Università degli studi di Catania

Facoltà di Informatica

Relazione progetto Computer Vision

Controllo violazione copyright video, tramite la generazione di SIFT e SURF signature

prof. Sebastiano Battiato

Antonio Stivala W8200132
Andrea Reale W82000133

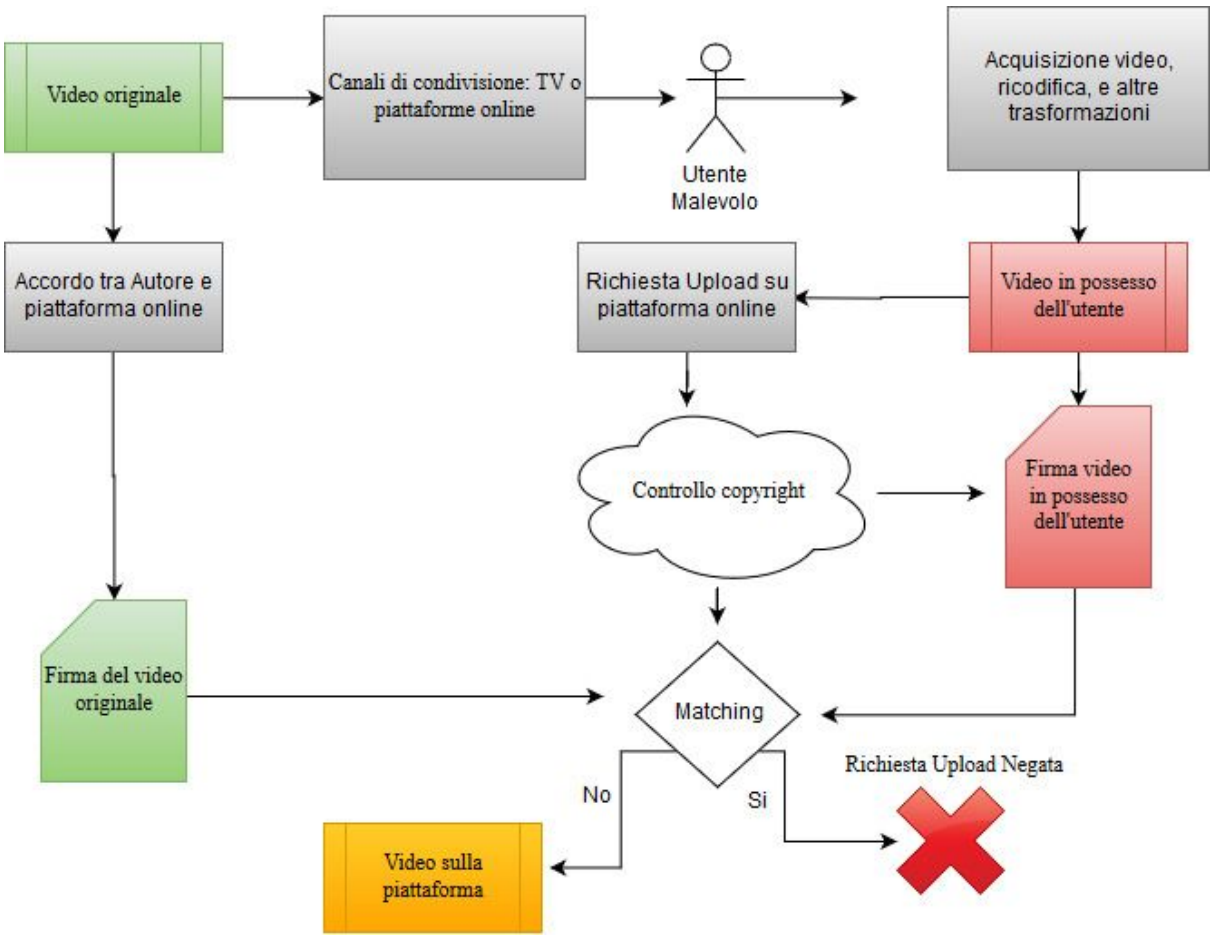
Catania, Marzo 2019

1. Introduzione

L'obiettivo del software sviluppato è di preservare il copyright di video caricati in un'ipotetica piattaforma. Quando l'utente richiede l'upload di un video, il gestore dovrebbe effettuare dei controlli sul contenuto in possesso dall'utente. Per preservare i diritti d'autore di fronte al numero elevato di contenuti multimediali nel web, è necessaria una collaborazione tra autore e piattaforma di video sharing. L'autore, può fornire un supporto attivo o passivo. Il supporto attivo consiste nell'inserire delle informazioni nascoste all'interno del video originale (watermarking e/o altri metodi steganografici), in modo da risalire all'autore. In questo caso, il proprietario deve fornire anche gli strumenti di estrazione della firma nascosta che verranno utilizzati dalla piattaforma per preservare i diritti d'autore. Il supporto passivo invece consiste nel delegare il gestore della piattaforma all'estrazione della firma, l'autore deve limitarsi a fornire il video originale.
Il software sviluppato si posiziona in quest'ultimo caso, l'autore fornisce il video originale dal quale verrà calcolata la firma tramite due tipi di keypoint: surf e sift.
Con il medesimo algoritmo, verrà generata la firma del video che l'utente vuole caricare sulla piattaforma, che da ora in poi chiameremo video query.
A questo punto bisognerà confrontare le firme del video originale e del video query.La metodologia di confronto è il brute force, ogni frame del video originale verrà confrontato con ogni frame del video query. Se un frame combacia con un altro, si passerà al successivo frame originale, evitando di confrontarli tutti migliorando il tempo computazionale, a discapito però della perdita dell'informazione del migliore match.
Per ridurre la complessità computazionale, si sono adottate varie tecniche: confronto locale, maschera e skip tramite voting.
Nel confronto locale, in caso di match, il frame originale successivo, verrà confrontato con un intorno locale a partire dall'indice dell'ultimo match. In questo modo se il match corrisponde con un allineamento temporale del video, i frame successivi combaceranno con i frame locali, risparmiando tempo computazionale.
Per limitare il numero di confronti totale tra descrittori, si è pensato di limitare il numero di keypoint. Per filtrare il numero totale di descrittori per ogni frame, si applica una maschera per si posiziona nella parte centrale del frame. La percentuale di frame che la maschera seleziona è di default il 30%, ma risulta essere un parametro modificabile.
Selezionando quindi solo una porzione di frame, si limita il numero di descrittori trovati su quest'ultimo.
La maschera a sua volta è stata divisa in 3 sezioni, in ognuna di queste verranno calcolati i descrittori. Per limitare il numero di confronti totali tra frame, si è implementato un sistema di voti per skippare frame che consideriamo "troppo diversi". Se il rapporto tra il numero di descrittori di ogni sezione supera un certo intorno, consideriamo i frame differenti, e non si procede al confronto tra keypoint.

2. Problema

Il problema consiste nell'identificare se un video o parti di esso, durante il caricamento(upload) in un'ipotetica piattaforma online, violi il diritto d'autore di altri video che considereremo originali. La pipeline è mostrata nella figura sottostante.



Come si vede dalla figura, una volta che il video originale è trasmesso in broadcast TV (o tramite canali alternativi, per esempio streaming on line o on-demand), un utente malevolo, con semplici ed economici dispositivi di acquisizione PVR (Personal Video Recorder), potrebbe entrare in possesso del video in questione.

Già in questa prima fase, il video acquisito, potrebbe risultare diverso dall'originale, in quanto i vari dispositivi PVR in mercato potrebbero applicare delle ricodifiche in formati proprietari o semplicemente diversi dall'originale.

Una volta acquisito il video, l'utente prima di effettuare l'upload sulla piattaforma, potrebbe apportare ulteriori modifiche allo stesso. Solitamente un'operazione comune per ridurre le dimensioni e velocizzare i tempi di upload è la compressione video.

Anche in questa situazione, dunque, il video originale subisce delle ulteriori alterazioni.

L'ultima fase del ciclo di vita del video è il caricamento effettivo nella piattaforma, ed è proprio in questa fase che si posiziona il software sviluppato. Durante l'upload infatti, il software tramite dei descrittori (SIFT e SURF), genera una firma del video che l'utente vuole caricare.

A questo punto è necessario **un accordo tra l'autore del video originale e la piattaforma online**, in quanto è necessario generare una firma dal video originale per poterla confrontare con i video sulla piattaforma. In questo modo potrebbe essere possibile stabilire se un video è coperto da copyright e quindi declinare la richiesta di upload.

Non sono state considerate possibili operazioni di editing malevolo atte ad occultare esplicitamente parte del contenuto o aggirare i software per il copyright detection. Nonostante ciò è evidente come durante l'intero ciclo di vita, il video venga trasformato, nel migliore dei casi almeno due volte. Ciò comporta che una qualsiasi tecnica di analisi, volta

ad identificare in maniera univoca un determinato contenuto video, risulta essere complessa e non del tutto esatta.

3. Proposta

Un approccio manuale al problema su tutti i video caricati su un'ipotetica piattaforma risulta essere difficile da realizzare. I numeri dei video caricati è sicuramente elevato, quindi comporterebbe l'utilizzo di un numero di risorse umane non indifferente. Oltremodo, ogni operatore dovrebbe conoscere tutti i video coperti dai diritti d'autore. Per queste motivazioni l'unico approccio valido è tramite tecniche automatiche.
Le tecniche automatiche si suddividono in due macro categorie, **attive e passive**. Le tecniche attive ad esempio il watermarking prevedono l'inserimento di informazione all'interno del file multimediale, in modo tale da rilevare e/o estrapolare indicazioni sull'origine e la provenienza del file. Tali tecniche infatti potrebbero essere resistenti alle alterazioni che un video subisce durante la pipeline in figura 1, oltre che essere robuste anche ad alcune tipologie di attacchi malevoli. Tuttavia l'utilizzo delle tecniche attive deve essere delegato al detentore dei diritti d'autore. Egli infatti deve costruire appositi strumenti di protezione da "nascondere" impercettibilmente nei video che vengono poi trasmessi ai propri utenti. Dovrà inoltre fornire alle piattaforme online i meccanismi per l'estrazione delle informazioni segrete.
Quindi si è convenuto utilizzare delle tecniche passive, le quali riteniamo siano l'approccio più utilizzato dalle piattaforme online di video sharing. Le tecniche passive si basano sull'estrazione di punti caratteristici dell'immagine chiamati **features.**
Per l'estrazione delle features si sono utilizzati e confrontati due tipi di algoritmi **SIFT** e **SURF.**
I keypoint SIFT risultano essere **invarianti per scala**, in quanto cercano i punti chiavi in differenti scale.Oltre ad essere invarianti **per rotazione e traslazione e luminanza**.
I keypoint SURF risultano invece essere una versione "approssimata" delle SIFT, meno robusta, ma più veloce in termini di complessità computazionale.
I risultati verranno mostrati con una serie di grafici ogni tipo di descrittore utilizzato (sift e/o surf).

4. Descrizione Software

Il software implementa:

- Creazione firma
- Analisi video query
- Plot dei risultati

4.1 Maschera

Sin da subito si è posti il problema computazionale. Ipotizziamo di avere un video di circa 20 minuti, registrato a 30 frame per secondo.

30 x 60 = 1800 frame al minuto

1800 x 20 = 36000 frame per l'intero video

Ogni frame prevede un certo un numero di SIFT/ SURF. Focalizziamoci su un singolo descrittore ed ipotizziamo che mediamente troviamo 100 SIFT per frame ottenendo:

36000x 100= 3.600.000 SIFT

Ogni keypoint è un array di 128 elementi il quale dovrà essere confrontato con un altro array di 128 elementi con una certa misura di distanza, nel nostro caso **L2 NORM.**

Si è posto quindi il problema di dover limitare in qualche modo il numero di SIFT/ SURF estrapolate da ogni frame. Si è quindi scelto di estrapolare i keypoint solo da sezioni ritenute più importanti di frame, applicando **una maschera.**

Si è scelto di prendere solo una percentuale dell'immagine (default 30%) rettangolare centrale, la quale si ritiene fondamentale e robusta a manipolazioni come specchiatura o cambio di aspect ratio. Inoltre questa maschera è a sua volta divisa in 3 porzioni, le quali saranno utili per effettuare uno skip attraverso un sistema di voting, basato sul numero di descrittori trovati su ogni porzione.

4.2 Generazione firma

L'insieme dei descrittori di ogni porzione di frame dell'intero video costituisce la firma del file.**Tuttavia si è reputato superfluo inserire ogni singolo fotogramma nella firma, in quanto molti fotogrammi temporalmente vicini tra loro risultano essere molto simili**. A tal proposito si è deciso di parametrizzare il numero di frame al secondo da estrapolare dal video. Per i nostri test si è fissato un frame al secondo.

Il metodo che si occupa di generare la firma del video è:

getRenctangleSignature*(pathInput, fileName, nFrame, flag, mode = 0, percentualeImmagine = 30)*

- ***pathInput:*** *percorso del file video in input*
- ***fileName:*** *nome del file Firma che verrà generato*
- ***nFrame:*** *numero di frame al secondo estratti default =1*
- ***flag:*** *False: carica file firma True: genera file firma*
- ***mode:***
    - *0 : descrittore SIFT*
    - *1 : descrittore SURF*
- ***percentualeImmagine:*** *percentuale dell'immagine da utilizzare come maschera*

Il metodo apre il video, e per ogni frame al secondo scelto dall'utente calcola le SIFT o le SURF sulla porzione di frame, caricando il descrittore (array di 128 elementi) su un file.

Si è utilizzata la libreria *pickle,* per poter scrivere e leggere oggetti, (nel nostro caso array) dal file.

4.3 Analisi video query e matching

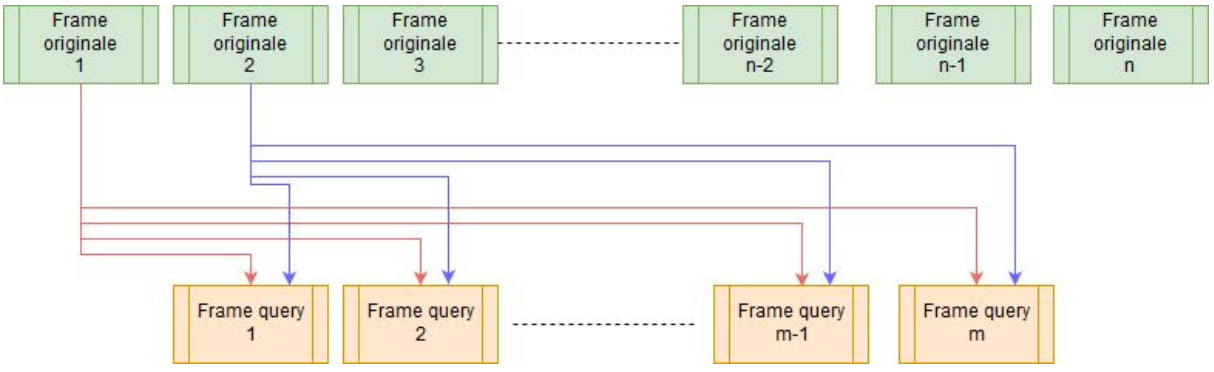
Definiamo “video query” il video che dev’essere analizzato per controllare se viola il copyright. In altre parole il video che l’ipotetico utente vuole caricare sulla nostra ipotetica piattaforma video online.
Come mostrato nella pipeline in figura 1, si ipotizza che il proprietario del video fornisca il video originale su cui poter calcolare la firma che verrà memorizzata nei database della piattaforma. A questo punto il video query che l’utente vuole caricare nella piattaforma online verrà controllato tramite il matching tra descrittori.
Dati due descrittori riferiti a due **keypoint** (A e B), affermare che i due punti sono simili equivale a calcolare la differenza “numerica” tra i due descrittori che va poi confrontata con un opportuno valore di soglia.

1. se distanza(A - B) < T i due descrittori sono simili;
2. se distanza(A - B) > T i due descrittori sono diversi

La metrica di distanza calcolata tra descrittori è la **L2 NORM.** Il valore di T invece viene fissato sulla base di un dataset di riferimento su cui viene calcolato un valore ottimale che minimizza gli errori.
In mancanza di un dataset di riferimento chiuso e limitato non si è potuta individuare una soglia ottimale per il matching tra descrittori.
In base ai test effettuati si sono stabilite delle soglie “percettivamente” discriminanti.
Fissate le soglie, che stabiliscono quando due descrittori sono simili è stato necessario identificare un ulteriore soglia che stabilirà quando **due frame sono simili.** In particolare stabiliamo il numero **minimo di descrittori che superano la soglia** affinchè due frame possano considerarsi simili.
Riassumendo se un certo numero di descrittori risultano essere simili allora i due frame in analisi verranno considerati come match.

Confronto brute force

L’approccio base del software è un approccio **brute force,** ovvero ogni frame del video originale, verrà confrontato con ogni frame del video query. Tuttavia questo accade nel peggiore dei casi ovvero quando i frame non hanno corrispondenza.

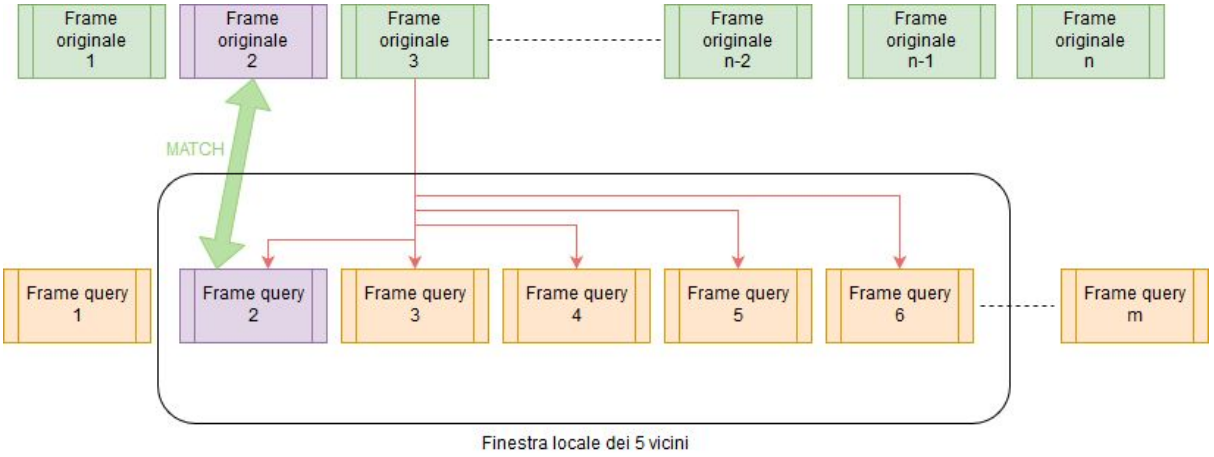


In caso di match, il software passa al frame originale successivo, senza considerare i restanti. **Questo comporta che un frame potrebbe non combaciare con il corrispettivo**

frame del video query(ipotizzando che il video query lo contenga), ma con il primo frame che gli permetta di soddisfare le soglie scelte. Questa scelta ci permette di risparmiare tempo di calcolo, perdendo però l'informazione del **migliore match**.

Confronto intorno locale

Per evitare quindi il confronto brute force, quando un frame originale combacia con un frame query, **il successivo frame verrà confrontato con un intorno locale** (i 5 frame successivi) a partire dall'indice del match. Questo permette di evitare che il frame originale successivo, ricominci il confronto dall'inizio. Questa implementazione risulta essere efficiente in caso di **allineamento temporale.**



Nell'esempio in figura, il frame originale 2 combacia con il frame query 2. L'algoritmo quindi passa al successivo frame originale 3, essendoci un match precedente, conserva l'indice e ricomincia il confronto con il frame query 2. Se il match corrisponde con **un allineamento temporale** probabilmente il frame originale 3 combacerà con uno dei frame all'interno della finestra locale.In questo caso la finestra temporale si sposterà in avanti. Nel caso in cui il frame originale 3 non dovesse combaciare con nessun frame della finestra locale, l'algoritmo ripartirà dall'inizio effettuando un confronto brute force.

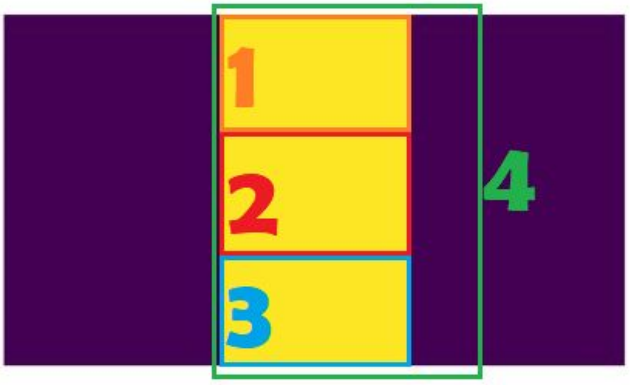
Filter Lowe

Le SIFT e le SURF soffrono di un numero elevato di Falsi Positivi, per questo motivo è consigliato in letteratura l'applicazione del **filtro Lowe**.
Preso un descrittore d, il suo miglior match d' e il suo secondo miglior match d''. Se i due match sono simili (ovvero il rapporto delle distanze **è superiore** alla soglia 0.8) probabilmente si tratterà di un falso positivo, quindi verrà scartato. L'idea base è che il descrittore d è unico, quindi può essere simile solo ad un unico descrittore d', se è simile anche a d'' verrà scartato perchè perde di unicità

```
def filterLowe2(matches,threshold=0.8):
    out_matches=[]
    for m1,m2 in matches:
        if m1[0]/m2[0]<threshold:
            out_matches.append(m1[0])
    return out_matches
```

Skip per voting

Per rendere la fase di analisi più performante, si è sviluppato un metodo per saltare il confronto con un frame in base al numero di sift/surf trovate. Se due frame hanno un numero di sift/surf troppo differente allora non verranno confrontati.
In particolare è stato implementato un metodo di skip basato sul voting. **Se il rapporto del numero di sift di ogni sezione tra due frame è in un intorno di 0.4 allora il voting per la suddetta sezione sarà positivo.**
Se almeno due sezioni su 4 hanno avuto riscontro positivo si potrà proseguire con il confronto tra descrittori



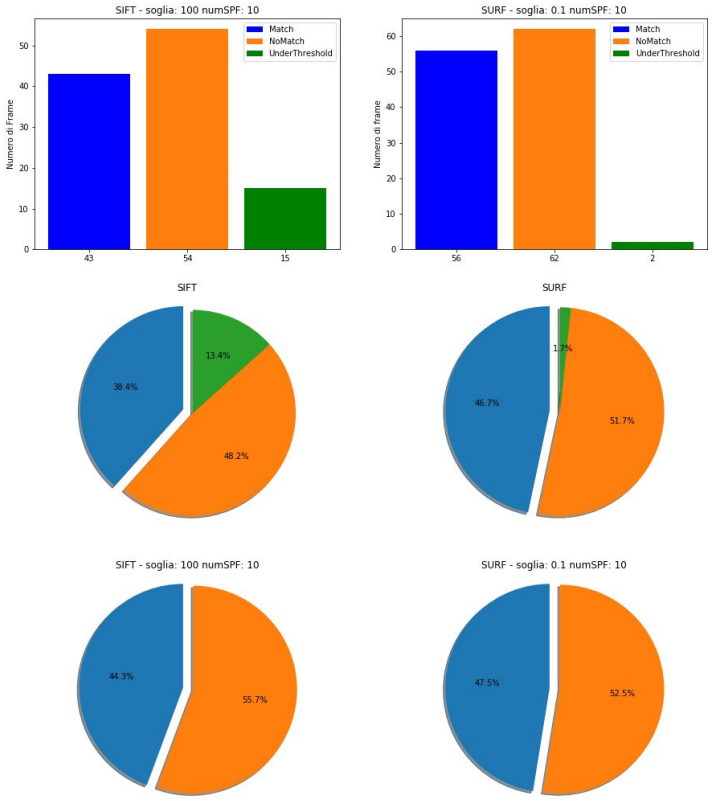
In caso contrario se 3 sezioni su 4 hanno un voting negativo il frame verrà skippato. Questo nei test effettuati ci permette di risparmiare tempo computazionale per frame che teoricamente dovrebbero essere molto differenti tra loro.

4.4 Plot Risultati

Infine il software genera dei file che mostrano i risultati con le soglie scelte comparando i due descrittori.
Nella colonna vengono mostrati i risultati del descrittore SIFT e a destra delle SURF.
Nei grafici a barre mostriamo:

- match: il numero di frame che soddisfano le soglie impostate
- noMatch: il numero di frame che non soddisfano le soglie impostate
- UnderThreshold: skip dovuti al numero di descrittori insufficiente per il match

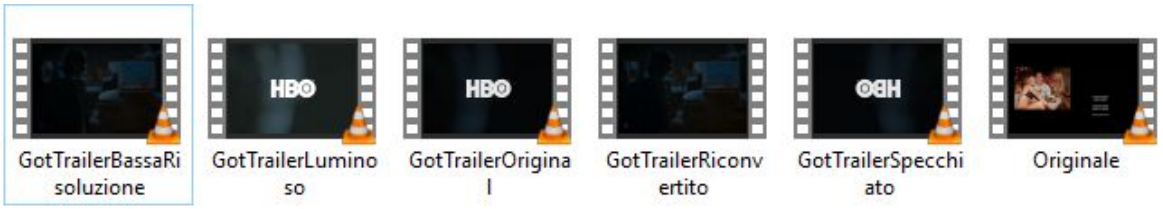
Nei grafici a torta mostriamo invece le percentuali. Nel primo caso nel calcolo della percentuale includiamo il numero di frame underThreshold, verrà escluso nel secondo grafico.



5. Test

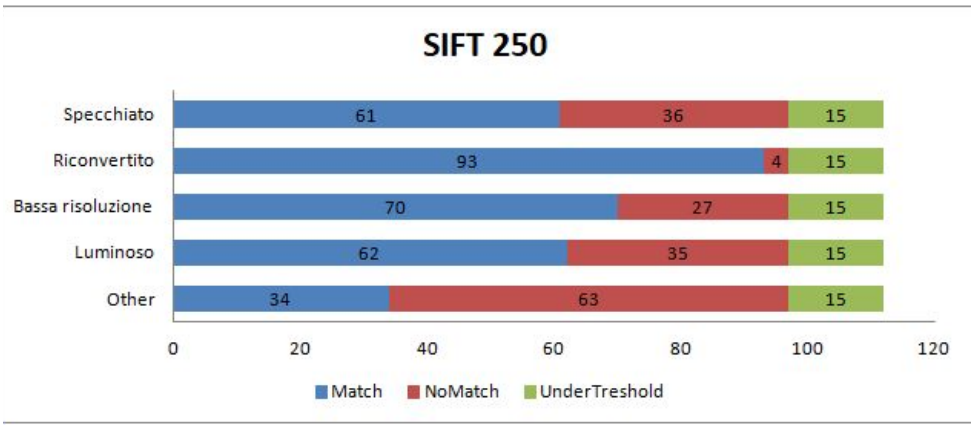
Per la fase di testing, è stato creato un piccolo dataset formato dai seguenti video:

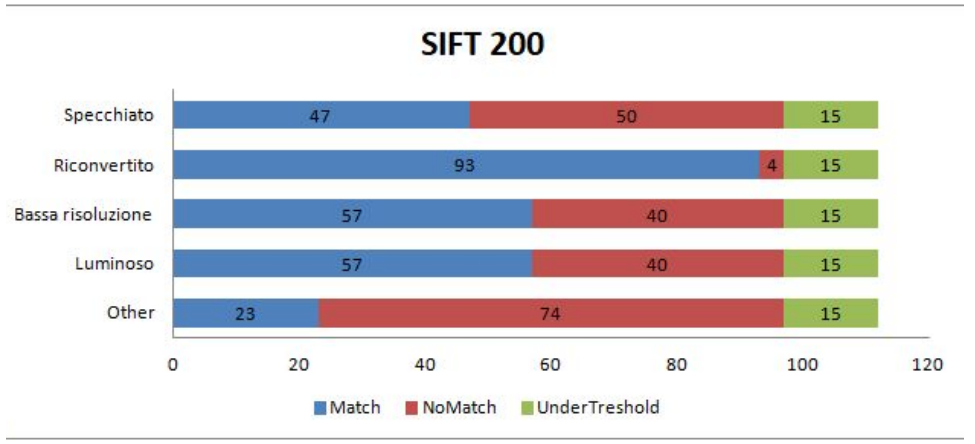
- **Originale:** il [video](#) scaricato da youtube nel formato .MP4
- **Specchiato:** è stato applicato al video originale un mirroring orizzontale
- **Riconvertito:** è stato scaricato il video originale in un formato differente nel nostro caso .WMV
- **Another:** è stato scaricato un video completamente differente dall'originale (trailer notte da leoni)
- **Bassa Risoluzione:** viene convertito il video originale da 1280x720 a 854x480 pixel
- **Luminoso:** viene aumentata la luminosità del video originale diminuendo il contrasto
- **Video del video:** si è catturato parte del filmato(solo 30 secondi) con uno smartphone S7 con aspect ratio 1:1 HD

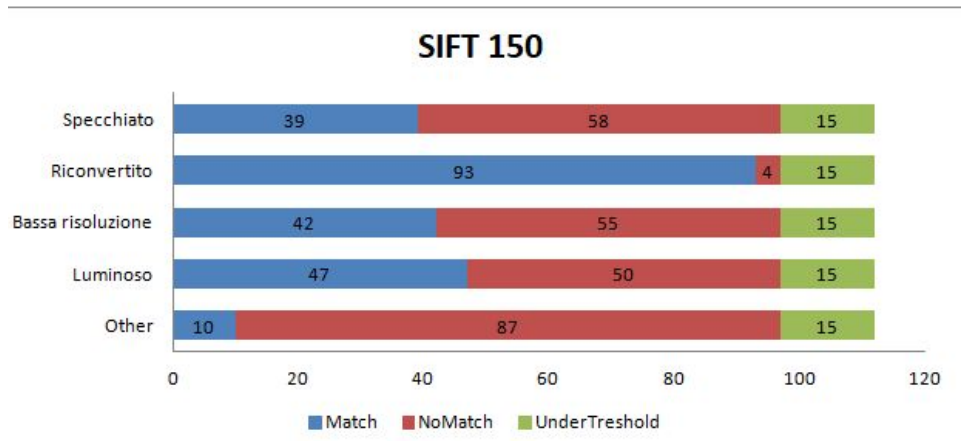


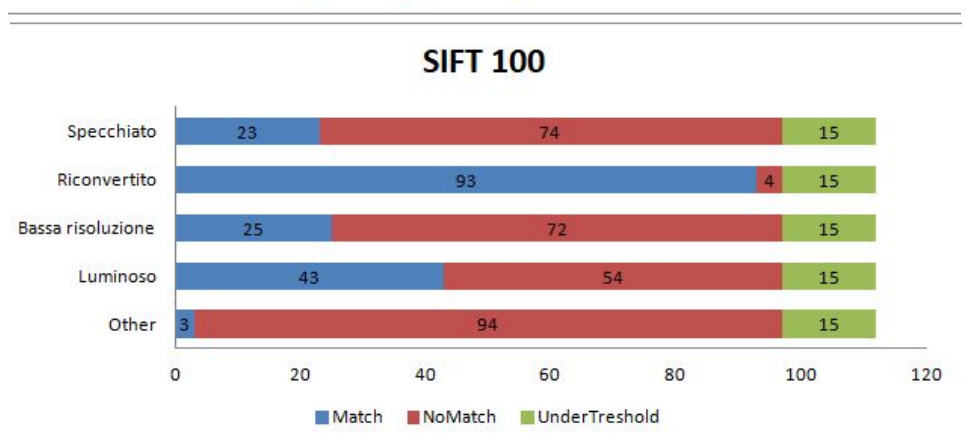
6. Risultati

6.1 SIFT chart

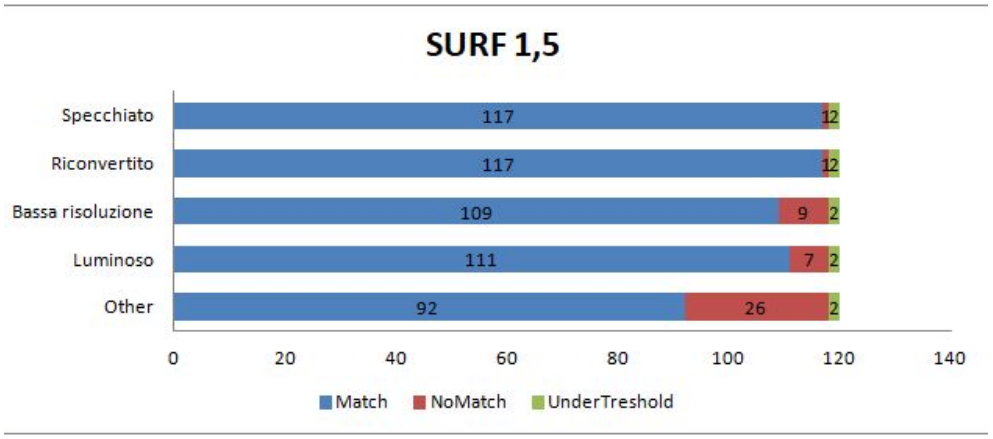


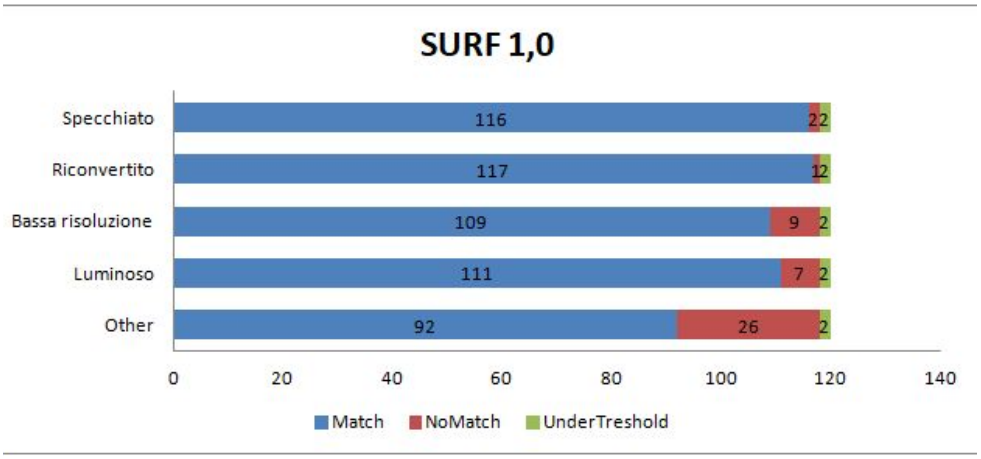


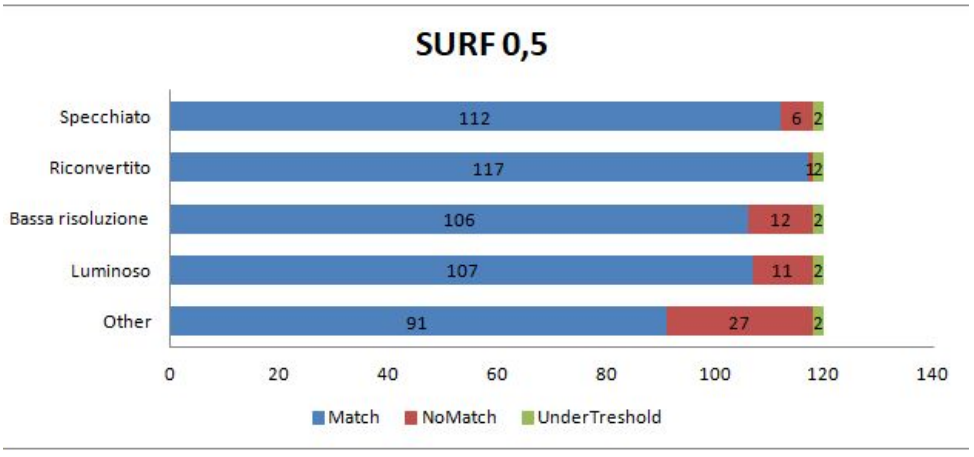


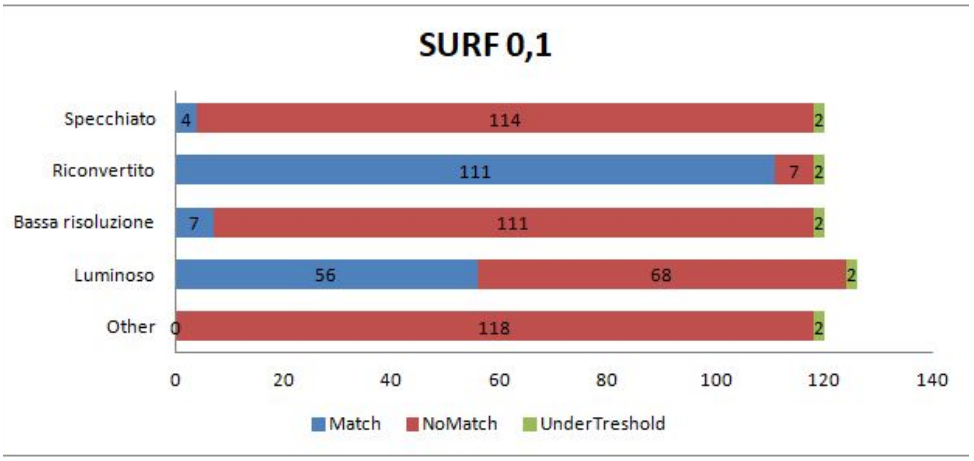


6.2 SURF chart









6.3 Interpretazione chart

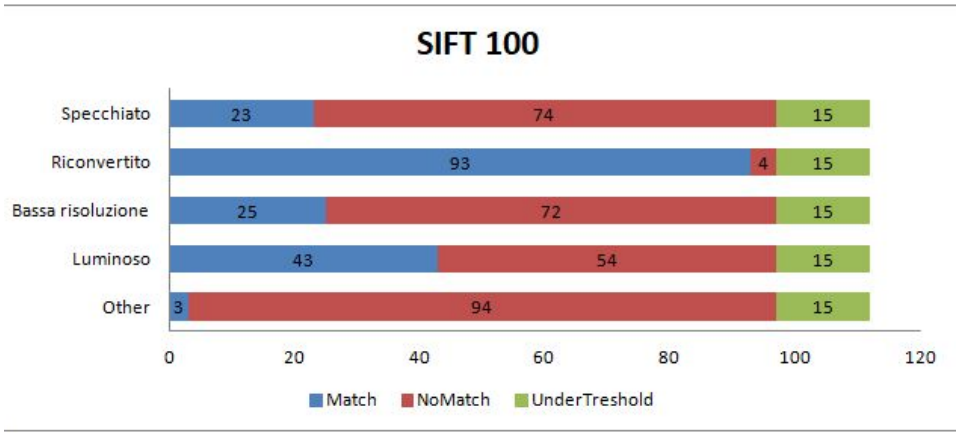
Il decisore automatico, non sempre porta a risultati ottimali, in quanto può generare:

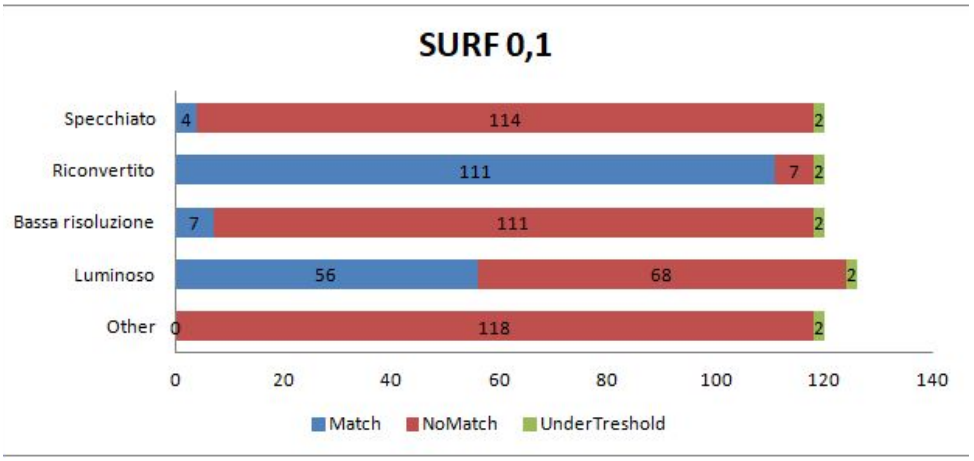
1. Vero Positivo (True Positive o TP), il sistema risponde in maniera affermativa ed effettivamente il video contiene contenuti coperti da copyright;
2. Vero Negativo (True Negative o TN), il sistema risponde in maniera negativa ed effettivamente il video non contiene contenuti coperti da copyright
3. Falso Positivo (False Positive o FP), il sistema risponde in maniera affermativa ma il video non contiene contenuti coperti da copyright;
4. Falso Negativo (False Negative o FN), il sistema risponde in maniera negativa ma il video contiene contenuti coperti da copyright.

Si noti come mentre i casi 1 e 2 sono situazioni di corretto funzionamento del sistema di decisione, i casi 3 e 4 sono situazioni di errato funzionamento.

Osservando i grafici, si potrebbe pensare che le SURF funzionino meglio in quanto presentano il numero maggiore di match. Tuttavia con le soglie scelte, tramite una semplice osservazione dei dati, si evince che le SURF come le SIFT soffrono di molti falsi positivi. Questo si può riscontrare guardando l'asse **Other**. Other rappresenta un video che non ha elementi in comune con il video originale, quindi ci aspettiamo un numero di match pari a zero. Tuttavia impostando le soglie in maniera "lasca" otteniamo un numero di match abbastanza elevato. Questo ci porta a considerare solo quelle soglie che minimizzano il numero di match nella sezione other, in modo da limitare il numero di falsi positivi.

Le soglie con il minor numero di match nella sezione other sono 100 e 0,1 per SIFT e SURF rispettivamente.



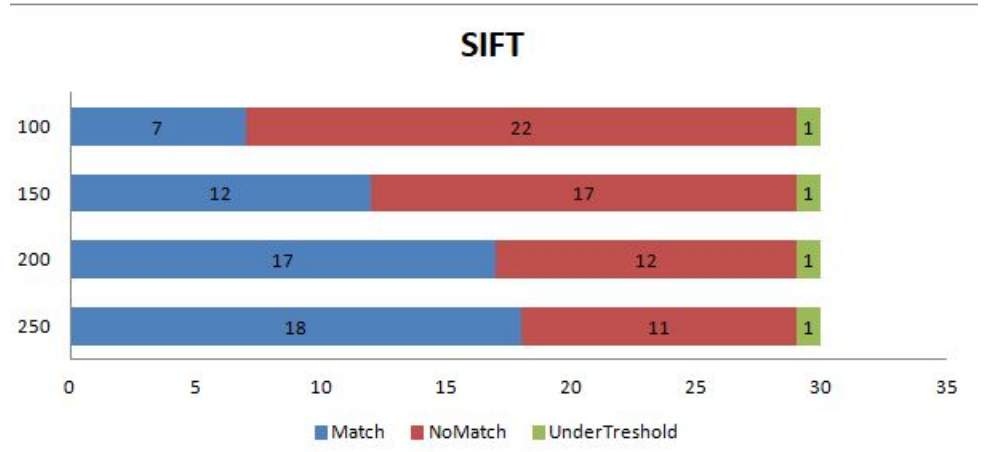


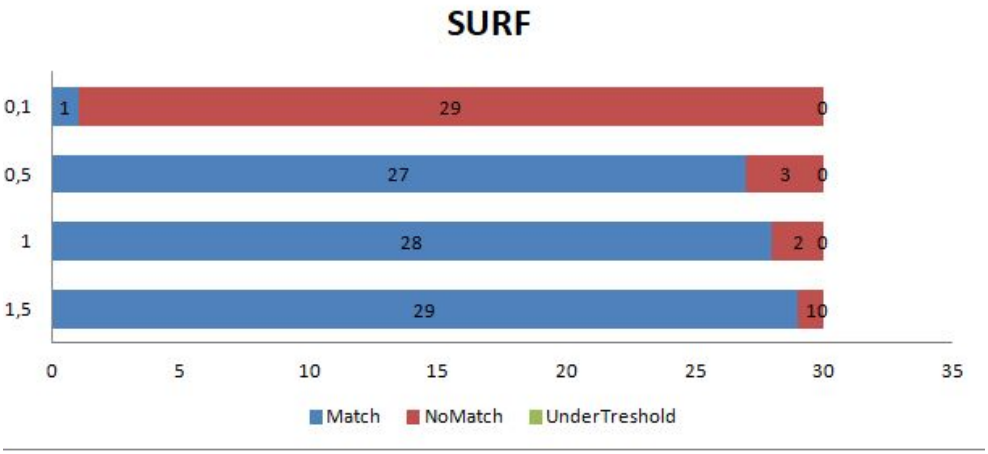
Fissate quindi queste due soglie (100 e 0,1), **possiamo dire che le SIFT sembrano essere più robuste per il mirroring (specchiato), risoluzioni più basse e riconversioni.**

Le SURF invece sembrano essere più robuste solo in situazioni di cambi di luminosità. Il video riconvertito da MP4 a WMV non ha messo in difficoltà nessuno dei due descrittori.

6.4 Video del video

Questo test è stato inserito in una sezione separata in quanto il numero di frame catturati è inferiore rispetto agli altri test effettuati. In questo caso si sono estratti solo 30 frame e si è voluto simulare un classico esempio di pirateria nei cinema. L'utente malevolo, tramite una videocamera (nel nostro caso uno smartphone) e un treppiedi, cattura l'intera sequenza video, per poi successivamente condividerla sul web, violando così i diritti d'autore.





Considerando anche in questo caso le soglie più stringenti che limitano il numero di falsi positivi, le SIFT trovano circa il 23% dei frame come match contro il 3% delle SURF.
Anche in questo caso le SIFT sono più robuste delle SURF.

8. Conclusioni

Il problema di preservare i diritti d'autore, in base al contenuto del video risulta essere un problema non banale. Nella pipeline del ciclo di vita di un video si evidenzia come quest'ultimo venga trasformato almeno due volte. Questo influisce negativamente in qualsiasi tecnica di analisi, volta ad identificare univocamente il contenuto dello stesso.

Dai test effettuati si evince che le SIFT risultano essere più robuste rispetto alle SURF nella maggior parte delle trasformazioni testate.
Tuttavia il numero di test effettuati è insufficiente per trarre delle conclusioni considerevoli, inoltre sono state tralasciate considerazioni sul tempo di calcolo per l'elaborazione dei descrittori. E' noto in letteratura che le SURF riescono ad essere calcolate in tempi molto più brevi rispetto alle SIFT. Tuttavia nel progetto sviluppato non si è posto come obiettivo principale l'efficienza, in quanto queste operazioni possono essere ammortizzate durante la conversione in vari formati che la piattaforma di video sharing effettua prima dell'upload del video. Ad ogni modo le operazioni per il calcolo dei descrittori possono essere parallelizzate (per esempio tramite il calcolo GPU) per ottenere una risposta più immediata. Inoltre dai grafici si evince che il risultato **dipende fortemente dalle soglie che vengono utilizzate, per questo motivo sarebbe opportuno sviluppare un metodo per l'ottimizzazione dei parametri.**