va a completare il descrittore è dato dall'indice della componente più presente. Questo numero intero, quindi, assume un valore tra 0 e 8, a seconda del valore di H più presente.

3.2.3 Struttura del programma

Il programma per l'estrazione dei descrittori di un video è un thread che viene richiamato da un programma esterno che ha il compito di lanciare in parallelo fino a 3 thread. Entriamo ora nel dettaglio del thread per l'estrazione dei descrittori. Questo programma riceve in ingresso il nome di un file che è il video

che deve essere analizzato. La prima operazione che viene effettuata è l'estrazione di 4 frame per ogni secondo di filmato e l'inserimento di questi frame in un vettore di dati di tipo Video, una classe creata per la modellazione di un video. Tra le funzioni della classe, la più rilevante è quella per il calcolo del vettore dei descrittori. Tale funzione chiama un metodo per il calcolo del descrittore sul singolo frame. Quest'ultimo metodo appartiene alla classe Descrittore, creata per rappresentare il descrittore di frame e per sviluppare i metodi per il calcolo dello stesso. Questa funzione va poi a richiamare i metodi per il calcolo del descrittore, suddivisi in due diverse funzioni. Una delle due funzioni effettua il calcolo della distribuzione dei punti SURF basandosi sui tre vocabolari visuali ad albero costruiti, mentre la seconda funzione effettua il calcolo del valore dominante della componente H della rappresentazione di colore HSV.

Entriamo ora nel dettaglio delle classi realizzate.

Classe per la rappresentazione di un video Questa classe serve per memorizzare le informazioni relative ad un video, analizziamone la struttura:

```
classVideo{
public:
FrameNos *f;        //vettore di oggetti frame
int numFrame;    //numero di frame
int *descr;    //vettore di descrittori
int numDescr;    //numero di descrittori
int altezza;
int larghezza;    // dimensioni di altezza e larghezza
// i cui valori vengono impostati nel thread
void azzera(){}
int scriviSuFile(){}
```

```
Video(){}
~Video(){}
int calcolaVettoreDescrittoriSURF(){}
}
```

Il primo attributo della classe, *f*, è un puntatore ad un vettore di tipo *FrameNos*. Questa è una classe, che in seguito analizzeremo nel dettaglio, per modellare un singolo frame. Con questo attributo viene dichiarato il vettore in cui vengono memorizzati tutti i frame che compongono il video (con un campionamento di 4 frame al secondo). Il secondo attributo, *numFrame*, è un intero che tiene traccia del numero di elementi che compongono il vettore di frame, mentre il terzo, *descr*, è un vettore di interi in cui ogni elemento è il descrittore di un frame. Nonostante venga calcolato un descrittore per ciascun frame del video, il numero di descrittori è diverso dal numero di frame, visto che per qualche frame potremmo avere lo stesso valore del descrittore. Il sistema, infatti, memorizza solo descrittori distinti, senza ripetizioni; per cui alla fine un video è trattato come una successione di descrittori interi a 6 cifre, diversi l'uno dall'altro. Un'altra considerazione interessante è che non conta l'ordine in cui compaiono i descrittori.

Gli attributi successivi, *altezza* e *larghezza*, sono i valori che rappresentano il formato del video, espresso in pixel. Passiamo all'analisi delle funzioni della classe. La prima di queste, *azzera()*, serve per azzerare gli attributi della classe, mentre la seconda, *scriviSuFile()*, è il metodo per aggiungere informazioni ai 3 file di testo che verranno letti dal programma per la creazione dell'albero CFP e che servono per memorizzare tutte le informazioni relative ai video analizzati. Viene inserita una nuova riga nel file *correspond.dat* contenente il numero identificativo del video e il nome del video. Con questo metodo viene scritta una nuova riga anche nel file *data.dat* in cui viene scritto il codice del video e poi

la sequenza di numeri che rappresentano i descrittori dei frame calcolati per il video. Questa funzione aggiorna anche il file *num.dat* che contiene il numero di codice del prossimo filmato che verrà analizzato e serve per poter assegnare un codice progressivo ai video che vengono elaborati. Le funzioni successive sono il costruttore e il descrittore che si occupano, rispettivamente, di allocare e deallocare la memoria per gli attributi della classe.

La funzione *calcolaVettoreDescrittoriSURF()* riveste una particolare importanza perchè si occupa del calcolo dei descrittori per i frame che compongono il video. In questa funzione vengono memorizzati i centri dei cluster, cioè le parole visuali, dei tre vocabolari ad albero creati precedentemente. Inoltre, nella funzione viene dichiarato un puntatore ad un tipo *DescrittoreSURF* che è il vettore di descrittori del frame. A questo punto vengono richiamati i metodi della classe *DescrittoreSURF* per calcolare il descrittore sul singolo frame e per calcolare il valore hash che consente di utilizzare un numero intero a 6 cifre per rappresentare il descrittore.

Classe per la rappresentazione del descrittore Durante lo svolgimento del progetto, è stata scritta una classe per la computazione di ogni tipo di descrittore realizzato, ma in questo documento viene riportato solo quello della versione finale del programma. In questo metodo è stato utilizzato un oggetto della classe *DescrittoreSURF* che è una classe ereditata dalla classe *Descrittore*. Infatti è stata implementata la classe *Descrittore* che contiene gli attributi e i metodi comuni al calcolo di tutti i tipi di descrittore. Da questa classe, poi, deriva una classe per ogni tipo di descrittore implementato. Di seguito viene presentato il codice della classe *Descrittore*.

```
class Descrittore{
public:
```

```
int *vettComp;  //vettore di componenti
int numEl;      //numero elementi
int hash;  //valore della funzione hash
virtual int calcolaHash()=0;
void calcolaDescrittore(FrameNos *f);
};
```

Da questa classe derivano, con un meccanismo di ereditarietà, tutte le classi per realizzare i metodi *calcolaHash( )* e *calcolaDescrittore(FrameNos *f)* per i singoli tipi di descrittore. L'ereditarietà è stata realizzata nel seguente modo:

```
class DescrittoreSURF : public Descrittore{...};
```

Il metodo *calcolaDescrittore(FrameNos *f)* della classe *DescrittoreSURF* riceve in ingresso un frame del video e vengono chiamate due funzioni per il calcolo dei descrittori. La prima delle due funzioni si occupa di analizzare la distribuzione dei punti SURF nel frame. Vengono estratti tutti i punti SURF e se ne analizza la distribuzione in base alla foresta di vocabolari ad albero, come spiegato nella sottosezione 3.2.1. In base alla distribuzione dei punti SURF, vengono riempiti alcuni campi della classe *FrameNos* per la memorizzazione delle parole visuali a cui si vengono associati i singoli punti SURF, in base alla *distanza Euclidea*. La seconda funzione chiamata, invece, calcola la distribuzione della componente H della rappresentazione di colore HSV, secondo la modalità descritta nella sottosezione 3.2.2.

L'altra funzione membro della classe *Descrittore*, la *calcolaHash()*, viene utilizzata per calcolare il numero intero di 6 cifre che rappresenta il descrittore di frame, sfruttando il lavoro del metodo *calcolaDescrittore(FrameNos *f)*.

Classe per la rappresentazione del frame Infine analizziamo la classe *FrameNos* che serve per modellare un singolo frame.

```
class FrameNos{
public:
IplImage *imm;
int classificaBin1[NUMCLUSTERSE];
int classificaBin2[NUMCLUSTERSE];
int classificaBin3[NUMCLUSTERPR];
int classificaH;
}
```

Vengono presentati solo gli attributi membro, senza specificare anche i metodi. Il primo attributo, *imm*, è il puntatore all'immagine rappresentata nel frame. *classificaBin1* e *classificaBin2* sono due vettori di NUMCLUSTERSE elementi. In questo caso NUMCLUSTERSE è fissato a 100 ed è il numero di foglie degli alberi che rappresentano i due vocabolari visuali a più livelli che sono stati memorizzati. Questo vettore viene riempito incrementando di un'unità le locazioni del vettore che rappresentano la parola visuale a cui è più simile ciascun punto SURF estratto. *classificaBin3* è un vettore di NUMCLUSTERPR elementi, cioè vale 10, visto che il terzo vocabolario visuale è costituito da un solo livello. *classificaH*, invece, è un intero utilizzato per memorizzare la componente H più presente nell'immagine, normalizzata tra 0 e 8, secondo il procedimento descritto nella sottosezione 3.2.2.

Funzione per l'estrazione dei punti SURF La funzione utilizzata per l'estrazione dei punti appartiene alle librerie *opencv* e il suo prototipo è il seguente:

void cvExtractSURF(const CvArr* image, const CvArr* mask, CvSeq** keypoints, CvSeq** descriptors, CvMemStorage* storage, CvSURFParams params)

Si procede all'analisi dei parametri della funzione:

image è l'immagine di ingresso a 8 bit in scala di grigi.

mask è la maschera di ingresso opzionale. Le feature vengono cercate solo nelle aree che contengono più del 50% dei pixel della maschera diversi da zero.

keypoints è il parametro di output; è un doppio puntatore alla sequenza di keypoints. La composizione della struttura CvSURFPoint è la seguente:

```
typedef struct CvSURFPoint {
        CvPoint2D32f pt; // posizione della feature
                         // all'interno dell'immagine
        int laplacian; // -1, 0 o +1. Segno del
                       // laplaciano nel punto.Può
                       //essere utilizzato per
                       //velocizzare il confronto
                       //tra feature
        int size; // dimensione della feature
        float dir; // orientazione della feature: 0..360
                   //gradi
        float hessian; // valore dell'Hessiana
} CvSURFPoint;
```

descriptors è un parametro opzionale di output; è un doppio puntatore alla sequenza di descrittori. A seconda del valore assunto dal parametro params.extended, ogni elemento della sequenza è un vettore di floating point a 64 o a 128 dimensioni. Se il parametro vale NULL, i descrittori non vengono calcolati.

storage è la locazione di memoria dove i keypoints e i descriptors vengono memorizzati.

params sono vari parametri dell'algoritmo messi a disposizione dalla struttura CvSURFParams, la cui composizione è la seguente:

```
typedef struct CvSURFParams{
        int extended; // 0 calcola i descrittori base
                      //(64 elementi ciascuno),
                      // 1 calcola i descrittori estesi
                      //(128 elementi ciascuno)
        double hessianThreshold; // vengono estratte
                    //solo le feature con keypoint.hessian
                    //più grande di questo valore. Buoni
                    //valori di default sono solitamente
                    //compressi tra 300 e 500.
int nOctaves; // numero di ottave usate per l'estrazione
              //(3 di default)
int nOctaveLayers; // numero di livelli interni ad ogni
                   //ottava (4 di default)
}CvSURFParams;
     CvSURFParams cvSURFParams(double hessianThreshold,
         int extended=0);//ritorna i parametri di default
```

La funzione è stata chiamata anche con il parametro di output opzionale, descriptors, che è il doppio puntatore alle componenti dei descrittori utilizzate per assegnare il singolo punto SURF a una parola visuale. Sono stato computati punti SURF a 64 dimensioni, quindi params.extended è stato settato a zero.

3.3 Creazione della struttura di indicizzazione

3.3.1 Modalità di utilizzo del CFP Tree

In questo paragrafo verranno descritte le modalità con cui è stato realmente costruito l'albero CFP, basandosi sul lavoro già svolto durante l'elaborato di Database Multimediali [DBMul09]. Per prima cosa illustriamo nel dettaglio le motivazioni che hanno spinto ad utilizzare la struttura CFPTree come metodo di indicizzazione. L'algoritmo estrae un descrittore (un numero a 6 cifre) per ogni frame di un filmato, per cui un video potrebbe essere visto come una sequenza di numeri, uno per ciascun frame del video. A questo punto si è pensato di utilizzare il CFPTree costruendolo a partire da un database con le righe impostate secondo un indice invertito rispetto ai frame contenuti all'interno del filmato. Infatti ciascun frame può essere etichettato con un numero che varia da 000000 a 999999, per cui il database di transazioni sarà composto da 1000000 righe. A questo punto ad ogni riga del database così creato vengono associati i codici dei video che contengono questo determinato frame. In questo modo si hanno una serie di corrispondenze che tengono traccia, dato il codice identificativo di un frame, di quali video lo contengono. Costruendo la struttura CFP sull'insieme di dati appena creato, si ottengono tutte le corrispondenze tra i filmati. Infatti, partendo da un elemento del primo nodo dell'albero e seguendo il percorso che da lui viene generato, si potranno osservare tutti gli elementi che compaiono nella stessa riga dell'elemento di partenza, con un supporto che scende scorrendo il percorso verso il basso. In questo caso gli elementi sono i video (rappresentati tramite il proprio codice identificativo) per cui creando l'albero CFP e scorrendolo verso il basso si identificano tutte le corrispondenze tra video. Per come viene creato l'albero CFP, il primo nodo contiene tutti gli elementi contenuti nel database (un numero di volte maggiore del supporto minimo) ordinati in

ordine crescente di supporto. Il supporto minimo per questa prima fase è stato fissato volontariamente basso in modo da selezionare la maggior parte dei video presenti nel database. Più precisamente tale valore è stato fissato a 15, così non vengono considerati solo i video con supporto minore di 15, cioè che non contengono più di 15 frame identificati con codici diversi. A questo punto, scorrendo verso il basso gli alberi che iniziano da ciascun elemento del primo nodo si identificano i filmati contenuti in quello iniziale. Tra questi vengono considerati solamente quelli che superano un supporto minimo che è diverso da quello utilizzato precedentemente. Infatti questo è un **supporto relativo percentuale** e viene calcolato rispetto al supporto dell'elemento del primo nodo dal quale è partito il sotto albero. La struttura dell'albero CFP ampiamente discussa nella sezione A.7 non è perfetta per questo tipo di problema, per cui sono state apportate delle modifiche al CFPTree. Lo scopo principale per il quale sono utilizzati i CFPTree è quello di ricercare le corrispondenze in un database di transazioni. Per questo tipo di applicazione è importante sviluppare la struttura CFP su più livelli in modo da poter ritrovare le corrispondenze tra le transazioni. Questa applicazione, però, deve avere una funzionalità diversa. Per come è strutturato l'albero CFP, il primo nodo contiene i vari elementi in ordine crescente di supporto minimo, e ogni sottoalbero che parte da un elemento contiene solo elementi con un supporto minore o uguale all'elemento radice del sottoalbero. Nella mia applicazione gli elementi del CFPTree sono filmati, per cui se consideriamo il primo elemento del primo nodo vuol dire che stiamo trattando il filmato che ha il minor supporto tra tutti quelli che superano il supporto minimo. Seguendo il sottoalbero di cui questo video è radice, incontreremo tutti gli altri video che hanno dei frame in comune al video di partenza. Per realizzare questa funzione nasce la prima differenza rispetto alla struttura base dell'albero CFP. Nel CFPTree vengono memorizzati i soli elementi che

hanno un supporto superiore a quello minimo prefissato e a questo punto non vengono più eliminati dalla struttura. Nel mio progetto è stato scelto un supporto minimo iniziale volutamente molto basso in modo da poter selezionare ed inserire nella struttura CFP il maggior numero possibile di filmati, che avessero comunque una durata accettabile. A questo punto nell'applicazione entra in gioco un secondo supporto minimo che è valutato in percentuale rispetto alla radice di ogni sottoalbero. Infatti un elemento viene mantenuto nell'albero CFP solo se il rapporto tra il suo supporto e il supporto della radice del sottoalbero in cui è contenuto supera un valore percentuale stabilito. In questo modo si lasciano nella struttura solamente i video che hanno una parte in comune che superi una percentuale stabilita con il video radice del sottoalbero.

Con la metodologia appena illustrata si vanno a eliminare dalla struttura dell'albero CFP alcuni elementi. Per come è realizzato l'albero CFP, l'elemento che si vuole eliminare potrebbe avere a sua volta dei figli. Questi però non possono essere eliminati perchè potrebbero essere dei video che in realtà si trovano anche in qualche altra diramazione del sottoalbero e sommando tutti i supporti nelle varie diramazioni potrebbero superare la percentuale di supporto minimo. Per questo motivo è stato deciso di sviluppare un albero CFP su due livelli soltanto. Il primo nodo dell'albero viene costruito come nella versione base dell'albero CFP e contiene tutti gli elementi con un supporto maggiore di quello minimo, ordinati in ordine crescente di supporto. A questo punto viene generato un sottoalbero per ogni elemento del primo nodo appena generato. I sottoalberi così creati, però, sono semplicemente una lista di elementi che compaiono nella stessa riga dell'elemento radice del sottoalbero (all'interno dell'insieme di dati) per un numero di volte superiore alla percentuale di supporto minimo. In questo modo è stata creata una struttura molto semplice da analizzare per i miei scopi. Infatti l'albero CFP contiene le sole corrispondenze tra i video che superano la

percentuale di similitudine stabilita.

In fase di realizzazione dei test, anche questo secondo valore di supporto minimo percentuale è stato mantenuto volutamente basso. Infatti, si utilizza l'albero CFP per suggerire gli N video più simili a ciascun video contenuto nel database; per questo c'è la necessità di mantenere un certo numero di figli per ciascun nodo iniziale del database, in modo che i figli possano essere almeno N. Inoltre, per come è strutturato l'albero CFP, ogni elemento possiede come figli solo gli elementi che hanno un supporto assoluto maggiore del suo. In questo modo, non si potrebbe mai suggerire come video simile un filmato con un numero di descrittori minore di quello del filmato stesso, perchè questo non potrebbe mai essere contenuto nella propria lista dei figli. Questa limitazione è stata superata andando a valutare la similarità tra due video in modo più articolato. Si realizza una matrice che per ogni video di ingresso tenga memorizzati i video più simili; gli elementi all'interno della matrice vengono inseriti con la seguente tecnica. Dopo aver costruito l'albero CFP, a partire da ogni nodo del primo livello dell'albero, si ha la lista dei video che hanno una percentuale di frame comuni, con il video memorizzato nel nodo padre, maggiore della soglia di supporto minimo percentuale. Ciascuno di questi video viene inserito nella matrice alla riga che contiene i video simili del nodo padre ma per ogni relazione padre-figlio la corrispondenza viene inserita nella matrice anche alla riga che contiene i video simili al nodo figlio. Al termine di questo procedimento, dopo una scansione di tutto l'albero CFP, la matrice creata contiene le relazioni tra tutti i video del dataset di partenza; andando ad ordinare ogni riga della matrice in base al supporto, si ha la lista dei N video più simili ad ogni video di ingresso.

3.3.2 Software realizzato

Dopo la fase di acquisizione delle informazioni dal dataset, si è proceduto alla realizzazione di una struttura in grado di memorizzare al suo interno tutte le informazioni che sarebbero poi servite per la costruzione dell'albero CFP. Queste informazioni sono tutte le righe del DB di transazioni di partenza, dopo aver eliminato gli item non frequenti, cioè che non superano il supporto minimo, ed aver ordinato quelli frequenti in ordine ascendente di frequenza all'interno di ogni riga. Sono inoltre presenti le informazioni per tenere memorizzati tutti i DB condizionali. Entriamo nel merito delle strutture dati realizzate.

Strutture dati Nodo1 e Nodo2 Queste due strutture sono fondamentali per la successiva definizione delle strutture dati da utilizzare per la creazione dell'albero CFP. Infatti si è pensato di memorizzare i pattern all'interno di una lista per poterli accedere più velocemente grazie ai puntatori inseriti. Si è creata una struttura in cui ciascuna riga del DB di transazioni di partenza è rappresentata da un oggetto di tipo *Nodo2* che segnala l'inizio della riga del DB. Dal *Nodo2* parte poi la lista di tutti gli elementi della riga del DB, ciascuno dei quali è contenuto all'interno di un oggetto di tipo *Nodo1*. La struttura *Nodo1* è composta dai seguenti campi:

```
struct Nodo1 {
        int dato;
        Nodo1 * dx;
        Nodo1 * giu;
        int visitato; };
```

Questa struttura deve rappresentare un nodo all'interno della lista; quindi contiene il dato (che è un intero), due puntatori ad oggetti di tipo *Nodo1* ed una

variabile intera denominata con visitato. I puntatori agli oggetti di tipo *Nodo1* servono rispettivamente per puntare al successivo elemento della riga del DB (*dx) e per tenere traccia del prossimo elemento del DB condizionale con lo stesso inizio (*giu). Lo scopo di questo secondo puntatore verrà chiarito in seguito. Inoltre è presente il campo visitato che è un attributo che viene aggiornato al momento delle scansioni dei DB condizionali secondo un criterio che verrà illustrato successivamente.

Vediamo ora la struttura *Nodo2*, che serve per memorizzare il punto di inizio di ogni riga del DB di transazioni

```
struct Nodo2 {
        int datoIniziale;
        int numNodiDx;
        Nodo1 * dx;
        Nodo2 * giu; };
```

I campi all'interno di questa struttura sono un intero che rappresenta il dato iniziale, un altro per tenere traccia del numero di elementi di ciascuna riga e due puntatori. Il primo di questi puntatori (dx) serve per puntare alla lista di *Nodi1* che rappresenta l'effettivo contenuto di ciascuna riga di elementi, mentre il secondo, denominato con giu è un puntatore ad un elemento di tipo *Nodo2* che è l'inizio della riga successiva.

La lista contenente tutti gli item Si descrive ora la modalità di realizzazione della lista all'interno della quale vengono memorizzati tutti gli elementi del DB di transazioni dopo la seconda scansione, che sono gli item con supporto maggiore di quello minimo, in ordine ascendente di frequenza. In questa classe viene memorizzato un puntatore ad un elemento di tipo *Nodo2* che serve per tenere traccia del primo elemento della lista. Basta questo puntatore per me-

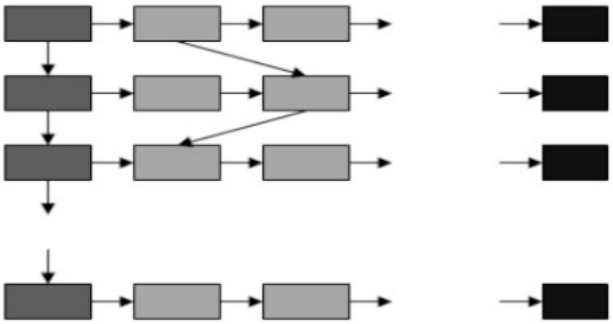

Figura 3.2: Modalità di rappresentazione degli elementi nella lista. I nodi iniziali sono quelli di tipo *Nodo2*, quelli centrali sono di tipo *Nodo1* e quelli finali sono zone di memoria vuote. Le frecce in figura rappresentano i puntatori. I puntatori tra gli elementi di tipo *Nodo2* sono quelli che servono per realizzare il DB condizionale, cioè collegano gli elementi iniziali di ciascuna riga del DB condizionale.

morizzare tutta la lista di elementi perchè da esso parte il puntatore alla riga successiva e il puntatore agli elementi della riga. In Figura 3.2 viene mostrata graficamente la memorizzazione degli elementi nella lista.

Un attributo che viene memorizzato è un vettore di interi (*vitem*) che serve per mantenere memorizzata la frequenza di ciascun item. Per esempio, se scriviamo *vitem*[i]=4, vogliamo indicare che l'*i*-esimo item ha una frequenza di 4 ripetizioni all'interno del DB di transazioni di partenza. Questo è un dato molto importante perchè durante il processo di creazione dell'albero CFP c'è sempre bisogno di conoscere le frequenze dei diversi item e quindi tramite questo attributo della classe si ha la possibilità di accedere rapidamente a queste informazioni. Per quanto concerne la creazione di questo vettore, c'è da dire che questa viene eseguita al momento della prima scansione del DB. All'interno di questa classe sono contenuti tutti i metodi che consentono di mantenere sempre aggiornata la lista di elementi, cioè i metodi per l'inserimento di una riga nella lista e per riordinare la riga, per riordinare la lista e per cancellare elementi dalla lista.

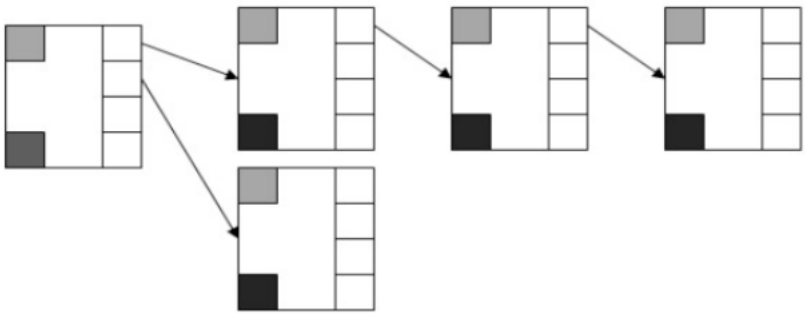

Figura 3.3: Rappresentazione di una struttura di appoggio per la costruzione dell'albero CFP. Ogni box presente in figura, rappresenta uno degli elementi della struttura di supporto. In tale struttura sono contenuti un dato che indica l'elemento contenuto all'interno del box (quadratino in alto a sinistra) e un puntatore ad un oggetto di tipo *Nodo2* che punta alla prima riga del DB di transazioni che contiene quel percorso specifico (quadratino in basso a destra). E' inoltre presente una lista dalla quale partono tutti i possibili cammini (parte da cui hanno inizio le frecce).

Altre classi di appoggio per la costruzione dell'albero CFP Per costruire l'albero CFP sono state utilizzate altre strutture di appoggio con le quali si è costruito un albero che consentisse di ricavare rapidamente tutte le informazioni per costruire i DB condizionali. A partire dal primo nodo che deve costituire l'albero CFP, si crea una struttura che per ciascuno degli item del primo nodo fa partire un sottoalbero contenente tutti i possibili cammini. Grazie alla Figura 3.3 viene esemplificata questa situazione.

Facendo riferimento a questa Figura, viene descritto il sistema di riempimento della struttura. Supponiamo di avere una parte del totale di righe di transazioni (che vengono rappresentati con valori numerici) che è la seguente:

c

c

c d

c d

Appoggiandoci alla Figura precedente, il primo box rappresenta la radice dell'intera struttura ed ha il puntatore al primo nodo vuoto, cioè settato a NULL. All'arrivo della prima riga del DB di transazioni, composta dal solo item *c*, si verifica l'esistenza di un puntatore ad un'istanza che rappresenti *c*. Poiché questo è il primo elemento che arriva, è chiaro che questa istanza non esista, per cui viene allocata (in figura questa situazione è rappresentata dalla freccia che va dal primo box al secondo) e al suo interno si setta la parte in alto a sx (rappresentante il dato contenuto) con il valore *c*, mentre il puntatore al primo Nodo2 contenente l'elemento (parte in basso a sx) assume l'indirizzo di memoria in cui è salvata questa riga appena inserita. All'arrivo della seconda riga del DB di transazioni, anch'essa composta solo da *c*, si verifica l'esistenza di un puntatore ad un'istanza che contenga *c*. Stavolta il puntatore esiste (lo abbiamo appena creato) e quindi accediamo a tale istanza. A questo punto si setta il puntatore (quadratino in basso a sx) all'indirizzo di memoria contenente questa riga appena arrivata creando una lista che al suo interno memorizzava già l'indirizzo inserito precedentemente che indicava l'altra riga contenente il solo elemento *c*. Quindi alla fine della procedura, si memorizzano tutti gli indirizzi delle righe contenenti il solo item *c*. Quando arriva la terza riga del DB di transazioni, rappresentata da *c d*, si parte sempre dalla radice, dalla quale si passa all'istanza contenente *c*. Stavolta, però, il procedimento non termina qui, ma si deve ripetere il tutto per l'item *d* a partire dall'istanza di *c*. Per cui viene creato un puntatore ad un'istanza contenente *d* il cui valore(parte in alto a sx) è settato a *d* mentre la parte in basso a sx (puntatore al primo elemento del DB di transazioni che sia *c d*) contiene proprio l'indirizzo di questa riga. Quindi si ripete il procedimento appena illustrato su tutte le righe del DB di transazioni. Grazie a questa struttura si ottengono diversi vantaggi. Viene occupata poca memoria, perché, come abbiamo appena descritto, la maggior parte del lavoro

è quella di settare correttamente dei puntatori. Infatti la struttura è composta essenzialmente da soli puntatori ed un solo elemento che indica il valore del dato. Il vantaggio principale si ottiene nella creazione dei DB condizionali, poiché, realizzando una visita anticipata di questo albero si ottengono i DB condizionali. Se si volesse sapere il DB condizionale di c, che quindi è quello che contiene tutte le righe che iniziano con c, basterà fare una visita della parte dell'albero che dalla radice va verso l'istanza di c. Nel momento in cui viene terminato il procedimento di risoluzione di questo DB, devono essere reinserite le righe nelle altre istanze del DB condizionale in base al prossimo item da analizzare. Per rappresentare questa struttura è stata creta una classe, nominata *NodoSpecial*. Tutti gli elementi di tipo *NodoSpecial* sono poi contenuti all'interno di un oggetto della classe *AlberoSpecial* in cui viene mantenuto l'albero CFP modificato la cui struttura è stata spiegata nella sottosezione 3.3.1. Finora si sono descritti tutti i procedimenti e le strutture dati per tenere traccia delle informazioni durante le scansioni del DB di transazioni e per creare e risolvere i DB condizionali. È stata infine creata una classe per mantenere realmente memorizzato l'albero e tutte le informazioni per arrivare alla costruzione. Inoltre da questa classe vengono richiamati tutti i metodi per svolgere i passi descritti finora.

Capitolo 4

Suggerimento di tag agli shot di un video

In questo capitolo viene descritto il sistema realizzato per la localizzazione temporale dei tag in un video e per il suggerimento degli stessi. L'idea di base su cui si appoggia il metodo, è la ricerca della similarità visuale tra frame del video e un set di immagini annotate ottenute a partire dai tag del video iniziale. A un video già etichettato presente su YouTube, viene applicato il sistema presentato nella tesi di Nencioni [Nencioni10]. Questo algoritmo elimina i tag poco rilevanti del video di partenza, quindi effettua un processo di espansione semantica dei tag più rilevanti basandosi sui framework *DbPedia* e *Wikipedia Miner*. Al termine della procedura, il video ha un insieme di tag più consistente rispetto a quello di partenza. Con questi tag vengono realizzate interrogazioni al sito *Flickr*, in modo da avere a disposizione un insieme di immagini identificative dei tag del video. Delle immagini *Flickr*, vengono mantenuti anche i tag, che sono utili per il suggerimento degli stessi a livello di segmento del video.

Il sistema realizzato in questa tesi, quindi, riceve in ingresso un video e un insieme di immagini annotate che dovrebbero essere rappresentrative dei concetti ritrovabili all'interno del video stesso. Il video di ingresso viene suddiviso in sotto-segmenti con un algoritmo semplice e veloce che analizza la luminosità dei frame per carpire variazioni significative tra un frame e il successivo. Quindi, viene estratto un frame per ognuno dei segmenti in cui è stato suddiviso il video. Per ognuno di questi frame viene calcolato il descrittore che è un vettore di 370 dimensioni. Di queste dimensioni se ne calcolano 80 per ognuno dei tre vocabolari visuali basati sui punti SIFT a 128 dimensioni. Si applica una procedura, che può essere rinominata come **TOP-SIFT** mirata alla valutazione dei punti SIFT più presenti all'interno di un'immagine. Visto che si utilizzano i migliori 80 punti per ciascuno dei tre vocabolari visuali creati, si hanno 240 dimensioni che tengono traccia della distribuzione dei punti SIFT nell'immagine. Altre 80 dimensioni sono date dal calcolo dell'**Edge Histogram Descriptor**, descrittore di tessitura dello standard MPEG-7, mentre le rimanenti 50 dimensioni tengono conto della distribuzione del colore, calcolate grazie all'utilizzo di un approccio basato sul **correlogramma di colore**.

Dopo aver calcolato il descrittore di un frame identificativo di uno shot del video, viene applicata una procedura per il confronto dello stesso con i descrittori, precedentemente computati, di tutte le immagini annotate scaricate da *Flickr*. Viene utilizzato un algoritmo basato sull'approccio $Vote^{+}$ descritto in [vanZwol08] per promuovere alcuni tag da associare allo shot. Nel resto del capitolo vengono descritti l'algoritmo di segmentazione del video, le modalità di calcolo del descrittore e il sistema per il confronto del frame con le immagini campione e il relativo meccanismo di suggerimento dei tag.

Figura 4.1: Ogni frame viene suddiviso in 2×2 sotto-blocchi.

4.1 Algoritmo di segmentazione del video

L'algoritmo utilizzato per la suddivisione del video in sotto-segmenti si ispira a quello descritto in [Deng08]. Ogni frame del video viene suddiviso in un certo numero di sotto-blocchi, quindi viene calcolato il valore medio di luminosità per ogni sotto-blocco e tali valori vengono ordinati. L'ordinamento appena creato viene utilizzato come feature per l'identificazione di sotto-segmenti; infatti se frame adiacenti tra loro presentano lo stesso ordinamento, allora essi appartengono allo stesso sotto-segmento. Una scelta importante risiede nel numero di sotto-blocchi in cui suddividere i frame; con un numero maggiore di sotto-blocchi è garantita una rappresentazione più dettagliata del flusso informativo presente nel video. In un'applicazione del genere non è richiesta una precisione estrema, per cui è preferibile ridurre il numero di sotto-segmenti in cui viene suddiviso il video di partenza. Per queste ragioni, nell'applicazione creata, ogni sotto-segmento è stato suddiviso in 4 sotto-blocchi, come mostrato in Figura 4.1.

Gli esperimenti realizzati hanno evidenziato che l'ordinamento dei 2×2 sotto-blocchi è sufficiente per rintracciare i sotto-segmenti e per distinguere un sotto-segmento da un altro.

La Figura 4.2 illustra un esempio dell'algoritmo di rilevazione dei sotto-segmenti.

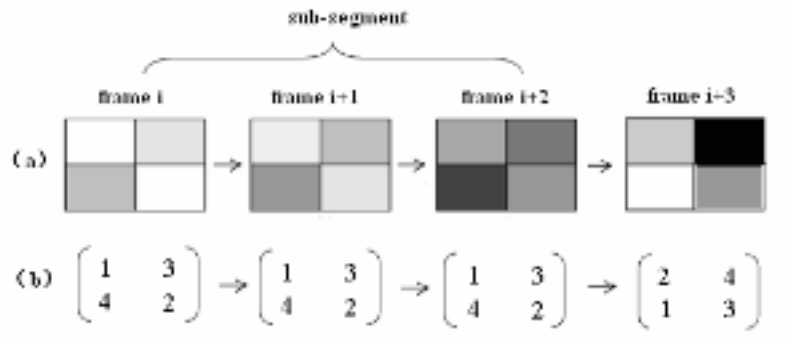

Figura 4.2: Un esempio dell'algoritmo di rilevazione dei sotto-segmenti

Assumendo che l'ordine della luminosità media dei quattro sotto-blocchi valga $(1, 3, 4, 2)$ sia per il frame i-esimo sia per l'$(i + 1)$-esimo e per l'$(i + 2)$-esimo, allora questi tre frame appartengono al medesimo sotto-segmento. Se, invece, l'ordinamento dei valori di luminosità dei quattro sotto-blocchi del frame $(i+3)$-esimo produce il vettore $(2, 4, 1, 3)$, che è diverso dal vettore del frame $(i + 2)$-esimo, allora quest'ultimo frame è assegnato ad un segmento diverso rispetto a quello che contiene i tre frame precedenti.

Attraverso questo algoritmo, i frame adiacenti che presentano lo stesso vettore di ordinamento della luminosità vengono assegnati al medesimo sotto-segmento. Per di più, la lunghezza di un sotto-segmento viene decisa dal numero di frame consecutivi con lo stesso ordinamento. Nel lavoro svolto, non sono stati considerati i sotto-segmenti che non fossero composti da almeno 3 frame, utizzando un campionamento di 4 frame al secondo. In questo modo non sono stati effettuati calcoli su quei sotto-segmenti troppo corti che spesso sono dati da cambi di scena improvvisi e quindi non includono contenuti particolarmente interessanti dal punto di vista di suggerimento ed identificazione dei tag. Nel sistema creato, è stato utilizzato un solo frame per rappresentare ogni sotto-segmento, mentre i tag suggeriti per quel frame vengono assegnati a tutto il sotto-segmento in cui è contenuto il frame stesso.

4.2 Calcolo del descrittore di un frame

Una volta selezionato un frame per ogni sotto-segmento del video, è stata realizzata la procedura per il calcolo del descrittore. Lo stesso procedimento viene applicato anche a tutte le immagini campione scaricate da *Flickr*, che servono per il confronto con i frame del video. Come già scritto in precedenza, il descrittore è formato da 3 tipi di informazione distinti: TOP-SIFT, Edge Histogram Descriptor e un correlogramma di colore.

4.2.1 TOP-SIFT

Per analizzare la distribuzione dei punti SIFT in un'immagine è stata realizzata una foresta di vocabolari visuali, seguendo l'approccio Bag-of-Visual-Words. A differenza di quanto fatto nel sistema per l'identificazione di video simili, in questo caso sono stati costruiti dei vocabolari visuali ad un solo livello.

Creazione dei vocabolari visuali Come nel sistema per il ritrovamento di video simili, è stata utilizzata la tecnica Bag-of-Visual-Words (BoW) nell'ambito dei contenuti visuali. Da un insieme di video o di immagini, si possono estrarre dei keypoint locali, che in questo caso sono i SIFT. Dopo aver raccolto tutti i keypoint, viene applicato un algoritmo di clustering in modo che i keypoint vengano raggruppati in un certo numero di cluster. Ogni cluster viene trattato come una parola visuale che va a far parte del vocabolario di parole visuali. Un singolo vocabolario è originato da un processo di quantizzazione, ottenuto applicando il *k-means clustering* sull'insieme di punti SIFT estratti da un database di video utilizzato appositamente per raccogliere punti. Viene applicato l'algoritmo k-means sull'intero insieme di dati di training, definendo k cluster, rappresentati da altrettante parole visuali. A questo punto, i dati di

training vengono partizionati in k gruppi, dove ciascun gruppo contiene i punti SIFT la cui parola visuale più simile tra le k create, è quella identificativa del gruppo stesso. Nel sistema creato, è stato utilizzato un valore di $k = 666$. Sono stati utilizzati tre vocabolari distinti, costruiti con la tecnica appena descritta, in modo da formare un insieme di vocabolari, costruiti con tre insiemi differenziati di video. Il motivo che ha spinto a realizzare tre vocabolari è che in questo modo si poteva variegare maggiormente l'informazione, altrimenti contenuta in un solo vocabolario. Nel formare un singolo vocabolario, sono stati scaricati video da *YouTube* cercando di coprire tutte le categorie in cui *YouTube* suddivide i suoi video.

Per realizzare questa fase di creazione dei vocabolari visuali, sono stati implementati due progetti in linguaggio C++. Il primo di questi progetti riceve in ingresso un insieme di video e per ciascuno di essi effettua una suddivisione in segmenti con la tecnica descritta in 4.1. Da ogni segmento viene selezionato un frame da cui sono estratti i punti SIFT a 128 dimensioni, che vengono quindi memorizzati in un file. Questo progetto è stato eseguito tre volte, una per ciascun vocabolario visuale da creare.

Il secondo progetto realizzato riceve in ingresso l'intera raccolta di punti SIFT ed applica una suddivisione in cluster con l'applicazione di un algoritmo $k-means$, tramite la funzione di *opencv* riportata di seguito:

```
double kmeans(const Mat& samples, int clusterCount, Mat& labels, TermCriteria
termcrit, int attempts, int flags, Mat* centers)
```

Il funzionamento della funzione e il significato dei parametri sono chiariti nella sottosezione 3.1.1. Il numero di cluster è stato scelto pari a 666 per aver a disposizione un numero elevato di cluster da cui selezionare quelli più presenti in un'immagine, grazie all'applicazione del metodo TOP-SIFT creato.

Calcolo del descrittore I cluster finali creati con il processo precedente vengono identificati come il *dizionario di parole visuali*. In modo del tutto simile al fatto che un documento consiste di parole testuali, si può immaginare che un'immagine sia formata da parole visuali. Dato che un documento può essere descritto da un insieme di parole particolarmente identificative, in un'immagine alcune parole visuali possono apparire più frequentemente rispetto alle altre, quindi possono essere maggiormente descrittive del contenuto dell'immagine stessa. Questa è l'idea su cui è fondato il metodo creato, che è stato chiamato TOP-SIFT.

Al frame identificativo di una sotto-sequenza vengono cambiate le dimensioni, portandole a 240×160. Successivamente vengono estratti i punti SIFT dall'immagine. Per ogni punto SIFT rilevato viene cercata la parola visuale più simile, in termini di *distanza Euclidea*, tra tutte le 666 che compongono un vocabolario visuale. In questo modo l'insieme di punti rilevato viene convertito in un istogramma di frequenza, identificativo della frequenza con cui compaiono le varie parole visuali all'interno dell'immagine. Per ogni frame vengono creati tre istogrammi di frequenza, uno per ciascun vocabolario visuale presente. Per formare una parte del descrittore dell'immagine vengono selezionate le parole visuali con la frequenza più alta all'interno dell'istogramma di frequenza. Dato che vengono utilizzate solo le migliori N parole visuali, il descrittore viene chiamato TOP-SIFT. Il valore di N è stato fissato a 80, per ognuno dei tre istogrammi. In questo modo si sono ottenute 240 dimensioni delle 370 complessive che formano il descrittore. Oltre al valore di frequenza delle parole visuali più presenti, devono essere memorizzati anche gli indici delle stesse, per poter poi valutare la similarità tra immagini, tramite la *cosine similarity*.

Per l'estrazione dei punti SIFT in un'immagine viene utilizzato l'algoritmo di Lowe [Lowe04]; la funzione utilizzata è la seguente:

```
extern int sift_features( IplImage* img, struct feature** feat );
```

La funzione rileva i punti SIFT in un'immagine utilizzando i parametri di default. Tutte le feature rilevate vengono poi memorizzate in un vettore puntato da feat. Il parametro img indica l'immagine nella quale rilevare le feature, mentre feat è un puntatore a un vettore nel quale memorizzare le feature rilevate. La funzione ritorna il numero di feature rilevate, oppure il valore -1 in caso di fallimento.

La struttura feature consente di rappresentare una feature affine ed invariante presente in un'immagine tramite la memorizzazione delle coordinate, della scala a cui è stata calcolata la feature, dell'orientazione e di ulteriori caratteristiche che sono utili per dettagliare la feature rilevata.

4.2.2 Edge Histogram Descriptor

Una parte del descrittore è composta dalle 80 dimensioni dell'Edge Histogram Descriptor, che è stato descritto in A.4. Per il calcolo effettivo di questo descrittore sono state utilizzate una serie di funzioni C++ già pronte, adattate al programma realizzato.

4.2.3 Calcolo del correlogramma di colore

Come già spiegato in A.6, il correlogramma di colore è una feature descrittiva delle relazioni spaziali tra colori in un'immagine. Una definizione più precisa è la seguente: il correlogramma di colore di un'immagine è una tabella indicizzata da una coppia di colori, dove il k-esimo ingresso di *(i, j)* specifica la probabilità di trovare un pixel di colore j a distanza k da un pixel di colore i, all'interno dell'immagine.

Il correlogramma è stato realizzato basandosi sulla componente di colore H della rappresentazione HSV, per cui la prima operazione da realizzare è la conversione dell'immagine dalla rappresentazione RGB a quella HSV. Per il calcolo del correlogramma, l'immagine è stata suddivisa in celle; più precisamente, lavorando solo con immagini 240×160, l'immagine è stata suddivisa in $GRIGLIAASSEX = 12 \times GRIGLIAASSEY = 8$ celle, in modo che ciascuna cella contenga 400 pixel. La struttura che rappresenta il correllogramma è stata creata come una matrice, in cui una dimensione è data dal numero di possibili accoppiamenti di colore, mentre l'altra dimensione è data dai possibili valori di distanza assumibili tra le coppie di celle presenti nell'immagine; per questo motivo le dimensioni della matrice che rappresenta il correlogramma sono state impostate a 81×18. La prima dimensione è data dal numero di coppie di colori riscontrabili. La componente H può assumere valori compresi tra 0 e 8, grazie all'applicazione del procedimento descritto di seguito. I 360^o gradi del dominio di H sono divisi in sei fasce di colore, come si può osservare nella parte sinistra di Figura A.8. La fascia del colore rosso sta a cavallo degli 0 gradi, cioè va da 0^o a 30^0 e da 330^o a 360^o. Per fare in modo che questa fascia sia continua, al valore di H che si ottiene viene sommato 30 in modulo 360, cioè:

$$H = (H + 30) \text{mod} 360$$

A questo punto si esegue la seguente operazione:

$$(H/360.001) \times 6$$

Con la divisione si porta il valore di H nell'intervallo [0,1]. Moltiplicando il valore ottenuto per 6 si ottiene un valore compreso tra 0 e 5 che corrisponde ad una delle 6 fasce di colore. Per distinguere anche i casi di bianco, grigio e nero

si effettuano dei controlli sui valori R, G e B e si assegna il valore 6 nel caso di bianco, 7 per il nero e 8 se il pixel ha un colore corrispondente ad una tonalità di grigio. Utilizzando questa procedura, il valore di colore di una cella è quello più frequente tra tutti i pixel della cella. Dato che i valori possibili di H sono 9, i possibili accoppiamenti di colore tra celle sono 9×9, da cui risultano gli 81 elementi della prima dimensione della matrice. In questo modo la matrice è formata da 81 righe, una per ciascun accoppiamento di color medio delle celle possibile. La seconda dimensione, invece, è data dalle possibili distanze tra coppie di celle. La distanza tra celle si ottiene con la seguente operazione:

$$dist_{12} = abs(valCellaX_1 - valCellaX_2) + abs(valCellaY_1 - valCellaY_2) \quad (4.1)$$

dove $valCellaX_1$e $valCellaX_2$ rappresentano le coordinate lungo l'asse delle ascisse delle celle tra cui viene calcolata la distanza. Questi valori possono variare tra 0 e 11, visto che la larghezza dell'immagine viene suddivisa in 12 parti. $valCellaY_1$ e $valCellaY_2$ sono l'analogo per l'asse delle ordinate; questi valori possono variare tra 0 e 7, visto che l'altezza dell'immagine viene suddivisa in 8 parti. Proprio per questo motivo si ha che la seconda dimensione della matrice che rappresenta il correlogramma vale 18, visto che al massimo due celle possono differire di 11+7 unità. Il procedimento prevede che un singolo elemento della matrice venga aggiornato secondo la modalità dell'Algoritmo 4.1.

Riferendosi a tale Algoritmo, GRIGLIAHX e GRIGLIAHY sono, rispettivamente, i numeri di parti in cui vengono suddivisi la larghezza e l'altezza dell'immagine. corr è la matrice che rappresenta il correlogramma, mentre H_i e H_j sono le componenti di colore H più presenti, rispettivamente, nella cella i-esima e nella cella j-esima. $dist_{ij}$ è il valore di distanza calcolato come nella Formula 4.1.

Algorithm 4.1 Modalità di aggiornamento degli elemetni della matrice che corrisponde al correlogramma

1. for(i=0;i<GRIGLIAHX×GRIGLIAHY;i++)
2. for(j=0;j<GRIGLIAHX×GRIGLIAHY;j++)
3. if(i>j)
4. then $\mathsf{corr}[\mathsf{H_i} \times 9 + \mathsf{H_j}][\mathsf{dist_{ij}}] + +$
5. end if
6. end for
7. end for

Algorithm 4.2 Calcolo del correlogramma utilizzando un'ulteriore suddivisione delle singole celle

1. for(i=0;i<GRIGLIAHX×GRIGLIAHY;i++)
2. for(j=0;j<GRIGLIAHX×GRIGLIAHY;j++)
3. if(i>j)
4. for(k=0;k<4×4;k++)
5. $\mathsf{corr}[\mathsf{H_{ik}} \times 9 + \mathsf{H_{jk}}][\mathsf{dist_{ij}}] + +$
6. end for
7. end if
8. end for
9. end for

Per avere un'informazione ancor più completa, ognuna delle 12×8 celle di 400 pixel in cui è stata inizialmente suddivisa l'immagine viene ulteriormente suddivisa in 4×4 aree, quindi ciascuna composta da 25 pixel. Quindi la componente H dominante viene calcolata su queste aree di 25 pixel e quando si confrontano due celle di primo livello, vengono effettuati 4×4 confronti, cioè uno per ogni area di 25 pixel. In questo modo il procedimento realizzato è quello presentato in Algoritmo 4.2.

Come si può osservare, il valore di $\mathsf{dist_{ij}}$ rimane analogo al caso precedente, cioè la distanza viene calcolata rispetto agli indici delle 12×8 celle di 400 pixel in cui è stata suddivisa inizialmente l'immagine. L'indirizzamento sulla prima dimensione della matrice, invece, avviene sulla base delle componenti H dominanti delle $k-$esime aree di 25 pixel all'interno dell'$i-$esima e della $j-$esima cella di 400 pixel, indicate rispettivamente con $\mathsf{H_{ik}}$ e H_{jk}.

In definitiva, tramite una matrice 81×18 viene memorizzato l'intero correlogramma di colore di un'immagine. Di questo correlogramma vengono selezionate solo le 50 componenti maggiori, che rappresentano le relazioni tra colori più presenti all'interno dell'immagine, quindi maggiormente rappresentative della distribuzione del colore e delle relazioni spaziali tra colori nell'immagine. Queste componenti sono le 50 dimensioni che vanno a completare il descrittore di immagini realizzato.

4.3 Confronto tra immagini

Per ogni segmento in cui viene suddiviso il video, è estratto un frame da confrontare con tutte le immagini campione ottenute con l'applicazione dell'algoritmo di Nencioni al video di partenza. Il confronto tra il frame e le immagini avviene sulla base del descrittore di 370 dimensioni illustrato nella sezione 4.2. Le dimensioni del descrittore sono date da tre diverse caratteristiche delle immagini, i punti SIFT, l'Edge Histogram Descriptor e il correlogramma di colore. Per questo motivo, la similarità tra immagini e quindi tra i loro descrittori, è stata valutata in modo distinto per le tre sorgenti diverse da cui provengono i dati che compongono l'intero descrittore. Alla fine si ottiene un valore di similarità complessivo tra il frame e una singola immagine campione, che consente di capire quali siano le immagini campione maggiormente simili al frame in questione.

La similarità tra le componenti del descrittore che rappresentano il TOP-SIFT viene valutata tramite la *cosine similarity*. Questa è valutata su tre gruppi di 80 elementi del descrittore, visto che i vocabolari visuali utilizzati sono tre. Introduciamo la formula della *cosine similarity* e poi vediamo come è stata utilizzata nel caso in esame:

$$sim_{ij} = \frac{\sum_{k=1}^{m} f_k(I_1) \times f_k(I_j)}{\sqrt{\sum_{k=1}^{m} f_k(I_1)^2 \sum_{k=1}^{m} f_k(I_j)^2}} \tag{4.2}$$

dove m è il numero di parole visuali che compongono il vocabolario, mentre $f_k(I_i)$ è il peso del k-esimo bin nel keyframe I_1.

Una volta selezionate le 80 dimensioni corripondenti allo stesso vocabolario per entrambi i descrittori da confrontare, si valuta la *cosine similarity*. Il denominatore viene calcolato come nella Formula 4.2, con il valore di $m = 80$, cioè vengono sommati i valori di $f_k(I_i)$ per le 80 parole visuali più presenti nei vocabolari visuali delle due immagini che vengono confrontate. In questo caso $f_k(I_i)$ rappresenta la frequenza percentuale del bin k-esimo all'interno dell'immagine i-esima. Per quanto riguarda il numeratore, il prodotto $f_k(I_1) \times f_k(I_j)$ viene effettuato solo se il k-esimo bin è presente tra gli 80 più numerosi di entrambe le immagini di cui si sta valutando la similarità. All'atto pratico, si prende il frame da confrontare con tutte le immagini campione, e si estraggono le 80 componenti provenienti da un unico vocabolario visuale. A questo punto, per ogni immagine campione, vengono selezionate le 80 componenti del descrittore corrispondenti allo stessto vocabolario visuale. Per valutare la *cosine similarity* tra il frame e la singola immagine campione, si considerano le frequenze percentuali delle 80 componenti del descrittore del frame che vengono moltiplicate per zero se la componente corrispondente non è presente anche tra le 80 del descrittore dell'immagine campione, altrimenti sono moltiplicate per il valore della frequenza percentuale nell'immagine campione. Con questa operazione viene calcolato il numeratore di Formula 4.2, mentre per il denominatore vengono eseguiti i prodotti delle 80 componenti di entrambe le immagini, quella rappresentativa dello shot e quella campione. Questo processo viene ripetuto per i tre distinti vocabolari visuali, in modo da ottenere tre valori di *cosine similarity* che legano l'immagine rappresentativa dello shot con ogni immagine

Algorithm 4.3 Calcolo della similarità tra correlogrammi

```
1. for(j=0;j<numeroImmagini;j++)
2. for(ind1=0;ind1<TOPN;ind1++)
3. for(ind2=0;ind2<TOPN;ind2++)
4. if(correlogramma[ind1]==MatriceCorrelogramma[j][ind2])
5. then similarita[j]+=((TOPN-ind1)×(TOPN-ind2)
6. end if
7. end for
8. end for
9. end for
```

campione. Utilizzando questa modalità di rappresentazione, il valore di *cosine similarity* calcolato cresce all'aumentare della similarità tra i TOP-SIFT delle due immagini confrontate.

La similarità tra le componenti del descrittore derivanti dall'*Edge Histogram Descriptor* di due immagini diverse viene calcolata tramite la *distanza Euclidea*. La parte di descrittore data dall'*Edge Histogram* è composta da 80 dimensioni, ciascuna è un numero intero che può assumere valori nell'intervallo $[0, 7]$. In questo modo, più due immagini sono simili, secondo i valori assunti dall'*Edge Histogram Descriptor*, e minore sarà il valore di *distanza Euclidea* che si ottiene.

Per quanto riguarda la similarità tra i correlogrammi di due immagini distinte, è stata progettata una misura che tenga conto della presenze comune di elementi in entrambi i correllogrammi. Infatti, il correlogramma di un'immagine viene rappresentato con le sole 50 dimensioni che corrispondono agli elementi con la maggiore frequenza, cioè alle relazioni spaziali di colore più presenti nell'immagine. Dato il correllogramma del frame che rappresenta uno shot, viene realizzato un confronto con i correlogrammi di tutte le immagini campione andando a valutare l'occorrenza di elementi comuni e pesandola a seconda della posizione degli stessi nelle rispettive graduatorie dei migliori 50. In questo modo si enfatizza la presenza di elementi comuni che occupino posizioni elevate di classifica. In Algoritmo 4.3 viene riportato lo pseudo-codice di questa valutazione:

Con questa procedura viene calcolata la similarità del frame con tutte le immagini campione che in totale sono numeroImmagini. Questi valori vengono poi memorizzati nel vettore similarita. TOPN è il numero di componenti utilizzate per rappresentare il correlogramma, cioè 50; il vettore correlogramma rappresenta il correlogramma del frame del video, cioè le 50 componenti più rilevanti dell'intero correlogramma calcolato sull'immagine. MatriceCorrelogramma è invece la matrice che memorizza i correlogrammi di tutte le immagini campione. Tramite questa procedura, più due correlogrammi sono simili e maggiore è il valore di similarità tra essi.

Dopo aver realizzato tutte le operazioni suddette, si hanno cinque valori di similarità, di diversa natura, che legano un frame del video con ogni immagine campione. Questi cinque valori devono essere uniti per ottenere un unico indice rappresentativo della similarità tra le immagini. Per poter fare ciò, ogni valore deve essere confrontato con un riferimento fisso, dal quale viene calcolata una distanza percentuale. La somma delle distanze percentuali su tutti e cinque i valori di similarità consente di ottenere un dato unico che esprima la reale corrispondenza tra le immagini.

Dopo una fase di test, sono stati stabiliti i valori di riferimento per i tre tipi di descrittori utilizzati. Con l'Algoritmo 4.4 viene presentata la procedura utilizzata per ottenere un indice di similarità unico tra il frame del video e le singole immagini campione.

La procedura consente di calcolare un valore unico di similarità, sommaDistanze che è un vettore di numeroElementi posizioni per memorizzare l'indice globale di similarità tra il frame e tutte le immagini campione. Si vogliono limitare i valori di similarità che oltrepassano (o sono inferiori, a seconda della misura di similarità utilizzata) una soglia prefissata che serve come riferimento per il calcolo di una distanza percentuale. Se la similarità supera la soglia, essa viene

Algorithm 4.4 Algoritmo per il calcolo della similarità tra il frame e le immagini campione

```
1. for(j=0;j<numeroImmagini;j++)
2. if(cosineSimilarityVoc1[j]>sogliaSIFT)
3. then cosineSimilarityVoc1[j]=sogliaSIFT
4. end if
5. if(cosineSimilarityVoc2[j]>sogliaSIFT)
6. then cosineSimilarityVoc2[j]=sogliaSIFT
7. end if
8. if(cosineSimilarityVoc3[j]>sogliaSIFT)
9. then cosineSimilarityVoc3[j]=sogliaSIFT
10. end if
11. if(euclDistEH[j]<sogliaEH)
12. then euclDistEH[j]=sogliaEH
13. end if
14. if(similCorrelogramma[j]>sogliaCorrel)
15. then similCorrelogramma[j]=sogliaCorrel
16. end if
17. sommaDistanze[j]+=(sogliaSIFT-cosineSimilarityVoc1[j])/(sogliaSIFT×2)
18. sommaDistanze[j]+=(sogliaSIFT-cosineSimilarityVoc2[j])/(sogliaSIFT×2)
19. sommaDistanze[j]+=(sogliaSIFT-cosineSimilarityVoc3[j])/(sogliaSIFT×2)
20. sommaDistanze[j]+=(euclDistEH[j]-sogliaEH)/euclDistEH[j]
21. sommaDistanze[j]+=(sogliaCorrel-similCorrelogramma[j])/sogliaCorrel
22. end for
```

fissato al valore della soglia stessa, in modo che quando viene calcolata la distanza percentuale questa risulti pari allo 0%. Questo indica che le due immagini presentano una relazione di similitudine forte per quel particolare tipo di descrittore. cosineSimilarityVoc1[j], cosineSimilarityVoc2[j] e cosineSimilarityVoc3[j] sono i valori di *cosine similarity* tra il TOP-SIFT del frame e i TOP-SIFT della j-esima immagine campione calcolati, rispettivamente, sul primo, sul secondo e sul terzo vocabolario visuale. Questi dati vengono limitati superiormente con il valore sogliaSIFT per poi calcolare la distanza percentuale rispetto al valore sogliaSIFT, che è lo stesso per tutti e tre i vocabolari. Il valore di soglia è stato scelto in modo che nella maggior parte dei casi la *cosine similarity* risultasse inferiore a questa soglia. Quando la *cosine similarity* supera il valore di sogliaSIFT, vuol dire che le due immagini presentano una forte similarità rispetto al vocabolario visuale in questione. Le stesse considerazioni sono state fatte anche durante il processo di decisione dei valori di soglia per l'Edge Histogram e per il correlogramma. Come si può notare nella riga 11 dell'Algoritmo 4.4, il valore della similarità tra gli Edge Histogram Descriptor, euclDistEH, viene limitato inferiormente, visto che è stato calcolato tramite una *distanza Euclidea* e quindi il valore è tanto più basso quanto più le immagini sono simili.

Dopo aver limitato i singoli valori di similarità, si procede al calcolo di sommaDistanze[j] che rappresenta il valore finale di distanza tra il frame del video e la j-esima immagine campione. Questo calcolo avviene sommando tutte le distanze percentuali rispetto al valore di soglia prefissato per i singoli tipi di descrittore utilizzati. Questo valore percentuale viene diviso per due quando si tratta la distribuzione dei punti SIFT, visto che questo tipo di descrittore porta il suo contributo tre volte, una per ciascun vocabolario visuale, e senza questa operazione avrebbe influito troppo sul valore finale della similarità. Questa operazione viene svolta moltiplicando per 2 il denominatore delle equazioni delle

righe 17-19 dell'Algoritmo 4.4. Dopo aver eseguito queste operazioni, il vettore *sommaDistanze* contiene le distanze percentuali tra il frame del video e tutte le immagini campione. Ordinando questo vettore in ordine crescente, ai primi posti del vettore avremo le immagini più simili al frame del video, accompagnate da una misura di distanza che risulta utile per stabilire se le immagini siano abbastanza simili al frame in questione.

Il valore di similarità appena calcolato può essere ancora modificato nel caso in cui si verifichino delle condizioni particolari. Infatti, il valore di distanza appena computato può essere abbassato o innalzato in alcuni casi particolari. Se nel confronto tra il frame e la j-esima immagine campione, i valori di *cosine similarity* ottenuti su tutti e tre i vocabolari visuali superano la soglia sogliaSIFT, allora il valore di *distanzaTotale[j]* viene decrementato di una determinata quantità. Il motivo è che se c'è accordo nella valutazione ottenuta su tutti e tre i vocabolari visuali, è probabile che le immagini in questione siano realmente simili. Allo stesso modo, la misura di distanza complessiva viene decrementata di quantità diverse a seconda di quanti e quali siano gli indici di similarità che superano la soglia stabilita propria dell'indice. C'è anche la possibilità di aumentare il valore di distanza complessiva se uno o più indici di similarità sono troppo distanti dal valore di riferimento per quell'indice. Queste operazioni vengono applicate per sfruttare al massimo il fatto di avere descrittori di diversa natura; infatti se una coppia di immagini risulta molto simile per la maggior parte dei tipi di descrittore, e magari non lo è solo per uno di questi, è giusto premiare questa situazione abbassando il valore di distanza. Viceversa, se una coppia di immagini risulta molto poco correlata per più tipi di descrittore, allora è utile aumentarne il valore di distanza, per far pesare maggiormente questa situazione.

4.4 Modalità di suggerimento dei tag

4.4.1 L'approccio realizzato

Per la modalità di suggerimento dei tag, il sistema creato si basa su quanto realizzato dagli autori dell'articolo [Li09] a cui è stata dedicata la Sezione 2.10 di questo documento. A partire da un'immagine, viene proposto un algoritmo di votazione che apprende la rilevanza dei tag accumulando voti dai vicini visuali dell'immagine di partenza. L'idea di base di questo metodo suggerisce che, data un'immagine, la rilevanza di ognuno dei suoi tag può essere inferita verificando il numero di volte che il tag in questione compare all'interno del set di immagini simili. In base a questo, maggiore è la frequenza di un tag all'interno del set, maggiore sarà la sua rilevanza. Non sempre i tag che hanno una frequenza molto elevata sono buoni; il rischio è quello di attribuire una rilevanza alta a quei tag che in realtà sono troppo generali. Una buona misura della rilevanza deve perciò considerare sia la distribuzione all'interno del set di immagini simili sia all'interno dell'intera collezione.

Queste affermazioni sono tutte molto interessanti per il nostro sistema, con l'eccezione che, nel mio algoritmo, i frame estratti dal video siano considerati come non annotati. Infatti, si è preferito non assegnare ai frame tutti i tag del video di partenza per poi validarli, visto che questi spesso non sono affidabili e sono fonte di rumore. Inoltre, al video di partenza viene applicato l'algoritmo di Nencioni [Nencioni10] che si occupa del raffinamento dei tag e della loro espansione semantica. Per questi motivi viene realizzato un processo per assegnare tag al frame del video in base alla similarità visuale con le altre immagini, considerando che il frame di partenza non sia già taggato.

Con il procedimento spiegato nella sezione precedente, si ottiene una misura di distanza tra il frame del video e tutte le immagini campione. Ordinando le

immagini in ordine crescente in base a questo valore di distanza, si hanno le k immagini più simili al frame in questione, ciascuna con un proprio valore di distanza.

La formulazione originale dell'algoritmo di [Li09], prevede che la rilevanza del tag w rispetto all'immagine I ed al set dei suoi k neighbors si definisca come:

$$tagRelevance\,(w, I, k) := n_w\,[N_f\,(I, k)] - Prior\,(w, k) \tag{4.3}$$

L'espressione 4.3 esprime come la rilevanza del tag w dipenda dal numero delle sue occorrenze nei k *nearest neighbors* dell'immagine I e dalla frequenza a priori del tag stesso. Il termine $n_w\,[N_f\,(I, k)]$ indica il numero di voti dei "vicini" visuali sul tag w. Mentre il termine $Prior\,(w, k)$ è la frequenza a priori del tag w nell'intera collezione e viene calcolata come:

$$Prior\,(w, k) \approx k\frac{\mid L_w \mid}{\mid \Phi \mid} \tag{4.4}$$

Questo metodo è stato applicato modificandone alcune caratteristiche. La differenza sostanziale sta nel fatto che i frame di partenza non sono già taggati, per cui non si deve efffettuare una loro validazione, bensì si devono assegnare tag al video in base a quelli presenti nelle immagini campione più simili al frame. Un'altra differenza importante è che viene effettuata una pesatura distinta dei tag delle immagini più simili, a seconda della misura di distanza tra il frame e l'immagine campione taggata. Con questa operazione si vogliono enfatizzare i tag delle immagini campione molto simili al frame, mentre si vuole dare un peso minore a quelle non molto simili. Infatti, andando a selezionare le k immagini campione più simili in tutti i casi, potrebbe succedere che un frame del video sia fuori dal contesto a cui appartengono i tag del video stesso. Per questo, non

Algorithm 4.5 Assegnazione dei pesi alle immagini campione

```
1. for(i=0;i<k;i++)
2. if(distanzaKNN[i]<=soglia1)
3. then fattoreMoltiplicativo[i]=4
4. end if
5. if((distanzaKNN[i]>soglia1)&&(distanzaKNN[i])<=soglia2)
6. then fattoreMoltiplicativo[i]=3
7. end if
8. if(distanzaKNN[i]>soglia3)
9. fattoreMoltiplicativo[i]=0
10. end if
11. if((distanzaKNN[i]>soglia2)&&(distanzaKNN[i])<=soglia3)
12. then fattoreMoltiplicativo[i]=(soglia3-distanzaKNN[i])*2
13. end if
14. end for
```

sarà molto simile a nessuna delle immagini campione e anche scegliendo le k più vicine dell'intero set è preferibile che i tag di quest'ultime vengano pesati meno. L'algoritmo creato assegna i pesi alle immagini in base al valore di distanza dal frame del video; i pesi assegnati all'immagini vengono estesi ai tag della stessa. Per chiarire questo aspetto, si analizzi l'Algoritmo 4.5.

In questo Algoritmo si effettua un ciclo che scorre le k immagini campione più simili. Per ognuna di esse viene valutato il livello di similarità con il frame, grazie al vettore distanzaKNN, dove distanzaKNN[i] è la distanza visuale dell'i-esima immagine campione dal frame, calcolata secondo il procedimento descritto nella Sezione 4.3. In base ai livelli di similarità si assegnano i pesi alle immagini, che vengono memorizzati nel vettore fattoreMoltiplicativo. La scelta dei valori di soglia è decisiva ai fini del raggiuntimento dell'obiettivo di assegnare pesi diversi alle immagini, a seconda del valore di distanza visuale tra frame e immagini. Le immagini molto simili sono quelle con un valore di distanza minore di soglia1. A queste immagini, e quindi ai loro tag, è assegnato un fattore moltiplicativo pari a 4. Questa soglia è stata settata in modo da garantire che quando la distanza tra due immagini sia inferiore alla soglia stessa, le due immagini siano

realmente simili. Questa situazione è stata verificata con una fase di testing; quando la distanza tra due immagini è inferiore a soglia1 è quasi certo che le due immagini rappresentino la stessa scena. Quando la distanza visuale è compresa tra soglia1 e soglia2, il fattore moltiplicativo assume il valore 3. Se, invece, la distanza tra il frame e un'immagine è maggiore di soglia3, il fattore moltiplicativo viene impostato a zero, cioè i tag dell'immagine non vengono considerati e non offrono alcun apporto nella procedura di suggerimento dei tag. La soglia3 è stata impostata in modo che se la distanza tra due immagini supera tale valore, è molto probabile che le due immagini non siano in alcun modo correlate. Viene adottata questa tecnica per poter utilizzare un valore di k piuttosito alto, in modo da selezionare un numero elevato di immagini da elaborare per suggerire i tag. Grazie a questo escamotage, però, se alcune immagini tra le top k sono troppo dissimili al frame, si fa in modo che esse non offrano alcun contributo. In questo modo, le immagini campione utilizzate per elaborare i tag non sono sempre k. Quando il valore di distanza tra frame e immagine è compreso tra soglia2 e soglia3, il fattore moltiplicativo assume un valore dato dall'equazione alla riga 12 dell'Algoritmo 4.5. Con questa equazione si imposta un fattore moltiplicativo che parte dal valore 3 e finisce al valore 0, e varia in modo direttamente proporzionale all'aumentare della misura di distanza tra frame e immagine campione.

Il passo successivo è il calcolo della rilevanza di tutti i tag presenti nelle k immagini campione più simili al frame del video in corso di elaborazione, sfruttando i fattori moltiplicativi precedentemente calcolati. La rilevanza di un tag è data dal numero di occorrenze del tag nelle k immagini più simili, moltiplicando ciascuna presenza per il fattore moltiplicativo associato all'immagine.

Quindi viene calcolata la frequenza a priori del tag nell'intera collezione, utilizzando l'equazione di Formula 4.4. In questa applicazione, questa formulazione

è stata modificata moltiplicando per 0.5 la parte sinistra dell'equazione, per far pesare meno la frequenza a priori del tag. Questo è stato fatto perchè il dataset di immagini ha dimensioni ridotte, visto che è difficile superare le 300 immagini. Inoltre, le immagini che compongono il dataset provengono da interogazioni realizzate su *Flickr* utilizzando tag o insiemi di tag che provengono dall'algoritmo di rafforzamento ed espansione semantica dei tag del video, per cui l'auspicio è che la maggior parte delle immagini, se non la totalità, sia inerente al contenuto del video e quindi dei suoi frame. Per questo, se un tag compare molte volte nell'intera collezione di immagini, potrebbe essere anche un tag particolarmente significativo per il video e quindi da suggerire. Moltiplicando la parte sinistra dell'equazione 4.4 per 0.5 si vuole proprio preservare questa proprietà che deriva da come vengono raccolte le immagini dal nostro sistema, senza, però, eliminare il succo dell'idea presente nell'articolo [Li09] che prescrive di computare la rilevanza di un tag secondo la Formula 4.3.

Al termine della valutazione della rilevanza dei tag, si ha un array che contiene un valore di rilevanza per ogni tag presente nelle k immagini elaborate. A questo punto si possono adottare due diverse strategie per suggerire i tag al frame del video. Una tecnica è quella di associare ad ogni frame del video un numero fisso di tag, che sono quelli con un valore più alto di rilevanza. In questo modo, però, si corrono due rischi; uno è quello di suggerire tag sbagliati per la necessità di doverne assegnare un numero fisso, andando a selezionare dei tag che hanno una buona posizione in graduatoria ma non una rilevanza abbastanza forte. Questo è il caso tipico dei frame al fuori del contesto a cui appartengono i tag del video. L'altro problema in cui si può incappare è esattamente l'opposto, cioè il caso in cui si suggeriscono pochi tag a causa del numero fisso, mentre ce ne sarebbero altri che hanno una buona rilevanza ma non vengono associati al frame. Per questi motivi, si preferisce adottare una tecnica diversa che consiste nell'associa-

re ad un frame i tag che superano un certo valore di rilevanza, lasciando intatte le possibiltà di non suggerire alcun tag ad una immagine, o viceversa associarne un numero consistente se sono molti i tag che oltrepassano il valore minimo di rilevanza. Una fase molto delicata è quella della scelta del valore di soglia della rilevanza, che determina quali tag vengono suggeriti ai frame del video.

Il procedimento descritto finora viene applicato a tutti i frame del video che identificano uno shot. Infatti, il video di partenza viene suddiviso in shot che contengono frame con la stessa distribuzione della luminosità. Per ogni shot viene estratto un frame al quale si applica la procedura di suggerimento dei tag che poi vengono associati all'intero shot, dato che si presume che le immagini appartenenti allo stesso shot rappresentino la stessa scena.

4.4.2 Come visualizzare i tag durante la visione del video

Dopo aver stabilito i tag da associare a un frame, e quindi al suo shot di appartenenza, si è pensato ad un modo per visualizzare questi suggerimenti. La tecnica adottata consiste nel creare dei sottotitoli in modo tale che durante una scena compaiano in sovraimpressione i tag suggeriti per quella scena. Per visualizzare i sottotitoli è stato creato un flie con estensione *.srt* che poi viene associato al video tramite un qualsiasi visualizzatore di filmati che abbia questa funzionalità. Per ogni scena da taggare si devono scrivere le seguenti righe all'interno del file *.srt*:

```
1
0:0:0,0 --> 0:0:0,75
LISTA TAG
```

La prima riga è il numero progressivo dello shot, mentre la seconda rappresenta l'intervallo temporale della scena. Il formato del tempo è ore:minuti:secondi,

centesimi di secondo; mentre la terza riga è l'elenco dei tag associati a quella scena. In questo caso, quindi, compariranno i tag indicati da LISTA TAG dall'inizio del filmato al 75-esimo centesimo di secondo.

Capitolo 5

Risultati sperimentali

5.1 Risultati del sistema per la rilevazione dei video simili

5.1.1 Valutazioni in termini di Average Precision

In questa sezione vengono descritti i risultati del sistema per la rilevazione della similarità tra video. La valutazione è stata eseguita su un archivio composto da 450 video, suddivisi in 30 classi, 2 per ciascuna delle 15 categorie di *YouTube*. Una classe è rappresentata da 15 video di diversa lunghezza e distinti tra loro. Il sistema realizzato calcola i descrittori di tutti i video contenuti nell'archivio, grazie ai quali viene costruito l'albero CFP che consente di recuperare le relazioni di similarità tra i video che compongono il dataset. Il CFP-Tree è stato implementato in modo che per ogni video contenuto nel dataset vengano suggeriti gli N video più simili.

La misura di valutazione del sistema è l'*Average Precision*, che viene calcolata

come il rapporto tra numero di suggerimenti corretti sul totale degli N video più simili. Un suggerimento è corretto se il video suggerito appartiene alla stessa classe di quello di cui si stanno valutando i video simili. Tale valutazione viene effettuata automaticamente dal sistema, grazie alla presenza di un file che contiene il ground truth.

Dopo aver raccolto i video che compongono il dataset e avere eseguito il programma che ne calcola i descrittori, la valutazione viene eseguita semplicemente lanciando il programma per la realizzazione del CFPTree. Questa darà in uscita l'*Average Precision*, calcolata in base al valore di N impostato dall'utente.

I risultati vengono presentati con 6 distinti valori di N; si parte dal valore N=1, cioè si valuta il video più simile ad ogni video contenuto nel dataset. Passando per i valori di N=4, 5, 7, 10, si arriva al valore di N=14 che rappresenta il massimo valore impostabile. Infatti, ogni classe è composta da 15 video, per cui il massimo numero di video che possono appartenere alla stessa classe di un singolo video è 14.

In Tabella 5.17 vengono mostrati i dati di *Average Precision* ottenuti al variare del valore di *N*. Tali valori sono stati calcolati per le singole categorie di *YouTube* e per l'intera raccolta di dati. In Figura 5.1 è stato inserito l'andamento dell'*Average Precision* per i distinti valori di N utilizzati in fase di testing. Si può notare che l'andamento ottenuto è piuttosto lineare; infatti l'*Average Precision* decresce quasi linearmente all'aumentare di *N*.

I valori ottenuti non possono essere confrontati con altri approcci già esistenti, visto che il dataset utilizzato è stato creato durante la realizzazione di questo sistema. Inoltre, in letteratura non sono presenti approcci che affrontino esattamente il problema come è stato trattato in questo lavoro di tesi; esistono, infatti, dei sistemi per la rilevazione dei *nearest neighbours* o per l'identificazione supervisionata di eventi all'interno di video. Per questi motivi non c'è la possibilità

	TOP1	TOP3	TOP5	TOP7	TOP10	TOP14
Auto & Vehicles	0,7	0,6	0,533	0,457	0,403	0,345
Comedy	0,833	0,789	0,653	0,595	0,533	0,452
Education	0,867	0,856	0,847	0,767	0,713	0,552
Entertainment	0,9	0,811	0,76	0,729	0,637	0,519
Film&Animation	0,9	0,822	0,793	0,752	0,673	0,579
Gaming	0,733	0,589	0,593	0,595	0,5	0,41
Howto & Style	0,933	0,789	0,767	0,724	0,62	0,526
Music	0,633	0,656	0,667	0,633	0,613	0,567
News & Politics	0,767	0,856	0,68	0,595	0,48	0,393
No-profit&Activism	0,833	0,778	0,707	0,681	0,64	0,576
People & Blogs	0,5	0,444	0,347	0,3	0,26	0,241
Pets & Animals	0,8	0,667	0,62	0,576	0,517	0,457
Science&Technology	0,367	0,3	0,26	0,271	0,22	0,198
Sport	0,833	0,767	0,667	0,6	0,537	0,452
Travel & Events	0,8	0,578	0,54	0,452	0,367	0,3
Totale	**0,76**	**0,679**	**0,625**	**0,582**	**0,511**	**0,438**

Tabella 5.1: Risultati complessivi per la similarità visuale

di confrontare il sistema realizzato, ma si può effettuare una valutazione dei risultati ottenuti andando a spiegare i motivi dei comportamenti adottati dal sistema.

L'andamento dell'*Average Precision* è comune per tutte le 15 categorie di *YouTube*, cioè decresce all'aumentare del valore di *N*. Le differenze stanno nei valori di *Average Precision* ottenuti, visto che in alcune categorie si hanno valori molto elevati, mentre per altre categorie i risultati non sono molto positivi. La spiegazione a questo comportamento è data dalla natura stessa dei filmati che rappresentano una categoria; per questo motivo entriamo nel dettaglio delle classi utilizzate per testare il sistema.

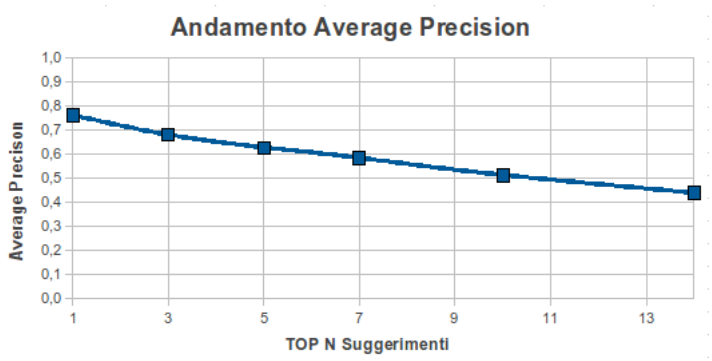

Figura 5.1: Grafico che rappresenta l'andamento dell'*Average Precision* sull'intera collezione dei dati. Sulle ordinate è presente l'*Average Precision* ottenuta con il numero di suggerimenti indicato sull'asse delle ascisse.

Auto & Vehicles Le 2 classi che rappresentano la categoria "Auto & Vehicles" sono le seguenti:

- *FormulaUno*: video che mostrano spezzoni di gare di Formula Uno;
- *FrecciaRossa*: video in cui è ripreso il treno Freccia Rossa.

L'*Average Precision* per questa categoria è leggermente inferiore alla media; questo è dato soprattutto dalla classe *FrecciaRossa* che contiene filmati di treni ripresi in situazioni diverse, per esempio treni in stazione, treni che transitano, treni fermi. Per questo motivo, all'aumentare del valore di N, si ha una diminuzione notevole dell'*Average Precision* visto che i video all'interno della stessa classe differiscono tra loro.

Comedy Questa categoria di *YouTube* è stata coperta con le seguenti classi:

- *AriaFresca*: spezzoni di puntate diverse della programma tv *Aria Fresca*.
- *ColoradoCafè*: spezzoni di puntate diverse del programma tv *Colorado Cafè*.

I dati di *Average Precision* per questa categoria sono leggermente superiori alla media generale; inoltre il comportamento è analogo per le due classi che compongono la categoria. Questo è facilitato dal fatto che una stessa classe contenga filmati che hanno lo studio televisivo comune. Una buona parte dei suggerimenti errati per i video di questa categoria è formata da video di altre trasmissioni tv che appartengono ad altre categorie, per la somiglianza degli studi televisivi in cui vengono registrate le diverse trasmissioni e degli eventi che si verificano nelle trasmissioni.

Education Le classi di video per la categoria "Education" sono:

- *Lezione*: lezioni diverse di matematica con un insegnante che scrive su una lavagna;
- *ManualeChitarra*: video che contengono lezioni di chitarra.

I risultati ottenuti con questa categoria sono decisamente sopra la media, soprattutto all' aumentare del valore di *N*. Tale comporatmento è dato soprattutto dalla presenza della classe *Lezione* che contiene 15 video abbastanza simili tra loro, per cui anche aumentando il numero di suggerimenti, rimane un'*Average Precision* piuttosto elevata.

Entertainment La categoria è coperta dalle seguenti classi:

- *Amici*: spezzoni di puntate diverse della trasmissione tv *Amici*;
- *DavidLetterman*: spezzoni di puntate diverse della trasmissione *David Letterman Show*.

Questa categoria consente di ottenere dati di *Average Precison* molto buoni, per la presenza degli studi televisivi comuni all'interno della singola classe di

filmati. Il comportamento è analogo in tutte e due le classi che compongono la categoria.

Film & Animation Le 2 classi di video che rappresentano la categoria sono le seguenti:

- *Pingu*: spezzoni di puntate diverse della serie tv *Pingu*;
- *Teletubbies* : spezzoni di puntate diverse della serie tv *Teletubbies*.

L'*Average Precision* ottenuta con questa categoria è superiore alla media complessiva, grazie agli ambienti simili in cui si sviluppano le puntate diverse delle due serie tv, che facilitano il ritrovamento di video simili. L'andamento dell'*Average Precision* è molto simile per entrambe le classi che compongono la categoria.

Gaming La categoria "Gaming" è rappresentata dalle seguenti classi:

- *Pes*: sequenze di partite del videogioco *Pro Evolution Soccer*;
- *SimCity*: sequenze di partite del videogioco *Sim City*.

Per questa categoria, i valori di *Average Precision* sono in linea con la media complessiva; inoltre l'andamento dell'*Average Precision* è analogo tra le due classi che rappresentano la categoria in questione.

Howto & Style Le classi di video che compongono la categoria "Howto & Style" sono:

- *Cooking*: video in cui è presente una persona che prepara una ricetta;
- *Makeup*: video in cui una ragazza si trucca.

I risultati in termini di *Average Precision* per questa categoria sono superiori alla media complessiva grazie al contributo della classe *Makeup*. Con tale classe si ottengono valori molto elevati di *Average Precision* per la reale similarità tra i filmati, visto che questi rappresentano sempre il primo piano del viso di ragazze, diverse tra loro, che si truccano.

Music La categoria "Music" è rappresentata dalle classi:

- *Madonna*: spezzoni di diverse tappe dello stesso tour della cantante Madonna;
- *XFactor*: spezzoni della trasmissione musicale *XFactor*.

L'andamento dell'*Average Precision* per questa categoria è leggermente diverso rispetto alla media, visto che tale dato rimane pressochè costante per tutti i valori di *N*. Questo comportamento è dato soprattutto dalla classe dei video di *Madonna* in cui la valutazione rimane molto buona al variare di *N*.

News & Politics Le classi di video che compongono la categoria sono le seguenti:

- *Parlamento*: video che rappresentano parti di sedute parlamentari;
- *TG4*: spezzoni estratti da diverse edizioni del TG4.

In questo caso i risultati in termini di *Average Precision* sono in linea con la media complessiva per entrambe le classi che compongono la categoria.

No-profit & Activism La categoria è rappresentata dalle classi:

- *Manifestazione*: video di una stessa manifestazione di protesta a Roma;

- *Protest*: video di una stessa manifestazione di protesta a Bruxelles.

I valori di *Average Precision* per questa categoria sono leggermente superiori alla media. Si ottengono risultati migliori per la classe *Protest* rispetto a *Manifestazione*, perchè i video della prima di queste due classi sono più simili tra loro.

People & Blogs I video che compongono la categoria appartengono alle seguenti classi:

- *OctopusPaul*: video del polpo Paul che prevede i risultati delle partite del Mondiale di calcio 2010;
- *Papa*: video del Papa ripreso alla finestra su Piazza San Pietro.

I risultati in termini di *Average Precision* sono inferiori alla media complessiva per la scarsa similarità tra i video che compongono la stessa classe. Inoltre, soprattutto per la classe *OctopusPaul*, i video hanno una durata più breve rispetto alla media dei video che compongono il dataset. Per questo motivo vengono calcolati meno descrittori di frame e c'è la possibilità che una certa percentuale di questi si possa ritrovare in un video più lungo appartenente a qualche altra classe rappresentata da molti descrittori in più.

Pets & Animals La categoria è composta dalle classi:

- *Delfino*: video che rappresentano delfini;
- *Elefante*: video che rappresentano elefanti.

I valori di *Average Precision* ottenuti per questa categoria sono in linea con la media complessiva calcolata su tutte le categorie. La categoria "Pets & Animals",

però, è una delle poche in cui le prestazioni tra le 2 classi che la compongono sono molto diverse tra loro. Per la classe *Delfino* si ottengono valori di *Average Precision* molto elevati; anche con $N=14$ l'*Average Precision* vale 0,738. D'altro canto, la classe *Elefante* porta a dei riscontri molto bassi in temini di *Average Precision*. Questo comportamento rispecchia la reale composizione delle singole classi; per la classe *Delfino* i video sono tutti simili tra loro, mentre nel caso di *Elefante* i contesti in cui sono ambientati i video sono molto diversi tra loro.

Science & Technology Le classi scelte per rappresentare questa categoria sono le seguenti:

- *Iphone*: video che mostrano alcuni problemi di funzionamento dell'Iphone;
- *Toshiba*: video di presentazione di diversi modelli di computer Toshiba.

Con questa categoria si hanno i peggiori risultati in termini di *Average Precision* su tutto l'insieme di dati. Il motivo è la scarsa similarità tra i video appartenenti alla stessa classe.

Sport I video della categoria "Sport" appartengono alle seguenti classi:

- *Basket*: spezzoni di partite di basket;
- *Sci*: spezzoni di gare di sci.

I valori di *Average Precision* ottenuti con questa categoria sono leggermente superiori alla media complessiva.

Travel & Events La categoria "Travel & Events" è formata da video appartenenti alle seguenti classi:

- *Etna*: video del vulcano Etna;

- *Niagara*: video delle cascate del Niagara.

I risultati ottenuti con questa categoria hanno un andamento leggermente inferiore alla media complessiva, soprattutto per valori più elevati di N.

In definitiva, i risultati riscontrati sull'intera collezione di dati sono piuttosto incoraggianti. Infatti nelle categorie in cui si riescono a trovare filmati piuttosto simili tra loro, i valori di *Average Precision* sono elevati, soprattutto per i casi in cui N è più basso. Le difficoltà di rilevazione avvengono per le categorie che contengono video appartenenti a classi per le quali non c'è sufficiente similarità tra i contenuti in termini di inquadrature, colori o distribuzione dei concetti. Questo può essere dimostrato dal fatto che i valori di *Average Precision* più elevati si ottengono per le categorie in cui i video appartenenti alla stessa classe hanno dei forti tratti comuni, come lo studio televisivo nel caso dei programmi TV o gli ambienti comuni presenti nella categoria "Education".

5.1.2 Tempi di esecuzione

Analizziamo i tempi di esecuzione del programma per il calcolo dei descittori dei video che compongono il dataset. Per tutte le prove realizzate è stato utilizzato un notebook con processore Intel Centrino 2 Duo P8600 2,40 GHz 4 GB RAM. L'intero dataset di 450 video contiene circa 19 ore e mezzo di filmati; per l'elaborazione complessiva del dataset sono necessarie circa 4 ore e mezzo. Questo vuol dire che per ogni secondo di esecuzione del programma vengono elaborati 4.333 secondi di video. I video non vengono ridimensionati ma sono trattati alle loro dimensioni originali. La maggior parte det filmati ha una risoluzione di 320x240, ma in alcuni casi la risoluzione è superiore.

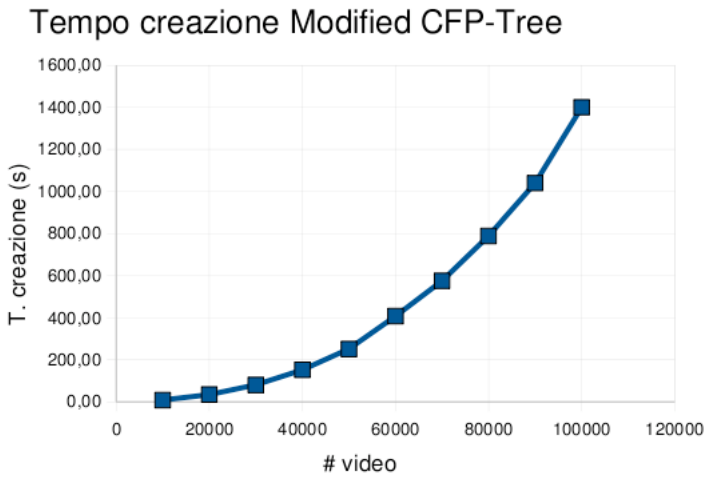

Figura 5.2: Tempi di costruzione del CFP-Tree all'aumentare del numero di video. L'andamento del tempo di costruzione dell'albero CFP rispetto alla quantità di video da analizzare è pressoché quadratico.

5.1.3 Tempi di costruzione del CFP-Tree

Durante la fase di testing realizzata, il CFP-Tree è stato utilizzato per l'indicizzazione di 450 video. Con un dataset di queste dimensioni i tempi di costruzione e interrogazione dell'albero CFP sono di circa un secondo. Per completezza, in Figura 5.2 viene presentato un grafico che mostra l'andamento del tempo di creazione del CFP-Tree, al crescere del numero di video che compongono il database. Questi dati sono stati ottenuti con un dataset generato casualmente, per simulare il funzionamento di un database di dimensioni estese.

5.2 Risultati del sistema per il suggerimento di tag agli shot del video

In questa sezione vengono presentati i risultati sperimentali riguardanti l'algoritmo per la ricerca della similitudine tra immagini, mirata al suggerimento di

tag a livello shot. Il funzionamento del sistema prevede il confronto di ogni keyframe, che rappresenta uno shot, con un insieme di immagini annotate scaricate da *Flickr*. In base alle immagini più simili al frame in questione vengono suggeriti dei tag, che costituiscono l'annotazione dell'intero shot contenente il frame. In questa sezione viene effettuata una valutazione della qualità dei tag suggeriti per ogni shot del video, in modo da fornire una vista sul funzionamento globale del sistema. La misura utilizzata per la valutazione è l'*accuracy* che, per un singolo filmato, viene calcolata come il rapporto tra il numero di tag corretti suggeriti e il numero complessivo di tag suggeriti. La correttezza di un singolo tag è stata ottenuta tramite annotazione manuale, realizzata a livello shot.

Per la valutazione del sistema sono stati utilizzati 60 filmati, 4 per ognuna delle 15 categorie con cui *YouTube* classifica i filmati. Per ciascun video è stata valutata l'accuratezza effettuando il rapporto tra il numero di suggerimenti corretti e il numero globale di suggerimenti, sommati su tutti gli shot che compongono il video. Le valutazioni sono state effettuate impostando 7 distinti valori della soglia utilizzata per il suggerimento dei tag, il cui funzionamento è stato descritto nella sottosezione 4.4.1. I valori di soglia utilizzati in fase di testing sono 1, 3, 4, 5, 7, 9, 11. Per giustificare l'utilizzo di tali valori di soglia, è utile ricordare che al momento del confronto tra il keyframe e tutte le immagini campione, viene assegnato un peso variabile tra 0 e 4 ai tag di quest'ultime, a seconda del grado di similarità tra il keyframe e le immagini campione. Per cui il peso di un tag viene dato dalla somma dei pesi del tag stesso, calcolato all'interno dell'insieme delle immagini più simili. Tale tag viene suggerito solo se il peso a lui associato supera il valore di soglia, che è quello che viene fatto variare in questa fase di test. In base a queste considerazioni, aumentando il valore di soglia verranno suggeriti sempre meno tag, ma verrà migliorata l'*accuracy*, visto che rimarranno solo i tag con un maggior livello di confidenza.

L'obiettivo è la ricerca della soglia che garantisca il miglior compromesso tra l'*accuracy* e il numero di tag corretti che vengono suggeriti ad ogni shot.

I risultati vengono presentati con una tabella per ognuna delle 15 categorie di *YouTube*. Per ogni tabella ci sono 5 colonne, cioè una per ciascuno dei 4 video che compongono la categoria, ed una aggiuntiva che riassume i risultati globali ottenuti sulla categoria. In ogni tabella ci sono 16 righe; la prima, denominata con **Durata**, indica la durata dei singoli video lungo le prime 4 colonne e la durata complessiva nell'ultima colonna. La seconda riga, # **shot**, è il numero di shot dei singoli video, con la somma di tali valori presentata nell'ultima colonna. Le rimanenti 14 righe sono l'*accuracy* e il numero medio di tag suggeriti per ogni shot, per i 7 valori di soglia utilizzati. In questo caso, i valori appartenenti alla colonna **Totale** sono la media dei valori presenti nelle altre 4 colonne della stessa riga. Vengono utilizzate due righe della tabella per ogni valore di soglia. Le righe denominate con **Acc. x** contengono i valori di acuratezza dei suggerimenti effettuati con il valore di soglia **x**, mentre quelle denominate con # **sugg. corr. x** presentano il numero medio di tag corretti suggeriti per ogni shot alla soglia **x**.

5.2.1 Analisi dei risultati sulle categorie di YouTube

Auto & Vehicles In Tabella 5.2 sono presentati i risultati ottenuti per la categoria "Auto & Vehicles". I filmati utilizzati sono i seguenti:

- Video1: Boeing in volo, che atterrano o che decollano;
- Video2: automobili Ferrari incidentate;
- Video3: aeroplani che atterrano o decollano all'aeroporto di St. Maarten;
- Video4: trattori incidentati e non.

	Video1	**Video2**	**Video3**	**Video4**	**Totale**
Durata	6'59"	4'55"	9'29"	4'44	26'07"
# shot	113	105	230	54	502
Acc. 1	0,510	0,248	0,413	0,455	0,407
# sugg. corr. 1	14,549	9,724	18,483	1,204	10,99
Acc. 3	0,709	0,414	0,751	0,744	0,655
# sugg. corr. 3	4,522	3,438	7,865	0,537	4,091
Acc. 4	0,824	0,563	0,866	0,741	0,749
# sugg. corr. 4	3,31	3,295	5,239	0,37	3,054
Acc. 5	0,914	0,604	0,888	0,733	0,785
# sugg. corr. 5	2,15	2,219	3,939	0,204	2,128
Acc. 7	0,924	0,723	0,922	0,875	0,861
# sugg. corr. 7	1,513	1,59	2,213	0,13	1,362
Acc. 9	0,9	0,759	0,932	1	0,898
# sugg. corr. 9	1,115	1,533	1,317	0,093	1,015
Acc. 11	0,936	0,849	0,933	1	0,93
# sugg. corr. 11	0,903	1,019	0,665	0,056	0,661

Tabella 5.2: Risultati categoria **Auto & Vehicles**

Gli insiemi di immagini associati ai 4 video di questa categoria contengono un buon numero di immagini effettivamente simili agli shot dei rispettivi video. Questo si può riscontrare con il numero medio di tag suggeriti per ogni shot, che è abbastanza elevato.

In questa categoria il valore di *accuracy* cresce all'aumentare del valore di soglia, fino ad un massimo di 0.93, mentre il numero medio di tag suggeriti per ogni shot decresce all'aumentare del valore di soglia. Il miglior compromesso si ottiene con la soglia impostata a 4 oppure a 5, quando si hanno valori di accuratezza interessanti, ma viene suggerito anche un buon numero di tag. In particolare, in questo caso, il valore di soglia da scegliere è 4, visto che, rispetto alla soglia impostata a 5, garantisce di suggerire quasi un tag in più per ogni shot, a fronte

	Video1	Video2	Video3	Video4	Totale
Durata	5'30"	0'21"	2'51"	0'28"	9'10"
# shot	141	5	43	12	201
Acc. 1	0,555	0,36	0,865	0,535	0,579
# sugg. corr. 1	1,801	3,6	10,163	6,417	5,495
Acc. 3	0,803	0,875	0,948	0,786	0,853
# sugg. corr. 3	0,723	1,4	6,767	1,833	2,681
Acc. 4	0,784	1	0,973	0,933	0,923
# sugg. corr. 4	0,539	1	5,884	1,167	2,148
Acc. 5	0,818	1	0,995	1	0,953
# sugg. corr. 5	0,383	1	5,023	0,333	1,685
Acc. 7	0,714	1	1	1	0,923
# sugg. corr. 7	0,177	0,6	2,698	0,083	0,89
Acc. 9	0,579	-	1	-	0,79
# sugg. corr. 9	0,078	0	1,349	0	0,415
Acc. 11	0,545	-	1	-	0,773
# sugg. corr. 11	0,043	0	0,605	0	0,162

Tabella 5.3: Risultati categoria **Comedy**

di una diminuzione di *accuracy* molto limitata.

Comedy La tabella 5.3 contiene i risultati per questa categoria. I video utilizzati per la valutazione della categoria "Comedy" sono i seguenti:

- Video1: esultanza particolare ad un goal realizzato in una partita di calcio;
- Video2: una serie di spezzoni di episodi curiosi avvenuti su campi di calcio;
- Video3: una divertente animazione della Statua della Libertà;
- Video4: una serie di immagini di gatti che sembrano parlare.

Per il testing di questa categoria sono stati scelti dei filmati che rappresentassero un concetto concreto, in modo da ottenere un insieme di tag da utilizzare per le interrogazioni a *Flickr* che consentisse di trovare immagini effettivamente inerenti con il contenuto dei filmati.

In questo caso, già a partire da un valore di soglia pari a 3, si ottengono dei valori di *accuracy* superiori a 0.85, cioè molto buoni. I risultati migliori si hanno per valori di soglia compresi tra 3 e 5, quando il compromesso tra *accuracy* e numero medio di tag suggeriti è molto buono. Per 2 dei 4 filmati utilizzati, a partire dal valore di soglia di 9, il sistema non riesce a suggerire alcun tag. Questo indica che non ci sono molte immagini campione realmente simili agli shot del video.

Education La Tabella 5.4 mostra i risultati della categoria "Education". I video utilizzati per il testing della categoria sono i seguenti:

- Video1: documentario sul Colosseo di Roma ai tempi dell'Impero Romano;
- Video2: giorno di lauree in un università inglese;
- Video3: video di presentazione dell'università di Oxford;
- Video4: video di presentazione della Princeton University.

In questo caso si può notare che l'*accuracy* aumenta fino ad un valore di soglia di 4, per poi tornare a scendere. Proprio la soglia 4 è il miglior compromesso tra numero di tag suggeriti e *accuracy*. In questo caso il numero medio di tag suggeriti ad ogni shot è più basso rispetto alle categoria precedenti; questo accade soprattutto per il Video3 che è quello di presentazione della Oxford University. In questo caso il sistema non riesce ad identificare abbastanza similitudini con le immagini campione, nonostante fossero presenti un buon numero di immagini valide, cioè con un contenuto informativo simile a quello degli shot del video.

	Video1	**Video2**	**Video3**	**Video4**	**Totale**
Durata	10'06"	6'01"	4'26"	2'02"	22'35"
# shot	208	134	116	40	498
Acc. 1	0,342	0,28	0,538	0,788	0,487
# sugg. corr. 1	4,125	7,448	1,043	3,25	3,967
Acc. 3	0,632	0,547	0,4	0,908	0,622
# sugg. corr. 3	2,572	2,985	0,052	1,725	1,834
Acc. 4	0,756	0,601	1	0,944	0,825
# sugg. corr. 4	2,159	1,694	0,017	1,275	1,286
Acc. 5	0,789	0,554	-	0,943	0,762
# sugg. corr. 5	1,615	0,918	0	0,825	0,84
Acc. 7	0,825	0,394	-	0,947	0,722
# sugg. corr. 7	0,885	0,209	0	0,45	0,386
Acc. 9	0,739	0,296	-	1	0,678
# sugg. corr. 9	0,409	0,06	0	0,3	0,192
Acc. 11	0,779	0,3	-	1	0,693
# sugg. corr. 11	0,259	0,022	0	0,175	0,114

Tabella 5.4: Risultati categoria **Education**

	Video1	**Video2**	**Video3**	**Video4**	**Totale**
Durata	0'36"	3'25	4'04"	3'14"	11'19"
# shot	15	19	33	68	135
Acc. 1	0,639	0,571	0,599	0,586	0,599
# sugg. corr. 1	1,533	3,158	9,151	4	4,461
Acc. 3	0,8	0,833	0,901	0,812	0,837
# sugg. corr. 3	0,267	1,053	8,576	2,029	2,981
Acc. 4	1	1	0,966	0,887	0,963
# sugg. corr. 4	0,067	0,368	7,818	1,382	2,487
Acc. 5	1	1	0,991	0,973	0,991
# sugg. corr. 5	0,067	0,158	6,485	1,059	1,942
Acc. 7	-	-	1	1	1
# sugg. corr. 7	0	0	3,151	0,412	0,891
Acc. 9	-	-	1	1	1
# sugg. corr. 9	0	0	1,242	0,103	0,336
Acc. 11	-	-	1	1	1
# sugg. corr. 11	0	0	0,061	0,044	0,026

Tabella 5.5: Risultati categoria **Entertainment**

Entertainment I risultati per questa categoria vengono schematizzati in Tabella 5.5. I video utilizzati per il testing sono i seguenti:

- Video1: una sequenza video che mostra il crash di un elicottero;
- Video2: l'elezione di Miss Universo;
- Video3: uno spezzone della trasmissione televisiva statunitense *Saturday Night Live Show*;
- Video4:uno spezzone di una trasmissione tv in cui vengono mostrati incontri di wrestling.

	Video1	Video2	Video3	Video4	Totale
Durata	3'15"	2'31"	2'27"	1'32"	9'45"
# shot	72	85	58	45	260
Acc. 1	0,852	0,579	0,432	0,297	0,54
# sugg. corr. 1	0,319	5,494	2,172	0,667	2,163
Acc. 3	-	0,824	0,955	1	0,926
# sugg. corr. 3	0	4,682	0,362	0,044	1,272
Acc. 4	-	0,925	1	1	0,975
# sugg. corr. 4	0	3,329	0,172	0,022	0,881
Acc. 5	-	0,985	1	-	0,993
# sugg. corr. 5	0	2,259	0,103	0	0,591
Acc. 7	-	1	1	-	1
# sugg. corr. 7	0	0,729	0,017	0	0,187
Acc. 9	-	1	-	-	1
# sugg. corr. 9	0	0,212	0	0	0,053
Acc. 11	-	1	-	-	1
# sugg. corr. 11	0	0,047	0	0	0,012

Tabella 5.6: Risultati categorie **Film & Animation**

I migliori compromessi tra *accuracy* e numero medio di suggerimenti per shot si hanno con i valori di soglia impostati a 4 o 5. In questo caso, si noti che per 2 video su 4 non si riesce a fornire alcun suggerimento per valori di soglia maggiori di 7, a causa dello scarso numero di immagini campione realmente correlate con il contenuto dei video.

Film & Animation In Tabella 5.6 sono presentati i risultati per questa categoria, ottenuti con i seguenti video di testing:

- Video1: trailer del film Avatar;
- Video2: trailer del film Mr. Fox;

- Video3: trailer del film New Moon;
- Video4: trailer del film Twilight.

In questo caso i migliori compromessi tra *accuracy* e numero medio di suggerimenti si hanno con i valori di soglia pari a 3 e 4. Infatti dal valore di soglia 5 in poi, vengono suggeriti pochissimi tag per l'intero video. Anche con le soglie 3 e 4 il numero di suggerimenti non è molto elevato a causa dello scarso numero di immagini realmente correlate con i video che si hanno tra le immagini campione. Questo comportamento è dato dalla difficoltà di reperire da *Flickr* immagini che coprano tutti gli shot di un trailer di un film, in cui appaiono molte scene diverse tra loro. Nel caso di test utilizzato, il trailer del film Avatar è quello che evidenzia maggiormente questa difficoltà, visto che nell'intero insieme di immagini campione ci sono solo 3 immagini realmente ritrovabili all'interno della sequenza video, nonostante la correttezza delle query realizzate su *Flickr* per il download delle immagini.

Gaming I risultati ottenuti con questa categoria sono elencati in Tabella 5.7. I video utilizzati per il testing sono i seguenti:

- Video1: trailer del videogioco Assassin's Creed;
- Video2: trailer del videogioco Need for Speed;
- Video3: spezzone di partita al videogioco Sim City;
- Video4: trailer del videogioco War Craft.

In questo caso i migliori compromessi tra *accuracy* e numero medio di tag suggeriti si hanno con i valori di soglia 3 e 4.

	Video1	**Video2**	**Video3**	**Video4**	**Totale**
Durata	3'45"	4'30"	3'24"	3'11"	14'50"
# shot	108	102	74	78	362
Acc. 1	0,38	0,416	0,546	0,555	0,474
# sugg. corr. 1	3,398	5,706	3,905	2,385	3,849
Acc. 3	0,862	0,784	0,859	0,897	0,851
# sugg. corr. 3	1,444	3,667	2,054	1,346	2,128
Acc. 4	0,938	0,819	0,915	0,938	0,903
# sugg. corr. 4	0,843	1,99	1,459	0,974	1,317
Acc. 5	0,948	0,864	0,981	0,94	0,933
# sugg. corr. 5	0,509	1,186	1,365	0,808	0,967
Acc. 7	0,962	1	1	1	0,991
# sugg. corr. 7	0,232	0,578	1,162	0,462	0,601
Acc. 9	1	1	1	1	1
# sugg. corr. 9	0,037	0,245	0,878	0,077	0,309
Acc. 11	-	1	1	1	1
# sugg. corr. 11	0	0,049	0,743	0,013	0,201

Tabella 5.7: Risultati categorie **Gaming**

	Video1	Video2	Video3	Video4	Totale
Durata	1'11"	4'56"	3'18"	3'54"	13'19"
# shot	25	116	79	99	319
Acc. 1	0,374	0,378	0,318	0,488	0,39
# sugg. corr. 1	3,68	6,052	1,342	4,586	3,915
Acc. 3	0,585	0,612	0,526	0,706	0,607
# sugg. corr. 3	1,52	3,233	0,38	2,94	2,018
Acc. 4	0,735	0,742	0,529	0,668	0,669
# sugg. corr. 4	1	2,431	0,228	1,889	1,387
Acc. 5	0,85	0,776	0,455	0,681	0,691
# sugg. corr. 5	0,68	2,06	0,127	1,293	1,04
Acc. 7	1	0,884	0,125	0,846	0,714
# sugg. corr. 7	0,32	1,448	0,025	0,556	0,446
Acc. 9	1	0,832	0	1	0,708
# sugg. corr. 9	0,28	0,897	0	0,495	0,418
Acc. 11	1	0,827	0	1	0,707
# sugg. corr. 11	0,28	0,621	0	0,354	0,314

Tabella 5.8: Risultati categoria **Howto & Style**

Howto & Style La Tabella 5.8 contiene i risultati dei test eseguiti sulla categoria "Howto & Style", ottenuti dall'analisi dei seguenti video:

- Video1: video che presenta l'Ipod Nano;
- Video2: video che insegna a cucinare una zuppa;
- Video3: filmato in cui viene mostrato come ottenere un makeup particolare;
- Video4: video in cui si insegna a cucinare la bistecca.

In questo caso i valori di *accuracy* sono leggermente inferiori rispetto ai casi precedenti a causa della maggior variabilità e al maggior numero di concetti pre-

	Video1	Video2	Video3	Video4	Totale
Durata	3'58"	3'34"	2'10"	4'05"	13'47"
# shot	105	50	47	82	284
Acc. 1	0,544	0,314	0,422	0,276	0,389
# sugg. corr. 1	2,486	2,98	3,383	1,085	2,484
Acc. 3	0,898	0,447	0,731	0,69	0,692
# sugg. corr. 3	0,419	0,34	0,809	0,354	0,481
Acc. 4	0,75	0,286	0,941	0,882	0,715
# sugg. corr. 4	0,057	0,08	0,34	0,183	0,165
Acc. 5	1	0	1	1	1
# sugg. corr. 5	0,10	0	0,19	0,122	0,103
Acc. 7	-	-	1	1	1
# sugg. corr. 7	0	0	0,022	0,024	0,012
Acc. 9	-	-	-	-	-
# sugg. corr. 9	0	0	0	0	0
Acc. 11	-	-	-	-	-
# sugg. corr. 11	0	0	0	0	0

Tabella 5.9: Risultati categoria **Music**

senti all'interno dei video. Questo induce ad un maggior numero di suggerimenti errati.

Music La Tabella 5.9 contiene i risultati ottenuti per questa categoria che è stata rappresentata dai seguenti video:

- Video1: video musicale di una canzone della cantante Lady Gaga;
- Video2: spezzone di una tappa di un tour musicale della cantante Lady Gaga;
- Video3: spezzone di una tappa di un tour della cantante Madonna;
- Video4: spezzone di una tappa di un altro tour della cantante Madonna.

	Video1	Video2	Video3	Video4	Totale
Durata	1'48"	2'08"	3'19"	2'13"	9'28"
# shot	42	54	85	33	214
Acc. 1	0,377	0,648	0,893	0,581	0,625
# sugg. corr. 1	4,214	9,37	5,11	2,606	5,325
Acc. 3	0,646	0,963	0,965	0,9	0,869
# sugg. corr. 3	1,214	5,352	2,235	0,818	2,405
Acc. 4	0,757	0,975	1	0,941	0,918
# sugg. corr. 4	0,667	3,593	1,647	0,485	1,598
Acc. 5	0,867	1	1	1	0,967
# sugg. corr. 5	0,31	2,444	1,177	0,242	1,043
Acc. 7	1	1	1	1	1
# sugg. corr. 7	0,095	0,926	0,682	0,152	0,464
Acc. 9	1	1	1	1	1
# sugg. corr. 9	0,048	0,148	0,353	0,03	0,145
Acc. 11	-	1	1	-	1
# sugg. corr. 11	0	0,056	0,118	0	0,044

Tabella 5.10: Risultati categoria **News & Politics**

In questo caso le migliori performance si hanno con le soglie 3 e 4, anch se il numero di suggerimenti è molto basso a causa dello scarso numero di immagini scaricabili da *Flickr* (e che quindi vanno a costituire l'insieme delle immagini campione) in grado di coprire la varietà di scene presenti in un video musicale o in un tour musicale.

News & Politics I risultati di questa categoria sono in Tabella 5.10 e sono stati ottenuti a partire dai seguenti video:

- Video1: video che mostra un'esplosione disastrosa in Korea;
- Video2: scontri al G20 di Londra;

	Video1	Video2	Video3	Video4	Totale
Durata	1'43"	2'23"	2'13"	3'05"	9'24"
# shot	43	65	68	78	254
Acc. 1	0,596	0,778	0,629	0,428	0,608
# sugg. corr. 1	4,256	0,108	1,868	4,256	2,622
Acc. 3	0,912	-	0,969	0,918	0,933
# sugg. corr. 3	1,93	0	0,912	1,141	0,996
Acc. 4	0,964	-	1	0,981	0,982
# sugg. corr. 4	1,233	0	0,574	0,679	0,622
Acc. 5	0,951	-	1	1	0,984
# sugg. corr. 5	0,907	0	0,353	0,423	0,421
Acc. 7	1	-	1	1	1
# sugg. corr. 7	0,512	0	0,103	0,051	0,167
Acc. 9	1	-	1	-	1
# sugg. corr. 9	0,256	0	0,059	0	0,079
Acc. 11	1	-	1	-	1
# sugg. corr. 11	0,14	0	0,029	0	0,042

Tabella 5.11: Risultati categoria **No-profit & Activism**

- Video3: la neve a Milano;
- Video4: lo tsunami di Phuket, in Thailandia.

Con questa categoria si hanno buoni risultati con i livelli di soglia 3, 4 e 5, con un'*accuracy* molto elevata ed un discreto numero di suggerimenti per ogni shot dei video.

No-profit & Activism La Tabella 5.11 contiene i risultati della categoria "No-profit & Activism", ottenuti a partire da seguenti casi di test:

- Video1: manifestazione a Boston;

	Video1	Video2	Video3	Video4	Totale
Durata	1'30"	1'52"	0'43"	4'13"	8'18"
# shot	44	40	15	118	217
Acc. 1	0,616	0,192	0,55	0,244	0,401
# sugg. corr. 1	5,932	7,8	7,333	1,746	5,703
Acc. 3	0,896	0,509	0,967	0,304	0,669
# sugg. corr. 3	2,727	4,025	3,867	0,347	2,742
Acc. 4	0,972	0,766	1	0,329	0,767
# sugg. corr. 4	1,591	3,025	2,8	0,212	1,907
Acc. 5	1	0,823	1	0,346	0,792
# sugg. corr. 5	0,955	1,625	2,133	0,153	1,217
Acc. 7	1	0,931	1	0,333	0,816
# sugg. corr. 7	0,159	0,675	1,6	0,051	0,585
Acc. 9	-	1	1	0,222	0,741
# sugg. corr. 9	0	0,1	1,133	0,017	0,313
Acc. 11	-	-	1	0	0,5
# sugg. corr. 11	0	0	0,6	0	0,15

Tabella 5.12: Risultati categoria **People & Blogs**

- Video2: manifestazione in Italia;
- Video3: spot Unicef per portare le vaccinazioni in Africa;
- Video4: spot Unicef per la scolarizzazione delle zone più povere.

I valori di soglia migliori per questa categoria sono 3 e 4, anche se il numero medio di suggerimenti per ogni shot è piuttosto basso rispetto ad altre categorie.

People & Blogs I risultati sperimentali ottenuti con questa categoria sono presentati in Tabella 5.12. La categoria è rappresentata dai seguenti video:

- Video1: video amatoriale di una prima comunione;

	Video1	Video2	Video3	Video4	Totale
Durata	2'05"	2'22"	1'33"	3'00"	9'03"
# shot	26	61	41	81	209
Acc. 1	0,474	0,281	0,905	0,563	0,556
# sugg. corr. 1	2,462	9,295	3,244	4,309	4,828
Acc. 3	0,679	0,546	1	0,783	0,752
# sugg. corr. 3	1,462	4,705	0,902	2,049	2,28
Acc. 4	0,8	0,74	1	0,841	0,845
# sugg. corr. 4	0,923	3,59	0,488	1,432	1,608
Acc. 5	0,765	0,805	1	0,88	0,863
# sugg. corr. 5	0,5	2,443	0,22	1	1,041
Acc. 7	0,714	0,977	-	0,875	0,855
# sugg. corr. 7	0,192	1,41	0	0,605	0,552
Acc. 9	1	1	-	0,926	0,975
# sugg. corr. 9	0,039	1,016	0	0,309	0,341
Acc. 11	-	1	-	0,882	0,941
# sugg. corr. 11	0	0,754	0	0,185	0,235

Tabella 5.13: Risultati categoria **Pets & Animals**

- Video2: intervista al Dalai Lama;
- Video3: video del polpo Paul che prevede un risultato durante i Mondiali di calcio 2010;
- Video4: video di un matrimonio celebrato con rito cattolico.

I migliori risultati si hanno con le soglie 4 e 5, quando si hanno buoni valori di *accuracy* e un buon numero di suggerimenti per ogni shot.

Pets & Animals La Tabella 5.13 contiene i risultati di questa categoria, ottenuti con i seguenti video:

- Video1: video di gatti;

	Video1	Video2	Video3	Video4	Totale
Durata	8'51"	1'14"	1'08"	1'35"	12'48"
# shot	253	26	31	44	354
Acc. 1	0,259	0,498	0,573	-	0,443
# sugg. corr. 1	6,079	5	8,129	0	4,802
Acc. 3	0,489	0,794	0,647	-	0,643
# sugg. corr. 3	1,822	1,038	3,839	0	1,675
Acc. 4	0,633	0,813	0,875	-	0,774
# sugg. corr. 4	1,328	0,5	2,935	0	1,191
Acc. 5	0,635	0,875	0,921	-	0,81
# sugg. corr. 5	0,854	0,269	2,258	0	0,845
Acc. 7	0,725	1	0,95	-	0,892
# sugg. corr. 7	0,439	0,115	1,226	0	0,445
Acc. 9	0,75	-	1	-	0,875
# sugg. corr. 9	0,202	0	0,871	0	0,268
Acc. 11	0,75	-	1	-	0,875
# sugg. corr. 11	0,095	0	0,548	0	0,161

Tabella 5.14: Risultati categoria **Science & Technology**

- Video2: video contenente immagini di cavalli;
- Video3: video di cani di razza Jack Russell;
- Video4: video di cani di razza Pinscher.

Come per la maggior parte delle categorie già analizzate, il miglior compromesso tra *accuracy* e numero di suggerimenti si ha con la soglia impostata a 4.

Science & Technology La Tabella 5.14 contiene i risultati della categoria "Science & Technology", rappresentata con i seguenti video:

- Video1: inaugurazione di un nuovo modello di Airbus;

	Video1	Video2	Video3	Video4	Totale
Durata	3'17"	1'28"	3'27"	1'56"	10'08"
# shot	48	25	67	35	175
Acc. 1	0,236	0,5	0,518	0,374	1,628
# sugg. corr. 1	8,854	2,6	3,075	3,429	4,49
Acc. 3	0,574	0,731	0,88	0,786	0,743
# sugg. corr. 3	5,833	0,76	1,313	2,629	2,634
Acc. 4	0,672	0,733	0,97	0,878	0,813
# sugg. corr. 4	4,354	0,44	0,955	2,257	2,002
Acc. 5	0,704	0,778	0,961	0,844	0,822
# sugg. corr. 5	2,979	0,28	0,746	1,543	1,387
Acc. 7	0,763	1	0,973	0,957	0,923
# sugg. corr. 7	1,208	0,12	0,537	0,629	0,624
Acc. 9	0,8	1	1	1	0,95
# sugg. corr. 9	0,5	0,04	0,284	0,286	0,278
Acc. 11	0,833	-	1	1	0,944
# sugg. corr. 11	0,313	0	0,179	0,086	0,145

Tabella 5.15: Risultati categoria **Sport**

- Video2: una serie di immagini di elicotteri;
- Video3: video di presentazione di un Iphone4G;
- Video4: video di presentazione di un nuovo cellulare della Nokia.

I migliori risultati si hanno per le soglie 4 e 5; il numero di suggerimenti non è molto elevato, soprattutto per il fatto che per il Video4 non si fornisce alcun suggerimento, nemmeno con la soglia più bassa dato che l'insieme di immagini campione è composto da sole 13 unità, nemmeno molto simili al reale contenuto dei frame del video.

Sport La Tabella 5.15 contiene i risultati della categoria "Sport", ottenuta con i seguenti video di test:

- Video1: sessione di test di Formula Uno;
- Video2: video in cui sono presenti surfisti;
- Video3: raccolta di immagini di pugilato;
- Video4: fasi di un match di tennis al torneo di Wimbledon.

I migliori riscontri numerici si hanno con le soglie 4 e 5.

Travel & Events La Tabella 5.16 riassume i risultati ottenuti per questa categoria. I video che rappresentano la categoria sono i seguenti:

- Video1: video che riprende il Colosseo di Roma;
- Video2: Palazzo Reale di Madrid;
- Video3: resort a Marsa Alam;
- Video4: cascate di Iguazu.

In questo caso i migliori risultati si hanno con la soglia impostata a 5. Questa è la categoria per la quale si ha il maggior numero di suggerimenti, anche con i valori più alti di soglia. Questo comportamento è dovuto al fatto che i set di immagini campione sono molto vasti e contengono molte immagini effettivamente simili agli shot dei video.

5.2.2 Analisi complessiva dei risultati

Dopo aver descritto i valori sperimentali ottenuti sui test realizzati sulle singole categorie di YouTube, vengono presentati i risultati globali dati dalla media dei valori ottenuti sui test delle singole categorie.

	Video1	Video2	Video3	Video4	Totale
Durata	3'36"	1'18"	0'58"	3'05"	8'57"
# shot	95	32	25	60	212
Acc. 1	0,772	0,609	0,382	0,669	0,608
# sugg. corr. 1	13,905	7,25	12,72	16,417	12,573
Acc. 3	0,888	0,852	0,594	0,827	0,79
# sugg. corr. 3	7,895	4,125	6,84	10,5	7,34
Acc. 4	0,9	0,843	0,628	0,879	0,813
# sugg. corr. 4	6,084	3,344	4,12	8,5	5,512
Acc. 5	0,93	0,933	0,693	0,921	0,869
# sugg. corr. 5	4,305	2,593	3,16	6,783	4,21
Acc. 7	0,962	0,958	0,741	0,977	0,91
# sugg. corr. 7	2,4	1,406	1,72	4,283	2,45
Acc. 9	0,995	1	0,912	1	0,977
# sugg. corr. 9	1,989	0,875	1,24	2,9	1,751
Acc. 11	1	1	0,909	1	0,977
# sugg. corr. 11	1,389	0,594	0,8	1,95	1,183

Tabella 5.16: Risultati categoria **Travel & Events**

I dati descritti in Tabella 5.17 sono originati dall'analisi delle annotazioni sugli shot di 60 video diversi. La durata complessiva dei video che compongono il dataset è di 3 ore 8'58", suddivisi in 4196 shot. I dati numerici sono descritti in Tabella 5.17, mentre la Figura 5.3 mostra l'andamento dell'*accuracy* al variare della soglia e la Figura 5.4 contiene il grafico con la curva data dal numero medio di suggerimenti per shot, al variare della soglia.

L'*accuracy* aumenta fino al valore di soglia 7, per poi tornare a decrescere, anche se il valore rimane pressochè costante intorno allo 0.9. Già con la soglia impostata a 4, però, si hanno buoni valori di *accuracy*. Il numero medio di suggerimenti per shot, invece, decresce con un andamento più deciso per i valori più bassi di soglia, mentre la diminuzione si fa più lieve con le soglie più elevate.

In definitiva, i compromessi migliori tra *accuracy* e il numero medio di suggerimenti si hanno per i valori di soglia impostati a 4 o 5. Infatti, con la soglia pari a 4, l'*accuracy* è già superiore all'84% e il numero medio di suggerimenti è superiore a 1.8 per ogni shot. Con la soglia impostata a 5, invece, l'*accuracy* arriva quasi al massimo valore assunto, mentre il numero medio di suggerimenti per shot rimane accettabile, scendendo fino a 1.297. Per valori superiori di soglia, l'*accuracy* rimane pressochè costante, mentre il numero medio di suggerimenti decresce troppo; per questo motivo tali valori di soglia non sono la scelta ottimale.

Il dataset utilizzato per i test è stato creato durante lo svolgimento di questo lavoro di tesi, per cui non vi è la possibilità di confrontare il sistema creato con approcci già esistenti. I valori di *accuracy* ottenuti, però, sono molto elevati e superano quelli di altri approcci che si sono occupati dello stesso tipo di problema. Questi valori molto buoni sono giustificati dal fatto che in questo approccio non viene suggerito un numero fisso di tag per ogni shot, ma i suggerimenti vengono effettuati solo se il tag supera un certo valore di confidenza, che è la

	Accuracy	# sugger.
Soglia =1	0,516	5,178
Soglia =3	0,763	2,504
Soglia=4	0,842	1,811
Soglia=5	0,881	1,297
Soglia=7	0,907	0,671
Soglia=9	0,899	0,393
Soglia=11	0,881	0,23

Tabella 5.17: Risultati complessivi per la fase di suggerimento dei tag a livello shot. Per ogni valore di soglia utilizzato, sono indicate l'*accuracy* e il numero medio di tag suggeriti per ogni shot, calcolate come media dei valori ottenuti sulle singole categorie.

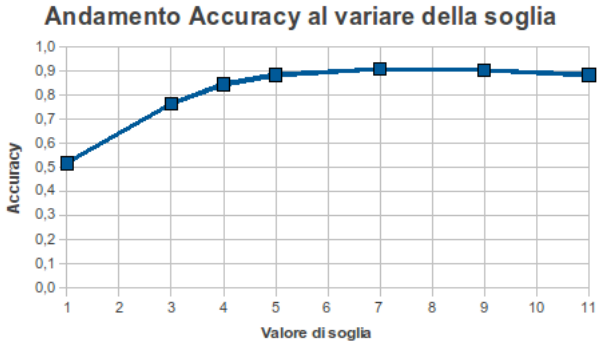

Figura 5.3: Andamento dell'*accuracy* al variare della soglia. I valori si riferiscono alle prove realizzate sull'intero dataset

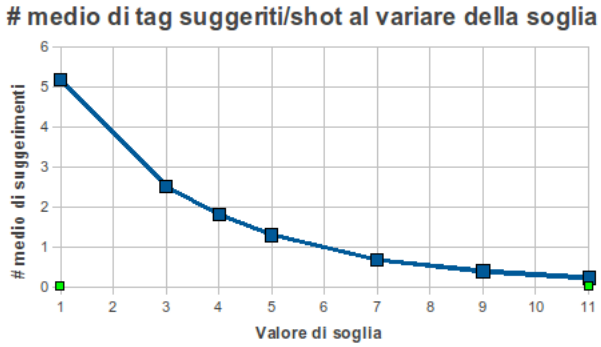

Figura 5.4: Numero medio di tag suggeriti per ogni shot al variare della soglia. I valori si riferiscono alle prove realizzate sull'intero dataset

soglia discussa finora. In questo modo, c'è la necessità di trovare il miglior compromesso tra *accuracy* e numero medio di suggerimenti, ma per valori di soglia pari a 4 o 5 si hanno dei comportamenti molto buoni. Basti pensare che non tutti gli shot necessitano di suggerimenti, perchè ve ne sono alcuni non correlati con il resto del video; per cui un numero medio di suggerimenti per shot di 1.811 (ottenuti con soglia pari a 4) è da considerarsi comunque valido.

Se fosse stato adottato il principio di suggerire un numero fisso di tag per ogni shot, ci sarebbero state alcune categorie di *YouTube* sulle quali si sarebbero ottenuti valori di *accuracy* molto bassi. Queste categorie sarebbero state quelle per cui si trovano poche immagini correlate realizzando le query su *Flickr*, cioè **Film & Animation**, **Music**, **No-profit & Activism**. Con l'approccio realizzato, l'*accuracy* rimane elevata, ma si ha un minor numero medio di suggerimenti per shot, rispetto alle altre categorie.

In definitiva, il sistema riesce effettivamente a ritrovare le immagini campione più simili agli shot dei video. Infatti, quando si hanno molte immagini campione correlate con il contenuto di un video si riesce a fornire un numero elevato di suggerimenti, mentre se ci sono poche immagini correlate, di conseguenza decresce il numero medio di suggerimenti. Questo è un indice di funzionamento del sistema; inoltre il fatto che l'*accuracy* rimanga sempre molto elevata dimostra che le immagini che vengono scelte come simili, che sono quelle dalle quali vengono estratti i tag da suggerire, sono effettivamente correlate con il contenuto del video.

5.2.3 Tempi di esecuzione

Analizziamo i tempi di esecuzione del programma per il calcolo dei descrittori. Per tutte le prove realizzate è stato utilizzato un notebook con processore Intel Centrino 2 Duo P8600 2,40 GHz 4 GB RAM. I tempi di esecuzione si riferiscono

alla generazione dei sottotitoli per un video. Il programma riceve in ingresso un video, una cartella di immagini ed un file che contiene i tag delle immagini. Per ogni immagine viene calcolato il descrittore, quindi il video viene suddiviso in shot e viene selezionato un keyframe per ogni shot. Per tale keyframe viene calcolato il descrittore, che poi è confrontato con tutti i descrittori delle immagini, per scegliere quelle più simili. Una volta selezionate le immagini più simili, vengono scelti i tag più rilevanti da tali immagini per associarli allo shot corrente.

Per l'elaborazione del descrittore delle immagini campione, in media sono necessari 0.25 secondi per ogni immagine. Per l'elaborazione completa di ogni shot del video, invece, sono necessari, mediamente, 0.34 secondi. Questo tempo è dato dal calcolo del descrittore del keyframe dello shot e dal confronto con tutti i descrittori delle immagini campione. Calcolando il numero complessivo di shot in cui sono stati suddivisi tutti i video che compongono il dataset e la durata complessiva di tali video, si ha che, mediamente, uno shot rappresenta 2.7 secondi di filmato. Grazie a tali considerazioni, si può dire che sono necessari 0.34 secondi di elaborazione per ogni 2.7 secondi di filmato, cioè 0.126 secondi di computazione per ogni secondo di filmato.

Capitolo 6

Conclusioni

In questo lavoro di tesi è stato affrontato il problema della ricerca della similarità visuale con due scopi distinti:

- realizzazione di un sistema per la ricerca e l'indicizzazione di video concettualmente correlati in un archivio di dimensione estesa;
- ricerca della similarità tra immagini, per trovare, all'interno di un insieme di foto annotate, quelle più simili allo shot di un video, con l'obiettivo di propagare i tag su tale shot. Questa seconda parte del lavoro sfrutta anche la tesi di Nencioni[Nencioni10] che fornisce l'insieme di immagini con cui confrontare gli shot del video.

Sono stati sviluppati due approcci distinti, data la profonda diversità delle questioni da affrontare. La linea comune è stata l'utilizzo dei *bag of visual words* e la combinazione tra descrittori globali e locali delle immagini.

L'applicazione specifica della prima parte del lavoro è la ricerca dei video correlati, contententi lo stesso tipo di concetto semantico. E' stata proposta una

soluzione che descrive i video combinando la distribuzione dei punti SURF rispetto ad una foresta di vocabolari visuali e informazioni di colore e li indicizza grazie all'utilizzo della struttura CFP-Tree. Le idee proposte sono piuttosto innovative, visto che in letteratura questo tipo di applicazione non è molto diffusa; piuttosto sono presenti ricerche sui *nearest neighbours* oppure lavori sull'identificazione supervisionata di eventi all'interno dei video. I risultati ottenuti sono incoraggianti e dimostrano l'efficacia della soluzione creata, affermando la validità dell'idea di combinare descrittori globali e descrittori locali delle immagini.

Per quanto riguarda la ricerca di immagini simili per il suggerimento di tag, è stato volutamente realizzato un sistema che privilegiasse il suggerimento di tag validi, a discapito del numero di suggerimenti. L'approccio creato utilizza la distribuzione dei punti SIFT più presenti, modellati utilizzando una serie di vocabolari visuali. In questo modo, viene creato un descrittore chiamato TOP-SIFT, che viene poi combinato con l'Edge Histogram Descriptor e con un correlogramma di colore. I risultati ottenuti sono molto soddisfacenti dato che la quasi totalità dei tag suggeriti risulta corretto. Questo dimostra la validità dell'idea che sta alla base del sistema realizzato. Infatti, per identificare dei concetti in un video, non ci si basa su un insieme di concetti già modellati e viene proposto uno schema completamente libero da qualsiasi forma di apprendimento. A partire da un video annotato, viene realizzata una validazione ed un'espansione delle annotazioni per realizzare delle query su *Flickr* che diano l'insieme di immagini annotate con cui cercare la similarità rispetto agli shot del video di partenza.

In definitiva, con l'esperienza maturata durante lo svolgimento del lavoro di tesi, si può affermare che l'utilizzo dei tag visualmente simili favorisce la qualità dei suggerimenti.

Infine, si può pensare ad un'applicazione concreta del progetto realizzato nella prima parte della tesi, quello relativo alla similarità tra video in archivi di grande dimensione. Nel sistema di suggerimento dei tag, la validazione e l'espansione dei tag del video di partenza viene effettuata, dall'algoritmo di Nencioni [Nencioni10], andando anche ad analizzare i *related videos* del video di partenza, forniti da *YouTube*. Questa fase potrebbe essere sostituita, o almeno integrata, con l'applicazione del sistema realizzato nella prima parte della mia tesi che si occupa proprio di trovare i video semanticamente correlati tra loro. Questo sistema potrebbe essere applicato anche a un insieme ristretto di video, come i *related videos* o i *related videos* dei *related videos*, per trovare i video più simili a quello da analizzare che quindi conterrebbero tag maggiormente correlati a quelli del video di partenza, consentendo così di ottenere una validazione ottimale e una prima espansione dei tag del video molto più robusta.

Appendice A

Strumenti utilizzati

L'Appendice viene utilizzata per descrivere gli strumenti più importanti utilizzati durante la realizzazione della tesi. Per inquadrare meglio la situazione, si ricordi che la parte di programma per la ricerca e l'indicizzazione di video visualmente simili utilizza una foresta di Bag-of-Visual Words ottenuta con punti SURF, combinandola con informazioni riguardanti la componente H della rappresentazione di colore HSV. Infine si utilizza il CFP-Tree come struttura di indicizzazione. Per tali motivi, in questa Appendice viene presentato il descrittore SURF in A.2. Nella sezione A.3, invece, viene descritto l'algoritmo di clustering applicato in modalità gerarchica per la costruzione della foresta di vocabolari di parole visuali. In A.5 è descritta la rappresentazione di colore HSV, mentre nella sezione A.7 viene presentato il CFP-Tree, utilizzato come struttura di indicizzazione. Il sistema per il suggerimento di tag a livello shot prevede l'utilizzo di un descrittore TOP-SIFT, integrato con l'Edge Histogram Descriptor e un descrittore ispirato al correlogramma di colore. Viene utilizzata la sezione A.1 per la presentazione del descrittore SIFT, la sezione A.4 per l'Edge Histogram Descriptor e la A.6 per il correlogramma di colore.

A.1 SIFT: Scale Invariant Feature Transform

I SIFT [Lowe04] sono feature estraibili da un'immagine per effettuare confronti affidabili tra viste distinte di un oggetto o tra scene distinte. Tali feature sono invarianti alla rotazione dell'immagine o alla ri-scalatura e consentono di ottenere confronti robusti anche nel caso di distorsioni affini, variazioni dei punti di vista 3D, aggiunta di rumore e variazione delle condizioni d'illuminazione.

Il costo di estrazione di queste feature è minimizzato utilizzando un approccio di filtraggi in cascata, nel quale le operazioni più costose sono applicate solo alle locazioni che superano un test iniziale. I passi più importanti per generare l'insieme di feature di un'immagine sono i seguenti:

1. Scale-space extrema detection;
2. Keypoint localization;
3. Orientation assigment;
4. Keypoint descriptor.

A.1.1 Scale-space extrema detection

Il primo passo per la rilevazione dei punti di interesse è l'identificazione delle locazioni e delle scale che possono essere ripetutamente assegnate con diverse viste dello stesso oggetto. Si vogliono locazioni che siano invarianti a cambi di scala; per rispettare questo tipo di richiesta il *kernel* più affidabile per lo *scale space* risulta essere una funzione Gaussiana. Per questo, lo *scale space* di un'immagine viene rappresentato come una funzione $L(x, y, \sigma)$, data dalla convoluzione tra una funzione Gaussiana $G(x, y, \sigma)$ e l'immagine di ingresso $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

dove * è l'operatore di convoluzione e:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}.$$

Per rilevare efficacemente le locazioni di punti di interesse stabili nello *scale space*, è stato proposto [Lowe99] di utilizzare un kernel per lo *scale space* dato dalla convoluzione tra una funzione che rappresenta la differenza di Gaussiane, invece della Gaussiana, e l'immagine d'ingresso. La differenza di Gaussiane viene calcolata come la differenza tra due funzioni Gaussiane calcolate a scale consecutive:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) =$$

$$L(x, y, k\sigma) - L(x, y, \sigma) \tag{A.1}$$

Un approccio efficiente per la costruzione di $D(x, y, \sigma)$ è quello mostrato in Figura A.1.

Determinazione degli estremi locali Per determinare i massimi e i minimi locali di $D(x, y, \sigma)$, ogni punto viene confrontato con i suoi otto vicini nell'immagine corrente e con i nove vicini delle immagini in scala superiore e inferiore (si veda Figura A.2).

Tale punto viene selezionato solo se più grande o più piccolo di tutti i suoi vicini. Un altro problema importante è la scelta della frequenza di campionamento dell'immagine e la scelta della scala alla quale si possono rilevare gli estremi in modo da massimizzarne la stabilità.

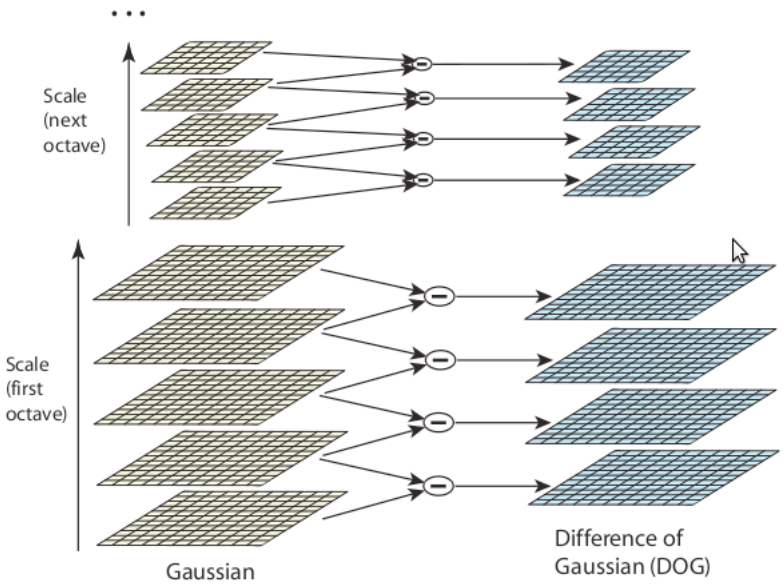

Figura A.1: Per ogni ottava dello scale space, viene ripetuta la convoluzione tra l'immagine iniziale e le Gaussiane per produrre l'insieme di immagini dello scale space mostrato sulla sinistra. Le immagini Gaussiane adiacenti vengono poi sottratte per produrre le immagini che rappresentano le Differenze di Gaussiane. Al termine di ogni ottava l'immagine Gaussiana viene sotto-campionata di un fattore 2 e il processo riparte.

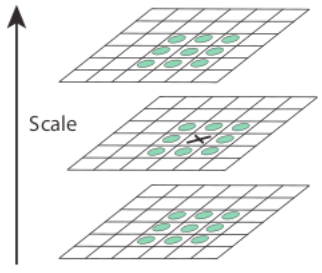

Figura A.2: I massimi e i minimi delle immagini delle differenze di Gaussiane sono determinati confrontando un pixel (indicato con X) e i 26 vicini: 8 alla scala corrente e 9+9 alle scale adiacenti (indicati con il cerchio).

A.1.2 Localizzazione accurata dei keypoint

Una volta determinati i keypoint candidati grazie al confronto con i pixel vicini, il passo successivo consiste nell'applicazione di una procedura che consenta di eliminare i punti con basso contrasto, e quindi sensibili al rumore, e quelli che sono mal localizzati lungo un edge. In [Brown02] è stato presentato un approccio per adattare una funzione 3D quadratica ai punti locali, per determinare la locazione del massimo. Questo metodo, che aumenta la stabilità del sistema, usa un'espansione di Taylor della funzione dello scale-space $D(x, y, \sigma)$; tale funzione viene shiftata in modo da avere l'origine nel punto sotto osservazione (*sample point*):

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{A.2}$$

ove D e le sue derivate vengono determinate nel *sample point*, mentre $\mathbf{x} = (x, y, \sigma)^T$ è l'*offset* da tale punto. La posizione degli estremi, $\hat{\mathbf{x}}$, viene determinata prendendo le derivate delle funzione suddetta rispetto ad $\mathbf{x}$ e uguagliandole a zero:

$$\hat{\mathbf{x}} = \frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}} \tag{A.3}$$

Il valore assunto dalla funzione nei punti estremi, $D(\hat{\mathbf{x}})$, è utile per eliminare gli estremi instabili che hanno basso contrasto. Questo si ottiene sostituendo l'equazione A.3 nell'equazione A.2, che risulta in:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x}.$$

A questo punto viene scelta una soglia per eliminare tutti i punti per i quali si verifica che $|D(\hat{\mathbf{x}})|$ è inferiore a tale valore di soglia.

Eliminazione dei response degli edge Per la stabilità non è sufficiente eliminare solo i keypoint con basso contrasto, ma si devono rifiutare anche quei punti debolmente definiti lungo gli edge, quindi molto più sensibili al rumore. Un picco della funzione DoG debolmente definito ha una grande curvatura principale lungo l'edge, ma ne ha una piccola definita nella direzione perpendicolare. La componente principale può essere determinata da una matrice Hessiana, $\mathbf{H}$, 2×2, calcolata nella posizione e alla scala del keypoint considerato:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{A.4}$$

Gli autovalori di $\mathbf{H}$ sono proporzionali alla componente principale di D. Seguendo l'approccio in [Harris88] si può evitare il calcolo esplicito degli autovalori, concentrandosi solo sul loro rapporto. Sia α l'autovalore con il modulo più grande e β quello più piccolo. Quindi, la somma degli autovalori può essere calcolata a partire dalla traccia di $\mathbf{H}$, e il loro prodotto a partire dal determinante:

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$Det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Nel caso sfortunato in cui il determinante è negativo, le curvature hanno segni differenti, così che il punto viene scartato non essendo un estremo. Sia r il rapporto tra l'autovalore di modulo maggiore e quello dal modulo minore, tale che $\alpha = r\beta$. Quindi,

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

e tale formulazione dipende solo dal rapporto tra gli autovalori, piuttosto che dal loro singolo valore. Il valore $(r+1)^2/r$ è al minimo quando i due autovalori sono identici e aumenta al crescere di r. Di conseguenza, per verificare che il rapporto tra le curvature principali sia al di sotto di un valore di soglia, r,si deve verificare solo la seguente:

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r}$$

Un valore attendibile per la soglia è $r = 10$, che elimina i keypoint che hanno un rapporto maggiore di 10 tra le curvature principali.

A.1.3 Assegnazione dell'orientazione

L'approccio per determinare le orientazioni più stabili per i keypoint prevede di utilizzare la scala del keypoint per selezionare l'immagine Gaussiana, L, con la scala più vicina, in modo che i calcoli vengano effettuati in modalità scala-invariante. Per ogni immagine campione, $L(x, y)$, il modulo del gradiente, $m(x, y)$, e l'orientazione del gradiente, $\theta(x, y)$, vengono pre-computati usando la differenza tra pixel:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

A partire dall'orientazione del gradiente dei punti prelevati all'interno di una regione attorno al keypoint viene costruito un istogramma delle orientazioni. Tale istogramma è costituito da 36 bin che coprono i 360 gradi assumibili dall'orientazione. Ogni campione aggiunto all'istogramma viene pesato con il modulo del suo gradiente e con una finestra circolare con pesi Gaussiani con σ pari a

1.5 volte la scala del keypoint. I picchi nell'istogramma delle orientazioni corrispondono alle orientazioni dominanti dei gradienti locali. Viene identificato il picco più alto nell'istogramma e poi vengono considerati tutti gli altri picchi che hanno un valore pari almeno all'80% del valore massimo trovato, che vanno quindi a formare un keypoint con quella orientazione. Di conseguenza, per le locazioni con molti picchi di valore simile all'interno dell'istogramma, vengono creati diversi keypoint con la stessa posizione e la stessa scala, ma con orientazione differente. Infine viene costruita una parabola con i tre punti più vicini ad ogni picco per interpolare la posizione del picco stesso con maggiore precisione.

A.1.4 Il descrittore locale dell'immagine

Con tutte le operazioni precedenti, sono stati assegnati una posizione all'interno dell'immagine, una scala ed un'orientazione ad ogni keypoint. Questi parametri compongono un sistema di coordinate 2D all'interno del quale sono descritte le regioni locali dell'immagine, fornendo di conseguenza l'invarianza a tali parametri. Il passo successivo è il calcolo di un descrittore per le regioni locali dell'immagine che sia altamente distintivo e il più invariante possibile rispetto a modifiche, quali variazioni di illuminazione o cambiamenti del punto di vista 3D. Un approccio per risolvere questo problema è stato proposto in [Edelman97]. Questo approccio è basato su un modello di visione biologica, in particolare viene considerato il caso di complessi neuroni della corteccia visuale e l'orientazione e la frequenza con cui rispondono a gradienti.

La Figura A.3 illustra la procedura di calcolo del descrittore del keypoint. La finestra Gaussiana, che in figura è rappresentata dalla circonferenza, viene utilizzata per dare enfasi minore ai gradienti lontani dal centro del descrittore, dato che questi possono essere maggiormente soggetti ad errori. Il descrittore del keypoint viene mostrato nella parte destra di Figura A.3. Il descrittore con-

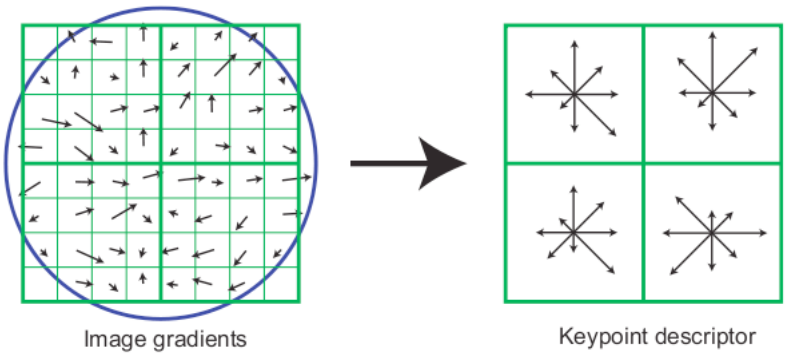

Figura A.3: Il descrittore di un keypoint viene creato calcolando prima il modulo e l'orientazione del gradiente in ogni punto campione all'interno dell'immagine utilizzando un'area attorno alla posizione del keypoint, come mostrato sulla sinistra. Questi vengono pesati con una finestra Gaussiana, rappresentata dalla circonferenza presente nell'immagine di sinistra. I punti identificati, vengono poi accumulati in un istogramma delle orientazioni per ogni insieme di 4×4 sotto-regioni, come mostrato sulla destra, dove la lunghezza di ogni vettore corrisponde alla somma dei moduli dei gradienti vicini a quella direzione, interni alla regione. La figura mostra un esempio di vettore di descrittori 2×2, calcolato a partire da un insieme di punti con cardinalità 8×8. In realtà, viene utilizzato un vettore di descrittori 4×4, ottenuto a partire da 16×16 punti campione.

sente variazioni significative della posizione del gradiente creando istogrammi delle orientazioni su sotto-regioni 4×4. La figura mostra 8 direzioni per ogni istogramma delle orientazioni, con la lunghezza di ogni vettore che corrisponde al modulo di quell'ingresso dell'istogramma.

Un problema da evitare è l'effetto dei contorni nei quali il descrittore cambia improvvisamente a causa dello spostamento di un punto da un istogramma ad un altro o da un'orientazione ad un'altra. Quindi, per distribuire il valore del gradiente di ogni punto negli istogrammi adiacenti, viene utilizzata l'interpolazione tri-lineare. In pratica, ogni ingresso in un bin dell'istogramma viene moltiplicato per un peso di $1 - d$ per ogni dimensione, dove d è la distanza del punto dal valore centrale del bin, misurata in unità dello spazio dei bin dell'istogramma.

Il descrittore è formato da un vettore contenente i valori di tutti gli ingressi dell'istogramma delle orientazioni, corrispondenti alle lunghezze dei vettori nella parte destra di Figura A.3. La figura mostra un vettore di 2×2 istogrammi delle orientazioni, anche se la rappresentazione reale è stata ottenuta con un vettore di 4×4 istogrammi, ciascuno composto da 8 bin che rappresentano le orientazioni. Per questo si ottengono vettori di feature di $4 \times 4 \times 8 = 128$ dimensioni per ogni keypoint. Infine, il vettore di feature viene modificato per ridurre gli effetti delle variazioni d'illuminazione. Per prima cosa, il vettore viene normalizzato alla lunghezza unitaria; il motivo di questa operazione è che una variazione nel contrasto dell'immagine in cui il valore di ciascun pixel viene moltiplicato per una costante moltiplicherà il gradiente per la stessa costante, quindi questo cambio di contrasto viene eliminato da questa normalizzazione del vettore. Inoltre, variazioni della luminosità, in cui ad ogni pixel dell'immagine viene aggiunta una costante, non modificano i valori del gradiente, dato che questi vengono calcolati come differenza tra pixel. Di conseguenza, il descrittore è invariante ai cambiamenti affini di illuminazione. Comunque, possono avere luogo variazio-

ni di luminosità non lineari a causa della saturazione della camera, o a causa di cambi di illuminazione che coinvolgono le superfici 3D con quantità distinte per le varie orientazioni. Questi effetti possono condurre a grandi variazioni nel modulo di alcuni gradienti, ma affliggono in misura minore l'orientazione dei gradienti. Di conseguenza, viene ridotta l'influenza dei gradienti più alti, stabilendo una soglia nel vettore unitario delle feature, in modo che non assumano valori maggiori a 0.2.

A.2 SURF: Speeded Up Robust Features

SURF [SURF] è un rilevatore e descrittore di punti di interesse invariante a variazioni di scala e a rotazioni. Le caratteristiche migliori sono la robustezza e la possibilità di essere calcolato e confrontato molto velocemente. Viene costruito sull'integrale delle immagini tramite una misura basata sulla matrice Hessiana per il rilevatore e su un descrittore basato sulla distribuzione.

A.2.1 Fast Hessian Detector

Il rilevatore (detector) SURF è basato sulla matrice Hessiana, grazie alle sue buone prestazioni in termini di tempi di calcolo e accuratezza. Dato un punto $\mathbf{x} = (x, y)$ in un'immagine, la matrice Hessiana $\mathcal{H}(\mathbf{x}, \sigma)$ in $\mathbf{x}$ alla scala σ è definita come segue:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{A.5}$$

dove $L_{xx}(\mathbf{x}, \sigma)$ è la convoluzione della derivata Gaussiana al secondo ordine $\frac{\partial^2}{\partial x^2} g(\sigma)$ con l'immagine I nel punto $\mathbf{x}$. I filtri Gaussiani di questo genere, però,

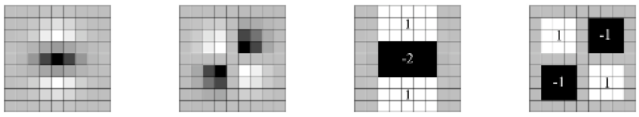

Figura A.4: Da sinistra a destra: la derivata parziale Gaussiana al secondo ordine nella direzione y e nella direzione xy e l'approssimazione usando il filtro "box". Le regioni in grigio assumono il valore zero.

non sembrano essere l'ideale in tutti i casi, perciò può essere utilizzata un'approssimazione che si ottiene applicando un filtro "box" 9×9, come mostrato in Figura A.4. In questo caso si ha un'approssimazione della Gaussiana con $\sigma = 1.2$; tali approssimazioni vengono denotate con D_{xx}, D_{xy} e D_{yy}. C'è la necessità di bilanciare i pesi relativi nell'espressione del determinante dell'Hessiana con $\frac{|L_{xy}(1.2)|_F|D_{xx}(9)|_F}{|L_{xx}(1.2)|_F|D_{xy}(9)|_F} = 0.912... \simeq 0.9$, dove $|x|_F$ è la norma di Frobenius. Questa approssimazione porta a:

$$\det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2. \tag{A.6}$$

Per calcolare lo *scale space* vengono usualmente utilizzate tecniche di *image pyramids*. Alle immagini vengono ripetutamente applicati smoothing con la Gaussiana e successivamente vengono sotto-campionate per ottenere i livelli più alti della piramide. Grazie all'utilizzo del filtro "box" e dell'immagine integrale, non c'è la necessità di applicare iterativamente lo stesso filtro all'output di un livello precedentemente filtrato, piuttosto possono essere applicati filtri di ogni dimensione direttamente all'immagine originale. Per ottenere diversi livelli di scala, quindi, vengono applicati anche filtri di diverse dimensioni con le stesse modalità del filtro 9×9, piuttosto che ridurre iterativamente le dimensioni dell'immagine. L'output del filtro 9×9 è considerato il livello di scala iniziale, al quale riferirsi con un valore di scala $s = 1.2$, perchè corrispondente alle derivate Gaussiane con $\sigma = 1.2$. I livelli successivi sono ottenuti filtrando l'immagine

con maschere gradualmente più grandi, prendendo in considerazione la natura discreta dell'immagine integrale e la struttura del filtro utilizzato. Entrando nel dettaglio, vengono applicati filtri di dimensione 9×9, 15×15, 21×21, 27×27, e via dicendo. Il rapporto di filtraggio rimane costante anche dopo l'operazione di scalatura; per questo, per esempio, il filtro 27×27 corrisponde a $\sigma = 3 \times 1.2 = 3.6 = s$. Allo scopo di localizzare i punti di interesse nell'immagine, viene applicata una *non-maximum suppression* con una finestra $3 \times 3 \times 3$. I massimi del determinante della matrice Hessiana vengono poi interpolati in scala e nello spazio delle immagini con il metodo proposto da [Brown02].

A.2.2 Descrittore SURF

Il primo passo verso la creazione del descrittore SURF consiste nel fissare un'orientazione riproducibile, basata sull'informazione di una regione circolare attorno al punto di interesse. Quindi, viene costruita una regione quadrata allineata all'orientazione selezionata e viene estratto il descrittore SURF da essa. Di seguito sono analizzati nel dettaglio i due passi sovraccennati.

Assegnamento dell'orientazione Allo scopo di rendere i punti di interesse invarianti alle rotazioni, viene identificata un'orientazione riproducibile per essi. A questo scopo, viene calcolata la risposta ottenuta all'applicazione di una *Haar-wavelet* nelle direzioni x e y. La situazione è rappresentata in figura A.5.

Questa operazione è compiuta in un intervallo circolare di raggio $6s$ attorno al punto di interesse, ove s è il valore di scala al quale il punto di interesse era stato rilevato. Inoltre, il passo di campionamento dipende dalla scala e viene scelto pari a s; anche le risposte della wavelet vengono calcolate al valore di scala corrente s. In base a questo, ad alti valori di scala, la dimensione della wavelet è grande; quindi viene utilizzata di nuovo l'immagine integrale per ottenere un

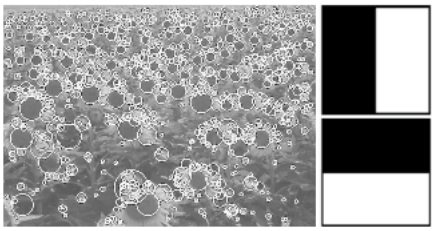

Figura A.5: Sulla sinistra ci sono i punti di interesse rilevati per un'immagine che rappresenta un campo di fiori. Con questo tipo di scena si può notare la natura delle feature ottenute grazie ad un detector basato sulla matrice Hessiana. Sulla destra le Haar-wavelet utilizzate per i SURF.

filtraggio veloce. Per calcolare la risposta (*response*) nelle direzioni x e y a qualsiasi scala, sono necessari solo sei passi. Una volta che i *response* della wavelet sono stati calcolati e pesati con una Gaussiana ($\sigma = 2.5s$) centrata nel punto di interesse, tali *response* vengono rappresentati come vettori in uno spazio in cui il *response* orizzontale viene rappresentato lungo l'asse delle ascisse, mentre il *response* verticale lungo l'asse delle ordinate. L'orientazione dominante viene stimata calcolando la somma di tutti i *response* interni ad una finestra scorrevole che copre un angolo di $\frac{\pi}{3}$. I *response* orizzontali e verticali interni alla finestra vengono sommati, in modo che tali 2 *response* sommati portino ad un nuovo vettore. Il vettore, tra i due, con la maggior lunghezza, dà la sua orientazione al punto di interesse.

Componenti del descrittore Per l'estrazione del descrittore, il primo passo consiste nel costruire una regione quadrata centrata intorno al punto di interesse ed orientata lungo l'orientazione scelta al passo descritto nel paragrafo precedente. La dimensione della finestra è $20s$. Esempi di tali regioni quadrate possono essere visualizzate in Figura A.6.

La singola regione viene suddivisa in sotto-regioni più piccole per un totale di 4×4; per ognuna di queste sotto-regioni sono calcolate poche semplici featu-

Figura A.6: Esempio di un'immagine in cui viene mostrata la dimensione del descrittore a distinti valori di scala

re con campionamento dei punti regolarmente spaziati di 5×5. Si identifica il *response* della *Haar wavelet* nella direzione orizzontale con d_x e quello nella direzione verticale con d_y. Per accrescere la robustezza del descrittore nei confronti di deformazioni geometriche e di errori di localizzazione, i *response* d_x e d_y vengono precedentemente pesati con una Gaussiana ($\sigma = 3.3$) centrata nel punto di interesse. Successivamente questi valori vengono sommati lungo tutte le sotto-regioni, in modo da formare un primo insieme di ingressi del vettore delle feature. Allo scopo di memorizzare informazioni sulla polarità dei cambi di intensità, vengono estratte anche le somme dei valori assoluti dei *response*, $|d_x|$ e $|d_y|$. Quindi, ogni sotto-regione ha un vettore di descrittori a 4 dimensioni $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Questo risulta in un vettore di descrittori a 64 dimensioni date dalle 4 dimensioni per ciascuna delle 4×4 sotto-regioni.

La Figura A.7 mostra il funzionamento del descrittore per tre pattern distinti.

Oltre al descrittore SURF a 64 dimensioni, esiste anche quello a 128 dimensioni. Il funzionamento è analogo al caso appena descritto, con la sola differenza che le somme di d_x e di $|d_x|$ vengono calcolate separatamente per $d_y < 0$ e per $d_y \geq 0$. Lo stesso accade anche per la somma di d_y e di $|d_y|$, per cui il numero delle feature raddoppia. Tale descrittore è più distintivo, e non molto più lento da calcolare, ma è più lento da confrontare a causa del maggior numero di dimensioni.

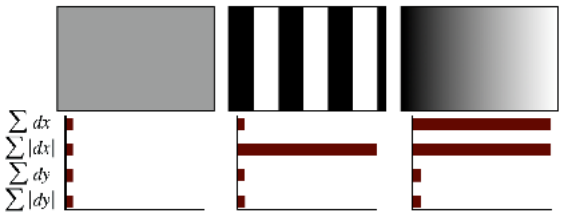

Figura A.7: Tre regioni distinte con i corrispondenti valori assunti dagli ingressi dei descrittori. Nel caso a sinistra, in presenza di una regione omogenea tutti i valori sono prevalentemente bassi. Parte centrale: in presenza di variazioni lungo l'asse x, $\sum |d_x|$ è l'unico valore molto alto. Nell'immagine più a destra si ha una regione con il valore di intensità che cresce gradualmente lungo l'asse x; in questo caso sia $\sum d_x$ che $\sum |d_x|$ assumono valori alti.

A.3 K-Means clustering

In questa sezione viene presentato l'algoritmo K-Means per la suddivisione di gruppi di oggetti in K partizioni. In questo lavoro di tesi l'algoritmo K-Means è stato applicato normalmente nella parte di progetto dedicata al suggerimento dei concetti agli shot dei video, mentre è stato applicato in modalità *gerarchica* nella parte di progetto riguardante la similarità visuale. Più precisamente è stato utilizzato un *clustering gerarchico divisivo* che, a partire dall'insieme di oggetti li va a suddividere in gruppi sempre più piccoli. In questo caso viene applicato l'algoritmo K-Means andando a suddividere l'insieme di dati in K parti, quindi è stato applicato nuovamente l'algoritmo su tutte le parti ottenute al passo precedente.

L'algoritmo K-Means è un algoritmo di clustering che permette di suddividere gruppi di oggetti in K partizioni sulla base dei loro attributi. È una variante dell'*Algoritmo di aspettazione-massimizzazione* il cui obiettivo è determinare i K gruppi di dati generati da distribuzioni Gaussiane. Si assume che gli attributi degli oggetti possano essere rappresentati come vettori, e che quindi formino uno spazio vettoriale.

A.3.1 Descrizione dell'algoritmo

L'obiettivo che l'algoritmo si prepone è di minimizzare la varianza totale intra-cluster. Ogni cluster viene identificato mediante un centroide o punto medio. L'algoritmo segue una procedura iterativa. Inizialmente crea K partizioni e assegna ad ogni partizione i punti d'ingresso o casualmente o usando alcune informazioni euristiche. Quindi calcola il centroide di ogni gruppo e costruisce una nuova partizione associando ogni punto d'ingresso al cluster il cui centroide è più vicino ad esso. Successivamente, vengono ricalcolati i centroidi per i nuovi cluster e così via, finché l'algoritmo non converge. Si procede ora ad una descrizione più formale dell'algoritmo.

Dati N oggetti con i attributi modellizzati come vettori in uno spazio vettoriale i-dimensionale, si definisce $X = X_1, X_2, ..., X_N$ come insieme degli oggetti. Si ricordi che si definisce partizione degli oggetti il gruppo di insiemi $P = P_1, P_2, ..., P_K$ che soddisfano le seguenti proprietà:

- $\bigcup_1^K P_i = X$: tutti gli oggetti devono appartenere almeno ad un cluster;
- $\bigcap_1^K P_i = \emptyset$: ogni oggetto può appartenere ad un solo cluster;
- $\emptyset \subset P_i \subset X$: un cluster deve contenere almeno un oggetto e nessun cluster può contenere tutti gli oggetti.

Deve valere anche che $1 < K < N$; non avrebbe infatti senso né cercare un solo cluster né avere un numero di cluster pari al numero di oggetti. Una partizione viene rappresentata mediante una matrice $U \in \mathbb{N}^{K \times N}$, il cui generico elemento $u_{ij} = 0, 1$ indica l'appartenenza dell'oggetto j al cluster i. Si indica con $C = C_1, C_2, ..., C_K$ l'insieme dei K centroidi. A questo punto si definisce la

funzione obiettivo come:

$$V(U,C) = \sum_{i=1}^{K} \sum_{X_j \in P_i} ||X_j - C_i||^2$$

e di questa viene calcolato il minimo tramite la seguente procedura iterativa:

1. Generazione di U_v e C_v casuali;
2. Calcolo di U_n che minimizza $V(U, C_v)$;
3. Calcolo di C_n che minimizza $V(U_v, C)$;
4. Se l'algoritmo converge STOP, altrimenti $U_v = U_n$, $C_v = C_n$ e si riparte dal passo 2.

Tipici criteri di convergenza sono i seguenti:

- Nessun cambiamento della matrice U;
- la differenza tra i valori della funzione obiettivo in due iterazioni successive non supera una soglia prefissata.

A.3.2 Pregi e difetti dell'algoritmo

L'algoritmo ha acquistato notorietà dato che converge molto velocemente. Infatti, si è osservato che generalmente il numero di iterazioni è minore del numero di punti. Comunque, di recente, in [Arthur06] è stato mostrato che esistono certi insiemi di punti per i quali l'algoritmo impiega un tempo superpolinomiale ($2^{\Omega(\sqrt{n})}$) a convergere. In termini di prestazioni l'algoritmo non garantisce il raggiungimento dell'ottimo globale. La qualità della soluzione finale dipende largamente dal set di cluster iniziale e si può ottenere anche una soluzione ben peggiore dell'ottimo globale. Dato che l'algoritmo è estremamente veloce,

è possibile applicarlo più volte e fra le scegliere la soluzione più soddisfacente tra quelle prodotte. Un altro svantaggio dell'algoritmo è che esso richiede di scegliere il numero di cluster, k, da trovare. Se i dati non sono naturalmente partizionati si ottengono risultati strani. Inoltre l'algoritmo funziona bene solo quando sono individuabili cluster sferici nei dati.

A.4 Edge Histogram Descriptor

Durante la progettazione del sistema per il suggerimento dei concetti agli shot dei video, è stato utilizzato anche il descrittore Edge Histogram Descriptor, appartenente allo standard MPEG-7. I descrittori dello standard MPEG-7 si dividono in diverse classi:

- Descrittori di colore;
- Descrittori di tessitura;
- Descrittori di forma;
- Descrittori di moto;
- Localizzazione;
- Altri descrittori.

Il descrittore utilizzato appartiene alla classe dei descrittori di tessitura ed è l'Edge Histogram Descriptor. Esso rappresenta la distribuzione spaziale di 5 tipi di edge, di cui 4 sono edge direzionali e uno è non direzionale. Dato che gli edge giocano un ruolo molto importante nella modalità di percezione delle immagini, essi possono essere utilizzati per recuperare immagini con significato semantico simile. Tutto ciò è vero in modo particolare per le immagini naturali

con una distribuzione non uniforme degli edge. Si possono ottenere risultati molto incoraggianti nel contesto del *videomatching* combinando questo descrittore con altri descrittori appartenenti alla classe del colore. Per l'estrazione del descrittore, un frame viene suddiviso in una griglia 4x4 e per ogni blocco viene calcolato un edge histogram, valutando la forza dei 5 tipi di edge e considerando quelli che superano una soglia prefissata. I valori dei bin vengono normalizzati nell'intervallo [0,1]. Infine una quantizzazione non lineare dei valori dei bin conduce ad una rappresentazione di 3 bit/bin. In questo modo il descrittore consiste di 80 bin (16 blocchi e 5 bin per blocco). Il modo più semplice per rappresentare gli 80 bin si ottiene utilizzando un intero per i 3 bit di ogni bin. In questo modo si ottiene una sequenza di 80 interi, ciascuno dei quali può assumere un valore tra 0 e 7.

A.5 Rappresentazione di colore HSV

HSV è l'acronimo di **Hue Saturation Value** (tonalità, saturazione e valore). Per saturazione si intende l'intensità e la purezza del colore, mentre la luminosità (valore) è un'indicazione della sua brillantezza. Ovviamente la tonalità indica il colore stesso. Il modello di colore HSV può essere concettualmente pensato come un cono invertito di colori con un punto nero nelle parte della punta e i colori completamente saturati attorno ad una circonferenza nella parte alta del cono. Questa situazione può essere visualizzata in Figura A.8.

La rappresentazione conica, sulla destra, è utile per visualizzare l'intero spazio HSV come un singolo oggetto. La parte sinistra di Figura A.8 corrisponde a mezza faccia del cono rappresentato nella parte destra della figura. La tonalità (H) viene misurata come un angolo intorno all'asse verticale, con il rosso a 0 gradi, il verde a 120 e il blu a 240. L'altezza del cono rappresenta il valore (V)

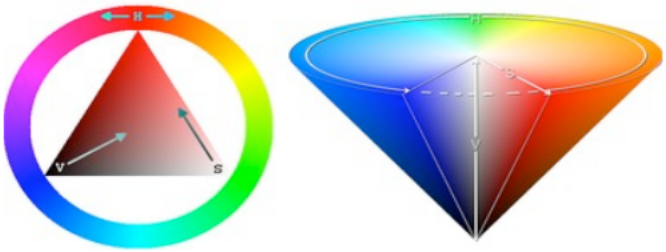

Figura A.8: Rappresentazione dello spazio di colore HSV

con lo zero che rappresenta il nero e l'uno il bianco. La saturazione (S) invece va da zero, sull'asse del cono, a uno sulla superficie del cono.

A.6 Correlogramma di colore

Il correlogramma di colore (color correlogram) [Huang] è una feature estraibile dalle immagini per l'indicizzazione e il confronto tra le stesse. Il punto di forza di questa feature risiede nel fatto che essa tiene in considerazione la correlazione spaziale tra colori; in questo modo vengono tollerate grandi variazioni nella forma e nell'apparenza di un oggetto, causate da modifiche del punto di ripresa o dello zoom della videocamera.

I caratteri maggiormente distintivi di questa feature sono i seguenti:

1. Inclusione della correlazione spaziale tra colori;
2. Può essere utilizzata per descrivere la distribuzione globale di correlazioni spaziali locali tra colori;
3. Semplice da calcolare;
4. Le dimensioni della feature sono contenute.

Il correlogramma di colore di un'immagine è una tabella indicizzata da una coppia di colori, dove il k-esimo ingresso di (i, j) specifica la probabilità di trovare un pixel di colore j a distanza k da un pixel di colore i, all'interno dell'immagine.

La differenza più rilevante rispetto ad un istogrmamma di colore è che l'istogramma cattura solo la distribuzione di colore in un'immagine, senza includere alcuna informazione sulla distribuzione spaziale del colore. Il correlogramma, invece, esprime come la correlazione spaziale tra coppie di colori cambia con la distanza.

A.7 CFP-Tree

In questa sezione viene descritta la struttura del CFP-Tree, la struttura di indicizzazione utilizzata nella parte di programma realizzata per la ricerca della similarità tra video.

A.7.1 Descrizione della struttura

L'acronimo CFP sta per **Condensed Frequent Pattern Tree** e serve per indicare un particolare tipo di albero che consente la memorizzazione di informazioni ripetute contenute in un database di transazioni e la possibilità di creazione di algoritmi molto efficaci per recuperare velocemente tali informazioni. L'estrazione di informazioni frequenti si basa sulla definizione di una *soglia di supporto minimo* che stabilisce il numero minimo di ripetizioni che un pattern deve presentare per essere ritenuto frequente. La strategia è quella di materializzare i pattern frequenti con un supporto minimo sufficientemente basso in modo che la maggior parte, se non tutte, delle richieste degli utenti possano essere esau-

TID	Transactions
1	a, b, c, f, m, p
2	a, d, e, f, g
3	a, b, f, m, n
4	a, c, e, f, m, p
5	d, f, n, p
6	a, c, h, m, p
7	a, d, m, s

Figura A.9: Esempio di database di transazioni

dite interrogando il set di pattern materializzato (che rappresenta l'albero CFP creato).

Per far comprendere al meglio il funzionamento dell'albero CFP si può utilizzare il database di Figura A.9 come esempio.

Supponendo di fissare una soglia di supporto minimo del 40% (quindi si selezioneranno solo i pattern che compaiono almeno 3 volte su 7 transazioni) si ottiene che i pattern frequenti sono quelli di Figura A.10.

Data una soglia di supporto minimo, il set completo di pattern frequenti potrebbe essere indesideratamente grande, specialmente se esistono pattern lunghi. Quindi vengono estratti i soli *pattern frequenti chiusi*. Un pattern frequente è chiuso se nessuno dei suoi *superset* ha il suo stesso supporto. Con superset di un pattern intendiamo gli altri pattern contenenti al loro interno il pattern in questione, ma che abbiano un supporto maggiore. Si ha che il set di pattern frequenti chiusi è la rappresentazione più coincisa dell'intero set di pattern frequenti senza perdita di informazioni e tale set può essere significativamente più piccolo del set completo di pattern frequenti, come possiamo osservare dalla

All Freqeunt Patterns
c:3, d:3, p:4, f:5, m:5, a:6
cp:3, cm:3, ca:3, pf:3, pm:3, pa:3, fm:3, fa:4, ma:5
cpm:3, cpa:3, cma:3, pma:3, fma: 3
cpma: 3

Figura A.10: Pattern frequenti del database di Figura A.9

Frequent Closed Patterns
d: 3, p: 4, f: 5, a: 6
pf: 3, fa: 4, ma: 5
fma: 3
cpma:3

Figura A.11: Insieme dei pattern frequenti chiusi

Figura A.11.

Detto ciò, solo i pattern frequenti chiusi sono quelli da includere nella struttura dell'albero CFP. La struttura di un albero CFP è detta condensata, non solo perchè esso memorizza i pattern chiusi più frequenti, ma anche perchè tali pattern condividono la memorizzazione dei loro prefissi nell'albero. C'è da dire che per ogni ingresso E in un nodo, supponiamo che il pattern che rappresenti il cammino dalla radice ad E sia p, vengono mantenute quattro parti di informazione:

1. L'ultimo item di p indicato con $E.item$;

2. Il supporto di p cioè il numero di ripetizioni del nodo all'interno delle transazioni nel DB, rappresentato con $E.support$;

3. Un puntatore alla radice del sottoalbero CFP che memorizza tutti i pattern che hanno p come prefisso che viene indicato con $E.child$;

4. Una bitmap hash per tenere memorizzati gli indici degli item presenti all'interno del sottoalbero che nasce da E, indicato con $E.bitmap$.

Un albero CFP gode anche di alcune proprietà rilevanti che lo rendono molto utile ed efficace per rispondere a due tipi particolari di query:

1. Query con vincoli di supporto minimo con le quali si vogliono estrarre tutti i pattern che siano ripetuti almeno un numero di volte maggiore al supporto minimo fornito nella query. Un esempio è una query del tipo: *Estrazione di tutti i pattern presenti almeno il 40 per cento delle volte.*

2. Query con vincoli sulla presenza di item con le quali si vogliono estrarre tutti i pattern contenenti degli specificati item. Un esempio è una query del tipo: *Estrazione di tutti i pattern contenenti gli item c e d.*

Le proprietà dell'albero CFP sono la **Proprietà Apriori** e la **Proprietà di Contenimento Sinistro**. La Proprietà Apriori stabilisce che per un ingresso E in un nodo, il supporto di ogni sottoalbero puntato da E non può essere superiore del supporto di E stesso. Ciò porta a notevoli vantaggi nell'esecuzione delle query con vincoli di supporto minimo. Infatti se il supporto di un ingresso non soddisfa le regole di supporto minimo, allora sarà inutile cercare all'interno del sottoalbero puntato da tale ingresso perchè all'interno vi saranno solo item con supporto minore o uguale a quello dell'ingresso stesso.

La Proprietà di Contenimento Sinistro stabilisce che per un ingresso E, l'item rappresentato da $E.item$, può apparire solo nel sottoalbero puntato dagli ingressi

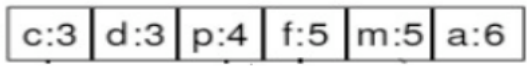

Figura A.12: Primo nodo dell'albero CFP.

prima di E o da E stesso. Questa proprietà può essere usata nelle query con vincoli sulla presenza di item poiché per trovare tutti i pattern contenenti $E.item$, si deve accedere solo ai sottoalberi puntati dagli ingressi precedenti E ed E stesso. Grazie al fatto che gli item all'interno di un nodo sono ordinati per ordine crescente di supporto è semplice verificare i notevoli vantaggi che porta questa proprietà soprattutto per alberi di notevoli dimensioni.

A.7.2 Costruzione dell'albero

L'algoritmo di costruzione dell'albero CFP è presentato in [Jeffrey03]. Dato un database di transazioni ed una soglia di supporto minimo, l'albero CFP viene costruito con due sole scansioni del DB di transazioni. Durante la prima scansione del DB, vengono estratti tutti gli item frequenti (cioè che sono presenti nelle transazioni per un numero di volte almeno uguale al valore della soglia di supporto minimo) e vengono ordinati in base alla loro frequenza, in ordine crescente. A questo punto viene creato un nodo di un albero CFP che contiene tutti gli item frequenti con i rispettivi supporti. Consideriamo $F = i_1, i_2, ..., i_m$ il set di item frequenti che vanno quindi a costituire il primo nodo dell'albero in fase di creazione, osservabile in Figura A.12.

Poi si procede ad una seconda scansione del DB nella quale viene costruito un DB condizionale per ogni item presente all'interno di F. Durante la seconda scansione del DB, gli item non frequenti (cioè presenti un numero di volte minore rispetto al supporto minimo) vengono rimossi da ogni transazione t presente all'interno del DB di partenza e gli item rimanenti vengono ordinati in accor-

TID	Transactions
1	c, p , f, m, a
2	d, f, a
3	f, m, a
4	c, p , f, m, a
5	d, p, f,
6	c, p , m, a
7	d, m, a

Figura A.13: DB di transazioni dopo la seconda scansione

TID	Transactions
1	c, p , f, m, a
2	c, p , f, m, a
3	c, p , m, a

Figura A.14: Database condizionale dell'*item c*. Sono presenti solo le transazioni in cui compare l'*item c* come elemento con supporto minore.

do al loro ordine in F. Una transazione t viene inserita all'interno di un DB condizionale D_j se il primo item di t è ij.

In Figura A.13 viene mostrato il DB di transazioni dopo le modifiche apportate dalla seconda scansione.

In Figura A.14 è presente un esempio di DB condizionale.

A questo punto sono stati costruiti i DB condizionali per tutti gli item del primo nodo; questi contengono le informazioni sufficienti per scavare tutti i pattern frequenti senza accedere più il DB originale. Il passo successivo è l'estrazione sui singoli DB condizionali seguendo lo stesso processo dell'estrazione sul DB di partenza. Una volta terminato il processo di estrazione su un DB condizionale, questo può essere scartato. Le transazioni di questo DB, però, saranno inserite

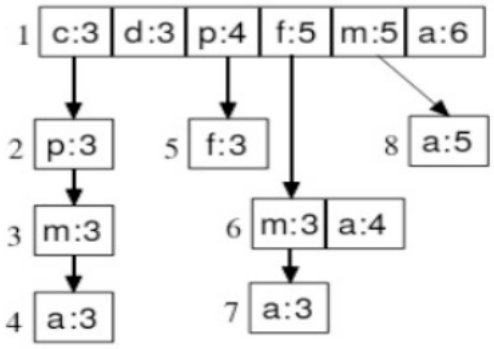

Figura A.15: CFP-Tree

negli altri DB condizionali in base alla seguente regola: una volta terminato il processo di estrazione ad esempio sul DB D_1, le transazioni di questo DB verranno inserite in un altro DB in base all'item successivo all'item i_1 di ogni transazione. Se per esempio in una transazione del D_1 l'item successivo a i_1 è i_4, allora tale transazione verrà inserita all'interno di D_4, chiaramente da i_4 in poi, cioè dall'item successivo ad i_1. Questo passo viene chiamato *pushright*. In Figura A.15 è mostrato l'albero CFP creato con la procedura appena descritta.

Per eliminare i pattern ridondanti durante il processo di estrazione, si devono seguire i seguenti due lemma.

LEMMA 1: *Nell'algoritmo un pattern p è chiuso se e solo se si rispettano le seguenti due condizioni:*

1. *Non ci sono pattern precedentemente estratti che siano superset di p ed abbiano lo stesso supporto di p.*

2. *Tutti gli item in Dp hanno un supporto minore di p.*

LEMMA 2: *Nell'algoritmo un pattern p non è chiuso perchè la condizione 1 nel Lemma 1 non è rispettata, allora nessun pattern estratto da D_p può essere chiuso.*

Basate sul Lemma 1 ci sono due condizioni di potatura per un pattern ridondante p:

1. Esaminare, qualora esista, un pattern precedentemente estratto q, che sia un superset di p ed abbia lo stesso supporto di q. Questa prova può essere fatta prima di estrarre un pattern da un DB condizionale. Se tale q esiste, allora non c'è bisogno di eseguire estrazioni su D_p basate sul lemma 2. Quindi l'identificazione di un pattern ridondante, non solo riduce le dimensioni dell'albero, ma evita anche i costi di estrazione non necessari.

2. Provare se esiste un item i che appaia in ogni transazione di D_p. Se tale i esiste non si devono considerare i pattern non contenenti i con l'estrazione da D_p.

Per incorporare queste due tecniche di potatura, gli item con lo stesso supporto di p sono rimossi da F prima che il nuovo nodo dell'albero CFP venga creato. Per ognuno di questi item i, viene creato un nuovo nodo che contiene solo i stesso. Il nodo E o il più recente creato puntano al nodo contenente i.

Bibliografia

[Arthur06] D. Arthur, S. Vassilvitskii. (2006). How Slow is the k-means Method?. In *Proceedings of the 2006 Symposium on Computational Geometry (SoCG).*

[Basharat08] A. Basharat, Y. Zhai, M. Shah. Content based video matching using spatiotemporal volumes. In *Computer Vision and Image Understanding 110 (2008) 360-377.*

[BBC] BBC motion gallery. URL http://www.bbcmotiongallery.com/Customer/RoyaltyFree.aspx.

[Brown02] M. Brown, D. Lowe. Invariant features from interest point groups. In: BMVC (2002).

[Darrell06] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. Technical report, MIT, 2006.

[DBMul09] A. Ferri, S. Smeraldo. Nearest Neighbours Indexing with Modified CFPTree.*Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze, Facoltà di Ingegneria, Elaborato per il corso di Database Multimediali , A. A. 2008/2009.*

[Deng08] Z.-P. Deng, K.-B. Jia, 2008. A Video Similarity Matching Algorithm supporting for Different Time Scales. In *Eight International Conference on Intelligence Systems Design and Applications*, pp. 570-574.

[Edelman97] S. Edelman, N. Intrator, and T. Poggio. 1997. Complex cells and object recognition. Unpublished manuscript: http://kybele.psych.cornell.edu/edelman/archive.html.

[GoogleVideo] Google video. URL http://video.google.com/.

[Grauman06] K. Grauman and T. Darrell. Approximate correspondences in high dimensions. In *NIPS '06, 2006.*

[Harris88] C. Harris, and M. Stephens, 1988. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, Manchester, UK, pp 147-151.

[Huang] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, R. Zabih. Image Indexing Using Color Correlograms.

[Jeffrey03] W. L. Jeffrey, X. Y. G. Liu, H. Lu, The Hong Kong Univ. of Science and Technology, The Chinese University of Hong Kong, On computing, storing and querying frequent patterns, 2003.

[Jiang97] J. J. Jiang, and D. W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of ROCLING X*, 1997.

[Jiang08] Y. G. Jiang, C. W. Ngo, Bag-of-Visual-Words Expansion Using Visual Relatedness for Video Indexing. In *SIGIR'08, July 20-24, 2008, Singapore.*

[Jiang09] W. Jiang, C. Cotton, S.-F. Chang, D. Ellis, and A. Loui. Short-term audio-visual atoms for generic video concept classification. In

MM'09: Proceedings of the seventeen ACM international conference on Multimedia, 2009.

[Kennedy09] L. S. Kennedy, M. Slaney, and K. Weindberger. Reliable tags using image similarity. In *Proc. of ACM MM Workshop on Web-Scale Multimedia Corpus*, Beijing, China, 2009.

[Li09] X. Li, C. G. M. Snoek, and M. Worring. Learning social tag relevance by neighbor voting. *IEEE Transactions on Multimedia*, 11(7):1310-1322, 2009

[Liu09] D. Liu, X.-S. Hua, L. Yang, M. Wang, and H.-J. Zhang. Tag Ranking. In *Proc. of International World Wide Web Conference (WWW)*, 2009.

[LIACS10] B. Thomee, E. M. Bakker, M. S. Lew (LIACS Media Lab, Leiden University, Netherlands). TOP-SURF: a visual words toolkit. In *ACM Multimedia 2010,* October 25-29, 2010, Firenze, Italy.

[LSCOM] L. Kennedy. Revision of LSCOM Event/Activity Annotations, DTO Challenge Workshop on Large Scale Concept Ontology for Multimedia. Technical report, Columbia University, December 2006.

[Lowe99] D. G. Lowe, 1999. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, Corfu, Greece, pp. 1150-1157.

[Lowe04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[MICC09] L. Ballan, M. Bertini, A. Del Bimbo, and G. Serra (Media Integration and Communication Center, Università degli Studi di Firenze). Video Event classification using string kernel. *Springer Science + Business Media, LLC 2009.*

[MICC10] L. Ballan, M. Bertini, A. Del Bimbo, M. Meoni, and G. Serra (Media Integration and Communication Center, Università degli Studi di Firenze). Tag suggestion and localization in user-generated videos based on social knowledge. *In WSM'10,*October 25-29, 2010, Firenze, Italy.

[Needleman70] S. B. Needleman, C. D. Wunsch, 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453.

[Nencioni10] Estensione e filtraggio per annotazione di video basata su Semantic Web, *Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze, Facoltà di Ingegneria, Tesi di Laurea Specialistica in Ingegneria Informatica, A. A. 2009/2010.*

[Nister06] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *In CVPR'06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[Philbin07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, 2007. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1-8.

[Plummer86] L. Lov'asz, M. D. Plummer, Matching Theory, North-Holland,1986.

[Rubner00] Y. Rubner, C. Tomasi, L. Guibas, The earth mover's distance as a metric for image retrieval. In *International Journal of Computer Vision 40 (2) (2000) 99-121.*

[Salton83] G. Salton, and G. McGill, 1983. Introduction to modern information retrieval. McGraw-Hill.

[ShotTagger] ShotTagger: Locating Tags in Video. *In MM'10,* October 19-24, 2010, Italy.

[SURF] H. Bay, A. Ess, T.Tuytelaars, L. Van Gool, 2008. Speeded-up-robust features (SURF). *Computer Vision and Image Understanding*, 110(3), 346-359.

[TRECVID] Trec video retrieval track (2005). URL http://www-nlpir.nist.gov/projects/trecvid/.

[vanZwol08] B. Sigubjornsson, R. van Zwol. Flickr tag reccomendation based on collective knowledge. In *Proc. WWW*, 2008, pp 327-336.

[Wang08] F. Wang, Y.-G. Jiang, C.-W. Ngo. Video Event Detection Using Motion Relativity and Visual Relatedness. In *MM'08*, October 26-31, 2008, Vancouver, British Columbia, Canada.

[Xu07] D. Xu and Shih-Fu Chang, Visual Event Recognition in News Video using Kernel Methods with Multi-Level Temporal Alignment", In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007.

[Yeh] T. Yeh, J. Lee, and T. Darrell (CSAIL, Cambridge, MA, USA). Adaptive Vocabulary Forests for Dynamic Indexing and Category Learning.