ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECONOLOGIE INFORMATICHE

GeoPhotoHunt

Studio di algoritmi di confronto di immagini per realizzare una caccia al tesoro fotografica

Relazione finale in:
Laboratorio di Basi di Dati

Relatore:

Prof.ssa Alessandra Lumini

Presentata da:

Andrea Zagnoli

II Sessione
Anno accademico: 2013-2014

Sommario

Introduzione

Questo studio si propone di realizzare un'applicazione per dispositivi Android che permetta, per mezzo di un gioco di ruolo strutturato come caccia al tesoro, di visitare in prima persona città d'arte e luoghi turistici. Gli utenti finali, grazie alle funzionalità dell'app stessa, potranno giocare, creare e condividere cacce al tesoro basate sulla ricerca di edifici, monumenti, luoghi di rilevanza artistico-storica o turistica; in particolare al fine di completare ciascuna tappa di una caccia al tesoro il giocatore dovrà scattare una fotografia al monumento o edificio descritto nell'obiettivo della caccia stessa. Il software grazie ai dati rilevati tramite GPS e giroscopio (qualora il dispositivo ne sia dotato) e per mezzo di un algoritmo di instance recognition sarà in grado di affermare se la foto scattata rappresenta la risposta corretta al quesito della tappa.

L'applicazione GeoPhotoHunt rappresenta non solo uno strumento ludico per la visita di città turistiche o più in generale luoghi di interesse, lo studio propone, infatti come suo contributo originale, l'implementazione su piattaforma mobile di un Content Based Image Retrieval System (CBIR) del tutto indipendente da un supporto server. Nello specifico il server dell'applicazione non sarà altro che uno strumento di appoggio con il quale i membri della "community" di GeoPhotoHunt potranno pubblicare le cacce al tesoro da loro create e condividere i punteggi che hanno totalizzato partecipando a una caccia al tesoro. In questo modo quando un utente ha scaricato sul proprio smartphone i dati di una caccia al tesoro potrà iniziare l'avventura anche in assenza di una connessione internet.

L'intero studio è stato suddiviso in più fasi, ognuna di queste corrisponde ad una specifica sezione dell'elaborato che segue. In primo luogo si sono effettuate delle ricerche, soprattutto nel web, con lo scopo di individuare altre applicazioni che implementano l'idea della caccia al tesoro su piattaforma mobile o applicazioni che implementassero algoritmi di instance recognition direttamente su smartphone. In secondo luogo si è ricercato in letteratura quali fossero gli algoritmi di riconoscimento di immagini più largamente diffusi e studiati in modo da avere una panoramica dei metodi da testare per poi fare la scelta dell'algoritmo più adatto al caso di studio. Quindi si è proceduto con lo sviluppo dell'applicazione GeoPhotoHunt stessa, sia per quanto riguarda l'app front-end per dispositivi Android sia la parte back-end server. Infine si è passati ad una fase di test di algoritmi di riconoscimento di immagini in modo di avere una sufficiente quantità di dati sperimentali da permettere di effettuare una scelta dell'algoritmo più adatto al caso di studio. Nello specifico:

- Capitolo 1: Indagine per ricercare altre applicazioni che implementano su dispositivi mobile l'idea de Geocaching o di un sistema di Content Based Image Retrieval.

- Capitolo 2: Analisi degli algoritmi di computer vision presenti in letteratura.
- Capitolo 3: Progettazione dell'applicazione front-end client e back-end server.
- Capitolo 4: Analisi e test su dataset di algoritmi di image recognition con supporto del software Matlab.

Al termine della fase di testing si è deciso di implementare su Android un algoritmo basato sulla distanza tra istogrammi di colore costruiti sulla scala cromatica HSV, questo metodo pur non essendo robusto in presenza di variazioni di luminosità e contrasto, rappresenta un buon compromesso tra prestazioni, complessità computazionale in modo da rendere la user experience quanto più coinvolgente.

Capitolo 1: Opere correlate

GeoPhotoHunt è un applicazione che nasce dall'idea di sfruttare la nuove tecnologie introdotte dagli *smartphone* per creare, gestire e giocare a cacce al tesoro interattive che permettono di esplorare città d'arte, città turistiche o percorsi naturali raccogliendo foto di luoghi di rilevanza storica, artistica e paesaggistica. Una caccia al tesoro fotografica è un percorso virtuale in cui le diverse tappe sono descritte da un indovinello che ha per soluzione un'immagine. In seguito all'introduzione delle macchine digitali le cacce al tesoro fotografiche hanno avuto un crescente successo, sia come momento di svago, che come percorso per la visita di città o regioni di interesse culturale. Lo scopo della caccia è quello di superare ciascuna tappa fornendo in risposta l'immagine che rappresenta la soluzione del rispettivo indovinello. L'uso di device evoluti come smartphone e tablet può migliorare l'esperienza di gioco in diversi modi, sia nell'invio delle soluzioni all'organizzatore, che nella gestione dei quiz e dei risultati. GeoPhotoHunt vuole essere una app che comprende, oltre alle funzionalità di creazione e gestione di una caccia, la capacità avanzata di valutazione delle risposte basate su algoritmi di computer vision per confronto automatico delle immagini.

Per poter partecipare ad una caccia al tesoro un giocatore potrà scaricare in locale l'archivio dei dati che include le varie tappe della caccia al tesoro: il percorso dell'intera caccia sarà guidato dall'applicazione che fornisce all'utente gli indovinelli grazie ai quali il giocatore si potrà recare nel luogo in cui deve essere fotografato un particolare soggetto (un edificio storico, una fontana, una piazza, una statua). A questo punto il giocatore dovrà scattare una foto al soggetto preso in considerazione dalla specifica tappa e sarà l'applicazione a determinare, mediante algoritmi di *instance recognition* e geo-localizzazione, se la foto scattata è la risposta corretta all'indovinello della tappa o meno. Quando la caccia viene completata il giocatore totalizza un punteggio (il punteggio diminuisce ogni volta che viene usato un aiuto per completare una tappa) che può decidere di condividere con la community online, in questo modo gli altri utenti potranno consultare la classifica dei migliori giocatori per una determinata caccia al tesoro.

In questi anni il mercato tecnologico sta vedendo la commercializzazione di smartphone e dispositivi mobili sempre più potenti e performanti. Basti pensare che il Nexus 5 realizzato da LG Electronix per Google è dotato di un processore quad-core con frequenza di clock fino a 2.26 GHz, grazie all'introduzione di queste nuove tecnologie si è pensato di poter affidare al device mobile la maggior parte del lavoro svolto dall'applicazione compresi gli algoritmi di *computer vision* che di norma presentano una complessità computazionale elevata. Si è deciso di fare questa scelta al fine di rendere l'applicazione indipendente dalla connessione internet: in

questo modo, per esempio, un utente che visita una città all'estero dove non ha una connessione internet può decidere di scaricare una caccia prima di partire per poi giocarci in modalità offline senza dover usufruire di una connessione. Così facendo il server viene alleggerito dai compiti computazionali e diventa un semplice strumento di supporto per la condivisione delle informazioni tra gli utenti e per la archiviazione dei file dati delle cacce.

1.1 Applicazioni che implementano l'idea del geocaching

L'idea di creare una community che possa giocare, creare diverse cacce al tesoro utilizzando il web non è nuova; sono già presenti diversi siti che permettono di partecipare a "cacce al tesoro" a livello mondiale come Geocaching [1] [*Fig. 1*] o Opencatching [2],. Per questi siti sono state già realizzate delle applicazioni per dispositivi Android che permettono di poter partecipare alla caccia seguendo le indicazioni dal proprio smartphone (per esempio l'app ufficiale di geocaching oppure c:geo [3]). Questo tipo di cacce al tesoro, note con il nome di geocaching, si basano sulla ricerca di contenitori (scatole che conservano il codice relativo alla determinata tappa) precedentemente nascoste da un appassionato. Gli indizi che guidano la ricerca in genere sono costituiti dalle coordinate GPS del luogo in cui il tag da scoprire è nascosto, e il raggiungimento dello scopo avviene marcando il tag per segnalare di averlo trovato (senza alcun controllo supervisionato). La differenza sostanziale tra queste cacce al tesoro e quelle create da GeoPhotoHunt è che queste ultime sono cacce fotografiche e per completare la tappa non sono necessari codici ma basta semplicemente scattare una foto al monumento in cui è situata la tappa.

Figura 1 - Geocaching app snapshot

Altre applicazioni simili che sfruttano gli smarphone al fine di proporre una mappa che offra una visita guidata all'interno di una città sono per esempio: Visito Tuscany [4] che segnala i maggiori punti di interesse per visitare le città d'arte toscane, oppure WhaiWhai un brand di guide turistiche che permette di vistare le città di tutto il mondo in un modo non convenzionale risolvendo enigmi e scoprendo storie originali [5] ora disponibili anche per smartphone, o anche Picture Geo Hunt [6] un'altra app che mostra foto di luoghi di interesse e con una bussola ti guida verso la loro posizione.

Infine, un'altra app che permette di creare giocare e condividere delle cacce al tesoro suddivise in tappe, è Scavify [7] [8] [*Fig. 2*], anche se questa applicazione ha un concetto di caccia al tesoro più ampio: infatti, l'obiettivo di ogni tappa non deve essere per forza un oggetto da trovare o un luogo da raggiugere, ma si può semplicemente trattare di un "compito" da svolgere come scattarsi un selfie assieme a un amico, pubblicare uno stato su Facebook ecc. dando così più spazio alla fantasia di chi crea la caccia.

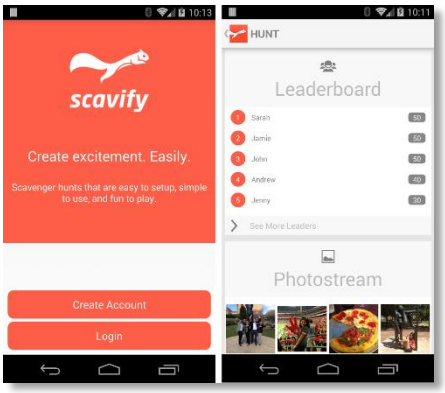


Figura 2 - Scavify snapshot

Dal punto di vista della ricerca scientifica, invece, esistono numerosi studi che hanno come argomento di interesse l'implementazione di framework o applicazioni su dispositivi mobili che permettano la visita di città, musei o permettano l'organizzazione di attività didattiche al di fuori della scuola; in particolare tutte queste ricerche sfruttano le potenzialità dei nuovi smartphone e tablet per aumentare e migliorare i contenuti di attività didattiche, ludiche o turistiche.

- Mobilogue [8] [9] è un framework sviluppato per rendere interattive le attività didattiche che vengono svolte al di fuori della scuola come visite ai musei, gite scolastiche, ecc. L'applicazione permette agli utenti, che possono essere la direzione del museo come gli stessi docenti, di creare contenuti interattivi per rendere più piacevole e coinvolgente l'attività didattica. I contenuti aggiuntivi possono essere: foto, audioguide, brevi descrizioni testuali, coordinate GPS e anche QR code che possono codificare quiz ai quali gli studenti potranno rispondere sempre tramite l'applicazione. A seguito della realizzazione dell'applicazione sono stati fatti test di usabilità e gradimento dell'applicazione in collaborazione con il museo Top Secret a Oberhousen in Germania, questi test hanno evidenziato che l'applicazione favorisce l'apprendimento senza distrarre gli studenti dalla visita al museo.
- Snap2play [10] è una applicazione per dispositivi mobili che consente di fare una caccia al tesoro fotografica ispirata al gioco *Memory;* infatti l'obiettivo del gioco è quello di cercare, "catturare" e accoppiare delle carte fisiche e delle carte digitali. Le carte "fisiche" sono costituite dalla foto dell'area in cui ci si trova; affinché il sistema verifichi che l'immagine sia corretta la foto viene inviata ad un server che si occupa di effettuare il

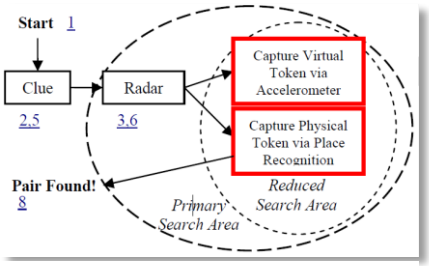


Figura 3 – Architettura dell'applicazione Snap2Play

match. Le carte "digitali" sono virtuali e sono localizzate tramite il GPS [*Fig. 3*] in un determinato punto, l'utente potrà individuarle solo tramite la fotocamera e potrà catturarle scattando una foto quando si trova in una posizione abbastanza vicina alla carta stessa). L'applicazione sfrutta il dispositivo GPS di cui il telefono è dotato per tracciare la posizione del giocatore il quale vedrà apparire sullo schermo del proprio dispositivo una sorta di radar che gli segnala dove sono situati i vari obiettivi da individuare. La documentazione dell'applicazione comprende anche una serie di risultati relativi ad un'alpha test dell'applicazione su un campione di utenti dai 14 ai 33 anni. Questi test hanno evidenziato che il 90% del campione ha apprezzato la tiplogia di gioco e come l'applicazione coinvolge il giocatore nella caccia al tesoro.

- Treasure – HIT [11] [*Fig. 4*] è un'applicazione sviluppata in un progetto che ha l'intenzione di mostrare come possono essere modificate e migliorate le attività ludico-didattiche al di fuori della scuola grazie ad applicazioni per smartphone che implementano um ambiente di gioco per location based games. L'architettura del progetto si divide tra: front-end web, ovvero l'interfaccia dedicata agli insegnati grazie alla quale possono creare le tappe dei percorsi guidati includendo indovinelli testuali e indizi fotografici e induviduando grazie alle Google APIs la locazione geografica degli obiettivi; e front-end per smarphone [*Fig. 5*] dedicato agli alunni che potranno seguire le istruzioni create dagli insegnati e raggiungre gli obiettivi delle varie tappe.

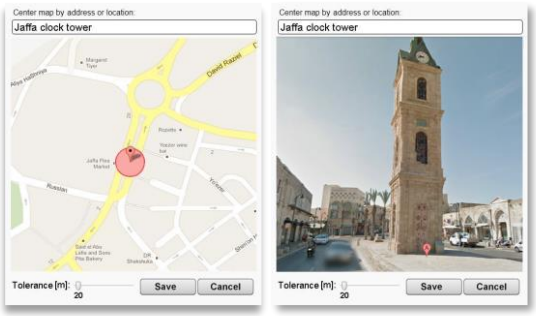


Figura 4 - Snapshot web interface di Treasure HIT

- Lo studio *Fast Authoring for Mobile Gamebased City Tours* [12] presenta un intuitivo sistema di gestione e creazione di visite turistiche di città tramite percorsi guidati con interfaccia web e un'applicazione per dispositivi iOS che permette di partecipare e seguire questi percorsi. L'applicazione [*Fig. 5*] implementa il percorso come una sorta di caccia al tesoro; nell'interfaccia web l'utente può decidere di creare la sua caccia tramite due pannelli di controllo: il primo visualizzando una mappa permette di geolocalizzare i luoghi delle varie tappe, il secondo chiamato *storyline window* permette

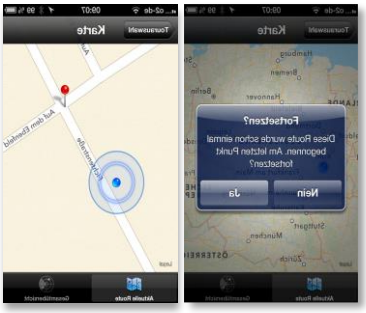


Figura 5 - Snapshot dell'applicazione

di aumentare il contenuto delle varie tappe aggiungendo informazioni, quiz o challenge che il giocatore può completare, in questo pannello è inoltre possibile personalizzare la connessione tra le varie cacce e il loro ordine. Dal lato del device iOS il giocatore tramite l'app *iRendARAR* puo scegliere tra le cacce esistenti quale giocare e così inziare il suo percorso esplorativo all'interno della città. Per trovare il luoghi di interesse gli sarà sufficiente seguire le indicazioni fornite dall'applicazione tramite una mappa, quando sarà giunto nel raggio di interesse dell'obiettivo (definito dal creatore) potra usufruire delle informazioni aggiuntive (se prensenti) cliccando sull'hot spot segnalato sulla mappa.

- See it [13] è un Location-Based Game (LBG) che si interfaccia all'utente via browser [*Fig. 6*]. Il sito è ottimizzato per essere consultato da uno smartphone ma può essere giocato anche tramite pc e stampando gli indizi.

  Il gioco si basa su una caccia al tesoro composta da una singola tappa la quale contiene un indizio testuale, un puntatore sulla mappa che indica il punto di partenza della ricerca al "tesoro", il quale dev'essere situato nel raggio di un chilometro da esso e da una serie di immagini o videoclip.

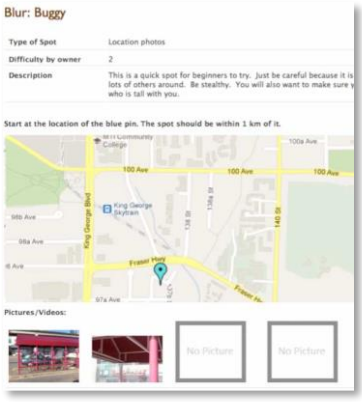


Figura 6 - Interfaccia grafica di See It che mostra il contenitore nascosto e gli indizi per trovarlo

  Il "tesoro" in See It è rappresentato da uno "Spot", recipiente contenente un registro nel quale gli utenti scriveranno il proprio nome una volta individuato.

  See It dà anche la possibilità agli utenti di creare fisicamente i propri Spot e inserirne la posizione nel sito web. Esistono diverse tipologie di Spot:

  i. Location Photo Spots: La tipologia più comune, contiene da una a cinque immagini che rappresentano la locazione dello spot.
  ii. Eye Spy Spots: similare alla tipologia Location Photo ma tutte le immagini devono essere scattate dal punto di vista del contenitore.
  iii. Location Video Spots: usa dei corti videoclip che mostrano la locazione del contenitore
  iv. 360 Spots: videoclip che mostra a 360° la locazione dal punto di vista del contenitore
  v. Path Spots: usa una serie di immagini che progressivamente portano vicino all'obiettivo il giocatore.

See it è stato realizzato con l'obiettivo di coinvolgere i ragazzi nel compiere attività fisica.

- QuestInSitu [14] [*Fig. 7*] è un'applicazione web disegnata e costruita per il supporto a visite didattiche basata su domande a risposta multipla.

Figura 7 - A destra l'editor per creare le rotte, a sinistra l'applicazione web utilizzata per giocare

Gli autori possono creare e localizzare delle domande all'interno di una mappa web dando origine ad una rotta. Le domande seguono lo "educational technology standard for assessment IMS Question & Test Interoperability" (QTI). QuesTInSitu permette anche di giocare le rotte dando la possibilità all'utente di rispondere alle domande geo localizzate di una rotta le quali risposte saranno automaticamente corrette. Usando l'applicativo l'autore può creare due diversi tipi di rotte per attività didattiche:

  - Real: rotte giocate realmente nel posto in cui è stata ideata
  - Virtual: rotte giocate virtualmente, non per forza nel posto nel quale le domande sono state geo localizzate dall'autore.

  L'applicazione di mappe web usata per giocare le domande è Google Maps. QuesTInSitu possiede una versione mobile-web la quale permette agli studenti di: seguire le rotte create dagli insegnanti come supporto a uscite didattiche, rispondere alle domande, ricevere feedback e punteggi e personalizzare la risposta multipla selezionata con immagini e commenti tramite uno smartphone.

1.2 Applicazioni che introducono l'image recognition nei sistemi mobile

Il problema dell'image recognition su piattaforme mobile non è nuovo, sono diversi gli studi che si sono occupati dell'implementazione di algoritmi di questo tipo per smarphone.

Un'applicazione proposta dallo studio *Smartphone Landmark Image Retrieval Based on Lucene and GPS* [15] sfrutta le coordinate GPS e il sistema server Lucene per poter classificare e riconoscere le immagini scattate dagli utenti. L'architettura dell'applicazione riadatta in modalità client-server la classica architettura di un sistema di image retrieval [Fig. 9 - 10] creando un interfaccia Android che funge da client e invia la foto scattata e le coordinate GPS

ad un Tomcat Webserver il quale ha lo scopo di estrarre le features e classificare la foto facendo uso del motore Lucene. Ovviamente in questo modo sia l'estrazione dei descrittori che il calcolo della similarità avvengono nel server, contrariamente a quanto ci si propone di fare nel progetto GeoPhotoHunt. Un pregio di questa architettura è invece l'ottimizzazione dello strumento di ricerca, infatti, grazie al motore Lucene, vengono consultati archivi di immagini molto grandi in poco tempo grazie ad un'efficacie indicizzazione delle risorse, nella fase di matching, inoltre, viene fatta una selezione in base alle coordinate GPS in modo da velocizzare ulteriormente la ricerca senza aumentare il rischio d'errore.

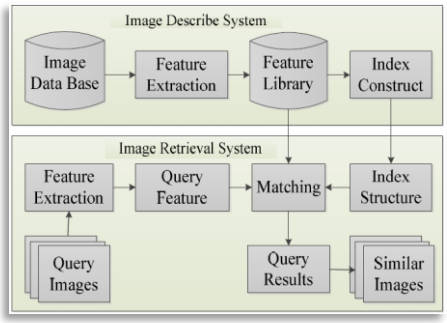


Figura 8 - Esempio di image retrieval system

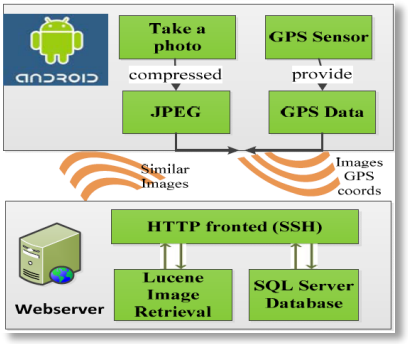


Figura 9 - Architettura del sistema

Lo studio *Image Retrieval System using Intuitive Descriptors* [16] invece si propone di introdurre e implementare il problema dell'image retrieval all'interno di uno smartphone facendo uso di tecniche basilari per il riconoscimento e match delle immagini. La scelta di implementare metodi che calcolano la distanza tra gli istogrammi di colore di un'immagine è dovuto principalmente a due fattori: il primo è costituito dall'intenzione di non sovraccaricare il processore dello smartphone in quanto alcuni metodi complessi per l'instance recognition sono caratterizzati da un a complessità computazionale molto elevata; il secondo invece dipende proprio dal fatto che si vuole introdurre il problema del riconoscimento di immagini gradualmente partendo quindi con un approccio basilare.

Nello studio condotto dal dipartimento di Computer Science della Florida Atlantic University [17] è stata validata un'architettura per trasferire parte del lavoro di image recognition su un dispositivo mobile Android. L'architettura prevede l'implementazione dell'algoritmo di feature extraction sullo smartphone, in particolare si è scelto di implementare l'estrazione dei SURF descriptor in quanto offrono un'efficienza quasi pari a quella dei SIFT descriptor ma hanno un peso computazionale molto inferiore. In questo studio viene utilizzata la libreria java *JopenSurf* compatibile con Android. Una volta estratto, il vettore di descrittori, viene inviato ad un server che si occupa del match con le foto archiviate. In questa fase, come prima operazione,

l’immagine viene categorizzata limitando così il numero di foto con le quali è necessario fare match. Nella fase di confronto vera e propria la distanza tra le immagini viene calcolata con la distanza euclidea. Ovviamente questo tipo di architettura soffre di una latency dovuta al tempo di risposta del server ma si dimostra robusta in termini di efficienza e risultati prodotti, infatti, l’algoritmo di match si è dimostrato resistente a variazioni di inquadratura, luminosità, dimensione dell’immagine e, in misura minore, di rotazione.

Capitolo 2: Studi correlati

2.1 Studi su image recognition precedenti

I Content-Based Image Retieval (CBIR) [18] sono sistemi per il riconoscimento di immagini in base al contenuto, ovvero basati su metodi di ricerca per similarità che non fanno uso di parole chiave. Le applicazioni che beneficiano dell'uso di sistemi CBIR al giorno d'oggi sono sempre più numerose e spaziano tra settori molto diversi tra loro: dalla vita privata al giornalismo alla medicina all'industria. Nei sistemi CIBR le immagini sono rappresentate sotto forma di valori numerici chiamati features o descrittori volti a rappresentare le proprietà dell'immagine in modo utile per poterle catalogare o confrontare.

In letteratura sono state studiate numerose tecniche di rappresentazione e confronto di immagini, ma la scelta di un metodo ottimale non è semplice perché le prestazioni di un sistema sono strettamente correlate al problema da risolvere. Ad esempio descrittori tessiturali si adattano bene al confronto di immagini satellitari per il problema del riconoscimento delle colture, mentre è preferibile usare descrittori basati sul colore per confrontare fotografie naturalistiche. In diversi studi [19] [20] è stato affrontato il problema di valutare e confrontare questi metodi dal punto di vista di efficienza e performance globali al fine di determinare quale tipo di descrittori o quali tipi di descrittori combinati siano più adatti ai diversi settori di sviluppo dei sistemi CIBR.

Come prima cosa va sottolineato che esistono due macro categorie di metodi per estrarre i descrittori:

- ***Approccio discreto**:* questo tipo di approccio produce delle features di tipo binario, per indicare se in una determinata immagine è presente o meno un determinato tipo di feature (come per esempio rilevare se una determinata parola è presente in un testo).
- ***Approccio continuo**:* Rappresenta un immagine come vettori di features, la cui similarità viene poi misurata con diversi tipi di misure di distanza (ad esempio la distanza Euclidea). Le immagini che hanno valori di distanza minori hanno un fattore di similarità maggiore.

L'approccio continuo è certamente il più diffuso e i numerosi metodi che ricadono in questa categoria possono essere ulteriormente distinti in:

- Metodi che rappresentano il colore delle immagini.

- Metodi che rappresentano la texture delle immagini.
- Metodi che descrivono informazioni locali delle immagini.
- Metodi che rappresentano la forma degli oggetti dell'immagine.

Un altro aspetto che non va sottovalutato in questo tipo di algoritmi per il confronto tra immagini è come viene calcolata la differenza o distanza tra le features estratte; infatti, a seconda del calcolo effettuato possiamo apprezzare una considerevole differenza sia in termini di efficienza che di prestazioni.

Le features prese in esame in questo studio sono:

- ***Istogramma di Colore**:* ovvero un istogramma che riporta per ogni sfumatura di grigio il numero di pixel di quel colore presenti nell'immagine. Una istogramma di colore può essere esteso a immagini non grayscale concatenando gli istogrammi di colore dei tre canali cromatici della scala RGB o HSL, in questo studio viene utilizzata la scala RGB e viene utilizzata la distanza di Jensen-Shannon per misurare la similarità tra i valori.
- ***Tamura features**:* un tipo di texture features basate sulla percezione umana di granularità, contrasto, direzionalità, regolarità e ruvidità. In questo studio sono state utilizzate alcune di queste per creare degli istogrammi che descrivono la tessitura dell'immagine dopo di che vengono confrontati con la distanza di Jeffrey.
- ***Global textures descriptor**:* è un descrittore di tessitura che riporta risultati soddisfacenti per immagini utilizzate in campo medico come radiografie.
- ***Invariant features histograms**:* sono degli istogrammi che raccolgono features che rimangono costanti se l'immagine viene sottoposta a trasformazioni affini: rotazione, traslazione, ridimensionamento.
- ***MPEG-7 Features**:* sono dei descrittore di colore scalabili basati su istogrammi di colore della scala cromatica HSV e codificati con la trasformata di Haar. Vengono presi in esame in quanto sono computazionalmente poco costosi.
- ***Local image descriptors**:* sono un nuovo tipo di descrittore che si sta rivelando molto efficiente in diversi campi di sviluppo, sfrutta l'individuazione all'interno dell'immagini di punti salienti (detti keypoints) e su questi punti vengono poi estratte delle features locali. In questo studio i keypoints sono stati individuati tramite l'algoritmo di HARRIS e su questi sono stati estratti i SIFT descriptors. Nello specifico i descrittori SIFT sono degli

istogrammi 3D che mantengono la posizione e l'orientamento del gradiente pesando la rilevanza del valore nella finestra bidimensionale centrata sul keypoint seguendo la distribuzione gaussiana. In questo modo concatenando gli istogrammi si denota come il gradiente locale attorno al punto è allineato. Un miglioramento dei descrittori SIFT sono i descrittori SURF (Speeded Up Robust Features) che estraggono i descrittori dai punti di interesse individuati partendo da un metodo basato sulla matrice di Hessian. I descrittori vengono estratti dalla regione attorno al punto come risposta alla Haar Wavelet.

- ***CENsus TRansform hISTogram (CENTRIST):*** [21] è un nuovo descrittore (ideato nel 2011) per il riconoscimento di paesaggi naturali e per la categorizzazione di scene, ad esempio per affermare che una foto ritrae un paesaggio marittimo o collinare o una camera da letto piuttosto che una cucina. Per operazioni di questo tipo (soprattutto per la categorizzazione di ambienti interni) si è dimostrato che le proprietà dei descrittori visuali comunemente utilizzati non sono adatte, per questo si è creato il descrittore CENTRIST che codifica le informazioni strutturali di un'immagine e tralascia le dettagliate informazioni tessiturali. Negli studi è stato fatto anche un confronto di prestazioni con i descrittori SIFT e GIST per questo tipo di problemi dimostrando che CENTRIST è più veloce, efficiente e facile da implementare.

- ***PACT****:* Principal component Analysis of Census Trasform histograms [22]; si tratta di un descrittore ottimizzato per riconoscere la tipologia di un determinato paesaggio o di un ambiente interno (es. foresta, costa, strada, camera da letto, salotto ecc.). PACT utilizza la trasformazione Census (CT) la quale dato un pixel ne ricalcola il valore nel seguente modo: vengono analizzati i pixel a esso circostanti in un area di dimensione e forma fissate. Ogni pixel di quest'area viene confrontato con il pixel generatore. Se il pixel contiene una gradazione di grigio maggiore del pixel generatore, viene associato un bit 1 mentre se ha gradazione inferiore un bit 0. Seguendo un ordine prefissato si concatenano i bit dell'area fissata generando una stringa binaria la quale verrà convertita in un numero decimale da 0 a 255. Il valore ottenuta rappresenterà il nuovo tono di grigio del pixel generatore. L'istogramma calcolato sull'immagine ottenuta tramite la trasformazione Census codifica implicitamente le forme globali dell'immagine di partenza che saranno essenziali per il riconoscimento del luogo o scena che l'immagine ha catturato. PACT è in grado di ottenere le componenti principali sintetizzando l'istogramma ottenuto dalla trasformazione Census. In breve PACT offre un risultato migliore su diversi dataset di immagini standard sia per la classificazione di

ambienti interni che esterni senza bisogno di parametrizzazione ad una velocità di valutazione superiore.

Capitolo 3: Progettazione & Implementazione

Il progetto GeoPhotoHunt (GPH) si propone di investigare e realizzare le tecnologie per la messa in opera di un servizio avanzato di caccia al tesoro fotografica per la visita di città d'arte o a scopo ludico per dispositivi Android. Il sistema realizzato è basato su un'architettura client – server ed è stato progettato per essere scalabile in modo da supportare una grande quantità di giocatori.

L'applicativo è stato progettato seguendo l'idea di dare la possibilità di giocare cacce al tesoro anche in assenza di una connessione ad internet, mantenendo quindi tutte le informazioni utili nel client.

Grazie al server è possibile registrarsi e autenticarsi, condividere con la community le proprie cacce e scaricare nuove cacce rese disponibili da altri utenti. Il server dà anche la possibilità di registrare il punteggio finale che si è ottenuto completando una caccia e gestire una classifica.

Tutta la dinamica di creazione e utilizzo del gioco è invece contenuta nel client, allo scopo di rendere il gioco indipendente dalla presenza di una connessione con il server. Lato client, GeoPhotoHunt permette di giocare completamente offline ogni singola caccia scaricata dal server o creata dall'utente, non sarà necessario eseguire login per giocare, bensì soltanto nel momento in cui si vorrà pubblicare il punteggio sul server.

3.1 Diagramma dei casi d'uso

Gli Use Case Diagram (diagrammi dei casi d'uso) in ingegneria del software rappresentano una lista di azioni che tipicamente definiscono le interazioni tra un attore (termine utilizzato in UML per definire un ruolo) e il sistema per il raggiungimento di un obiettivo. L'attore può essere un umano, un sistema esterno o il tempo.

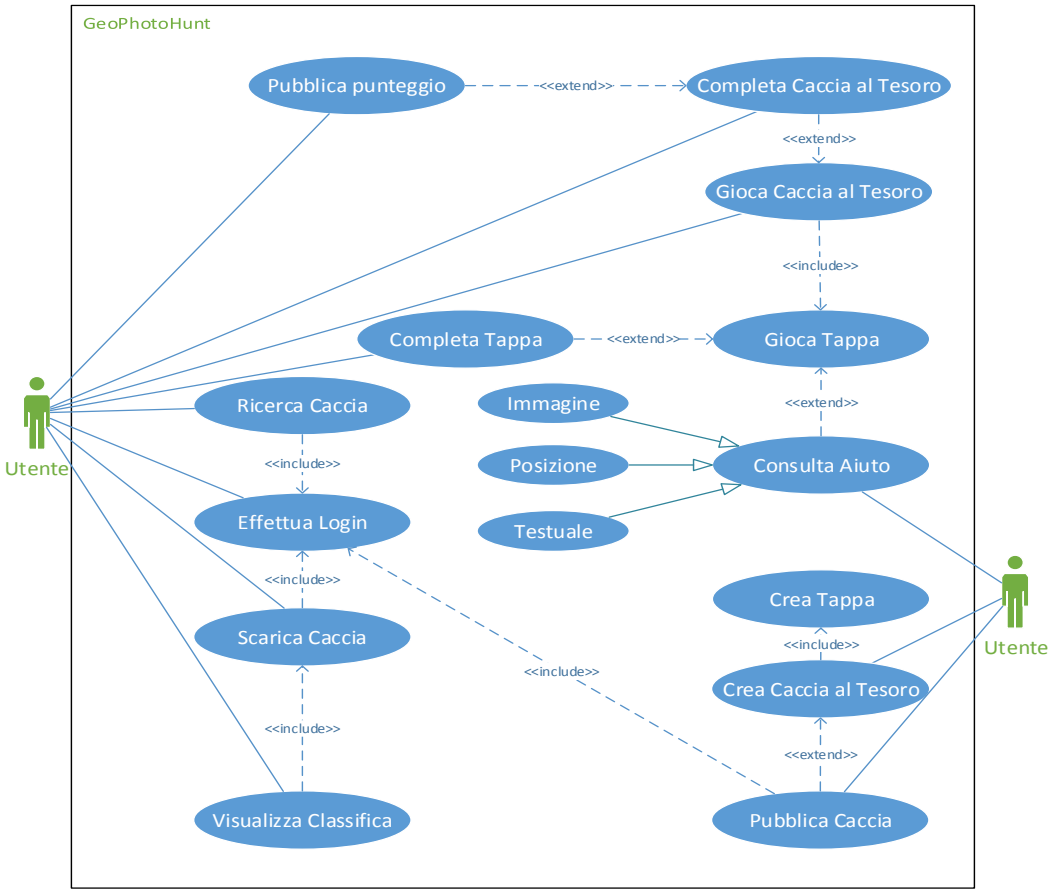


Figura 10 - Diagramma dei casi d'uso di GeoPhotoHunt

Nell'applicazione client GeoPhotoHunt, l'unico attore che può compiere azioni è l'utente. Le azioni si differenziano in tre diverse categorie:

Giocare: Comprende tutte le azioni che consentono di portare a termine una caccia al tesoro come giocare e completare le tappe, richiedere gli aiuti e completare la caccia al tesoro.

Creare: Comprende tutte le azioni che consentono la creazione di una caccia al tesoro ex novo o modificarne una esistente come scegliere il nome della caccia, la tipologia e altre informazioni generali, creare le tappe digitando l'indizio, inserendo gli aiuti e caricando l'immagine sulla quale verrà eseguito l'algoritmo di matching con una seconda immagine fornita come risposta da un giocatore al quale, nel caso l'esito sia positivo, permetterà il completamento della tappa.

Community: Comprende tutte le azioni che consentono all'utente di interagire con la community online tramite il server. Questa categoria di azioni permette all'utente di condividere con tutto il mondo le proprie cacce al tesoro create, ricercare e scaricare cacce al tesoro caricate da altri utenti iscritti al server di GeoPhotoHunt, pubblicare i risultati ottenuti completando cacce al tesoro di altri utenti e visualizzare le classifiche.

Le cacce al tesoro scaricate e i relativi punteggi ottenuti non sono vincolati a un singolo account identificato sul server, bensì allo smartphone. Per vincolare le informazioni all'account sarà opportuno comunicare le stesse al server tramite le azioni che permettono la pubblicazione del punteggio ottenuto completando una caccia o la pubblicazione di una caccia al tesoro precedentemente creata. Anche se può risultare contorto, questo metodo rende l'applicazione ancor meno vincolata alla presenza di una connessione internet.

3.2 Diagramma delle activity

Figura 11 - Diagramma Gioca Caccia

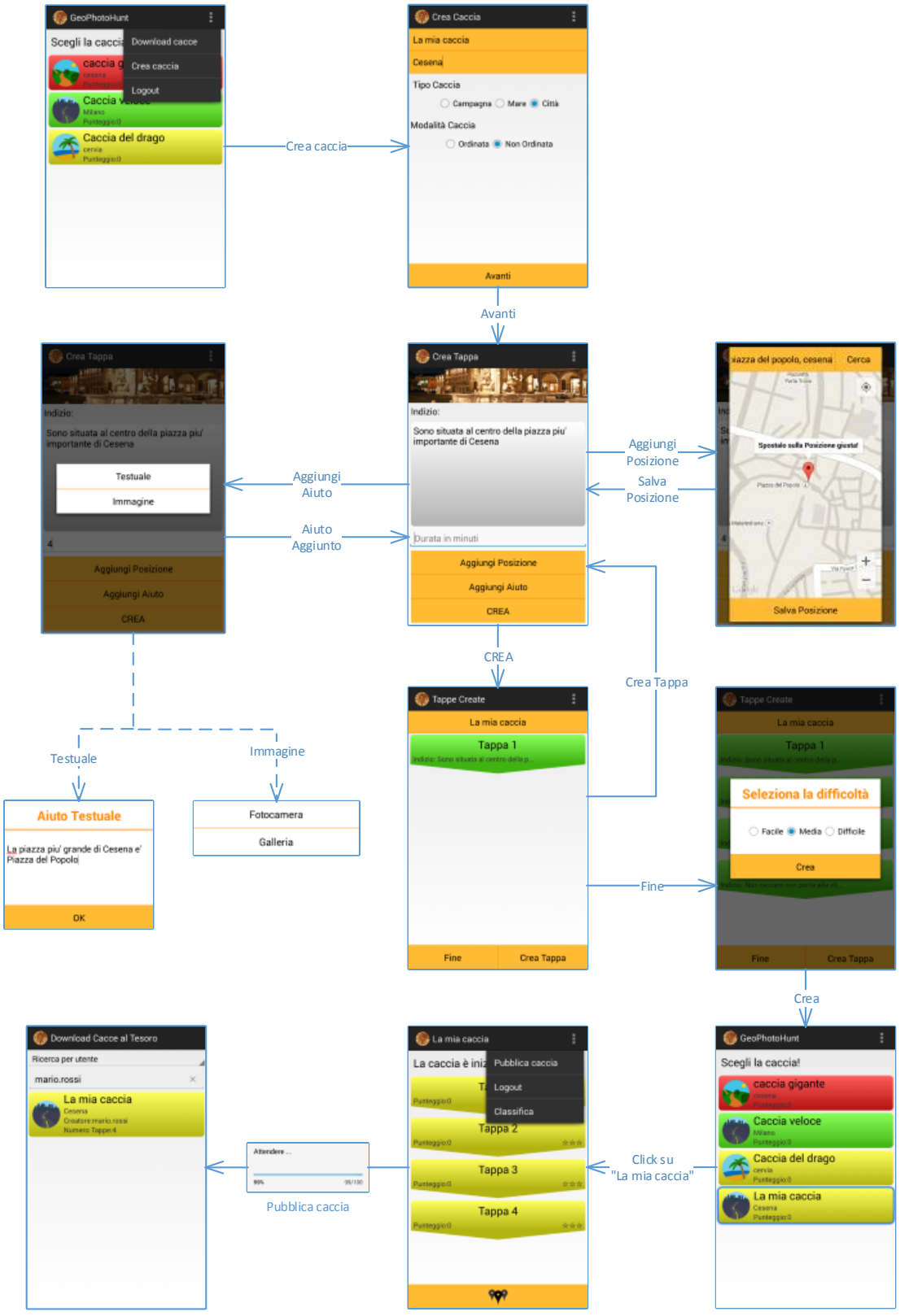


Figura 12 - Diagramma creazione di una caccia al tesoro

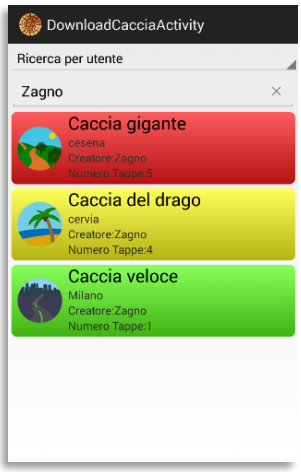


Figura 13 - Download cacce

Installata l'applicazione sul proprio dispositivo, al primo utilizzo GeoPhotoHunt non conterrà cacce al tesoro, per iniziare a giocare sarà necessario scaricare una nuova caccia o crearne una. Per avere accesso al download di nuove cacce al tesoro, sarà obbligatorio registrarsi alla community online e accedervi. La pagina dedicata al download di nuovo contenuto [*Fig. 13*] mostra inizialmente le cacce al tesoro presenti nella città dell'utente e permette la ricerca di cacce al tesoro in tre diverse tipologie di ricerca: ricerca per utente, per nome e per città.

La lista delle cacce contiene le informazioni principali di ogni caccia, selezionando una voce dalla lista, si otterranno tutte le informazioni relative alla caccia selezionata e il bottone per effettuarne il download.

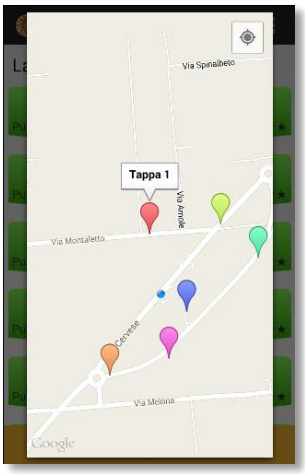


Figura 14 - Resoconto tappe terminate

Al completamento del download la caccia apparirà nella pagina principale dell'applicazione dalla quale sarà possibile avviarla.

Avviata una delle cacce scaricate presenti nella pagina principale, sarà possibile giocare le tappe, controllare la classifica, caricarla sul server nel caso si sia il creatore e, tramite un bottone posto alla base della pagina, visualizzare la mappa sulla quale saranno posizionati i segna posti delle tappe già portate a termine [*Fig. 14*].

Sono due le varianti di caccia al tesoro implementate:

- **Ordinata:** si tratta di una vera e propria caccia al tesoro divisa in tappe le quali possono essere giocate solo sequenzialmente, ovvero l'indizio di ogni tappa può essere visualizzato solo al completamento della precedente.
- **Non ordinata:** ogni tappa di una caccia al tesoro non ordinata può essere giocata indipendentemente dalle altre, senza l'obbligo di seguire un ordine preciso.

La scelta delle due modalità di gioco viene fatta dal creatore della caccia e determina un diverso approccio al percorso, nel primo caso il percorso è prestabilito da chi crea la caccia, nel secondo caso è l'utente che può leggere tutti gli indizi e decidere l'ordine da seguire per minimizzare gli spostamenti.

Figura 15 - Tappa di una caccia al tesoro

Ogni tappa è costituita da un indizio testuale, solitamente scritto a indovinello, il quale porta l'utente a localizzare e identificare un soggetto da fotografare. Viene in aiuto alla localizzazione una bussola speciale che punta all'obbiettivo e permette la visualizzazione della distanza in metri da esso [*Fig. 15*]. Nel caso non fossero sufficienti gli aiuti di base è possibile richiederne ulteriori tramite il bottone in basso a destra [*Fig. 15*] a forma di bulbo luminoso. Gli aiuti aggiuntivi sono di tre tipologie differenti: immagine, testo, posizione. Ogni tappa è sempre provvista dell'aiuto aggiuntivo che identifica l'obbiettivo sulla mappa mentre gli aiuti testuali e fotografici possono non essere previsti. La richiesta di un aiuto aggiuntivo determina una penalità al punteggio finale della tappa. Localizzato e identificato l'obbiettivo, per completare la tappa sarà necessario fotografarlo con la fotocamera del dispositivo. Accettando la foto catturata il sistema verificherà la correttezza della risposta fornita. La verifica viene basata su tre diverse informazioni:

- Distanza tra la posizione dell'obiettivo e dell'utente
- Orientamento dell'utente rispetto all'obiettivo
- Somiglianza tra l'immagine scattata e quella campione

Al termine della verifica un messaggio mostrerà l'esito [*Fig. 16*]; se positivo, l'utente sarà reindirizzato alla pagina principale della caccia al tesoro.

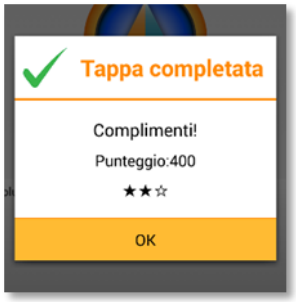


Figura 16 - Completamento tappa

Figura 17 - Creazione di una nuova tappa

Portate a termine tutte le tappe previste dalla caccia al tesoro corrente, apparirà un messaggio in sovraimpressione con il punteggio finale ottenuto sommando i punteggi parziali di ogni tappa, sarà inoltre possibile inviare il punteggio al server.

GeoPhotoHunt offre anche la possibilità di creare cacce al tesoro. Per accedere al servizio non occorre essere iscritti alla community online. Per creare una caccia per prima cosa occorre specificare nome, città, scenario della caccia (campagna, mare, città) e la tipologia (ordinata, non ordinata). Inserite le informazioni principali si verrà reindirizzati alla pagina adibita alla creazione delle tappe [*Fig. 17*] dalla quale sarà possibile inserire l'immagine di risposta, digitare l'indizio,

inserire in media quanto tempo in minuti occorre per portare a termine la tappa, aggiungere la posizione geografica dell'obbiettivo cercando l'indirizzo e spostando il segna posto nella giusta posizione [*Fig. 18*] e inserire gli aiuti extra (testo aggiuntivo, immagine). Premendo il bottone CREA [*Fig. 17*] la tappa verrà aggiunta alla caccia al tesoro e il sistema reindirizzerà l'utente alla pagina principale del servizio dalla quale sarà possibile creare una nuova tappa, modificare una tappa già creata o terminare la creazione della caccia.

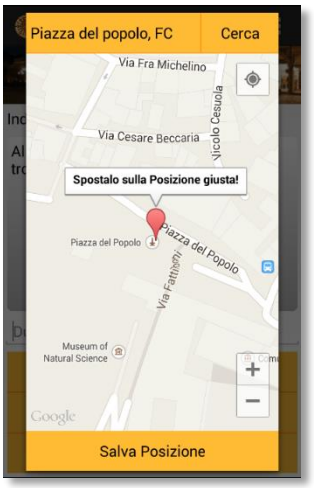


Figura 18 - Selezione della posizione dell'obiettivo

Se si sceglie di terminare la creazione, la nuova caccia apparirà nella lista delle cacce giocabili consultabile alla pagina principale dell'applicazione. Selezionando la caccia creata dalla lista, sarà possibile pubblicarla online rendendo disponibile il download a tutta la community di GeoPhotoHunt.

3.3 Progettazione logica dell'applicazione

La progettazione logica del sistema è divisa nelle due parti che lo compongono, ovvero lato client e lato server.

Il primo storicizza tutte le cacce al tesoro nella memoria di massa del dispositivo le quali sono composte da un file di estensione geoph contenente tutte le informazioni come indizi delle tappe, posizioni degli obiettivi ecc. e una cartella contenente le immagini degli aiuti fotografici nel caso la caccia ne sia provvista. Si utilizza un database locale per memorizzare le informazioni salienti di ogni caccia al tesoro. Il Client gestisce l'applicazione in termini di utilizzo dello smartphone adoperando dispositivi hardware come la fotocamera, l'antenna GPS, il giroscopio e il magnetometro, e interrogando il server per ottenere, aggiornare e creare contenuto necessario al funzionamento.

Lato server il sistema memorizza tutte le informazioni della community nella base di dati online, in particolare gli utenti e le cacce al tesoro caricate, e riceve le interrogazioni dal client.

3.3.1 Lato client

La storicizzazione delle cacce al tesoro sul dispositivo è stata progettata in modo da non appesantire l'esecuzione dell'applicazione. Ogni caccia al tesoro, composta da un file

contenente tutti i dati e una cartella contenente gli aiuti fotografici delle tappe, può raggiungere una dimensione notevole. Se questa dimensione viene moltiplicata per tutte le cacce in possesso all'utente, il caricamento della pagina iniziale dell'applicazione [*Fig. 1*3], che richiede le informazioni di base di tutte le cacce memorizzate, avrebbe un costo computazionale elevato. Per ovviare al problema si è deciso di adottare un database locale nel quale indicizzare le cacce in possesso.

Struttura di una caccia al tesoro

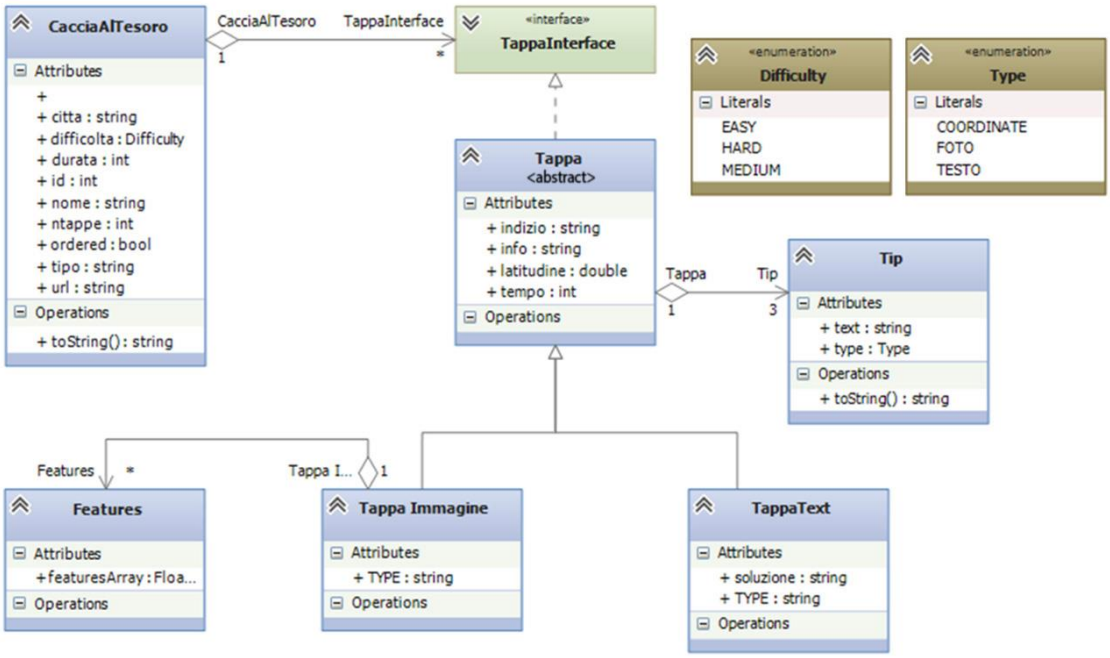


Figura 19 - Diagramma delle classi (Caccia Al Tesoro)

Il diagramma delle classi [*Fig. 19*] rappresenta come è strutturata una caccia al tesoro in GeoPhotoHunt. Le informazioni principali come durata, difficoltà, nome della caccia risiedono nella classe principale mentre le tappe sono collegate alla caccia al tesoro. La classe Tappa è astratta, non può essere implementata, non vale lo stesso per le classi tappa immagine e tappa text, derivate di Tappa. Ogni tappa è prevista di almeno un aiuto (Tip) fino ad un massimo di tre. La classe Features è parte della classe tappa immagine e contiene le informazioni estratte dall'immagine fornita come soluzione. Nel file geoph è contenuta la serializzazione della classe Caccia Al Tesoro [*Fig. 19*].

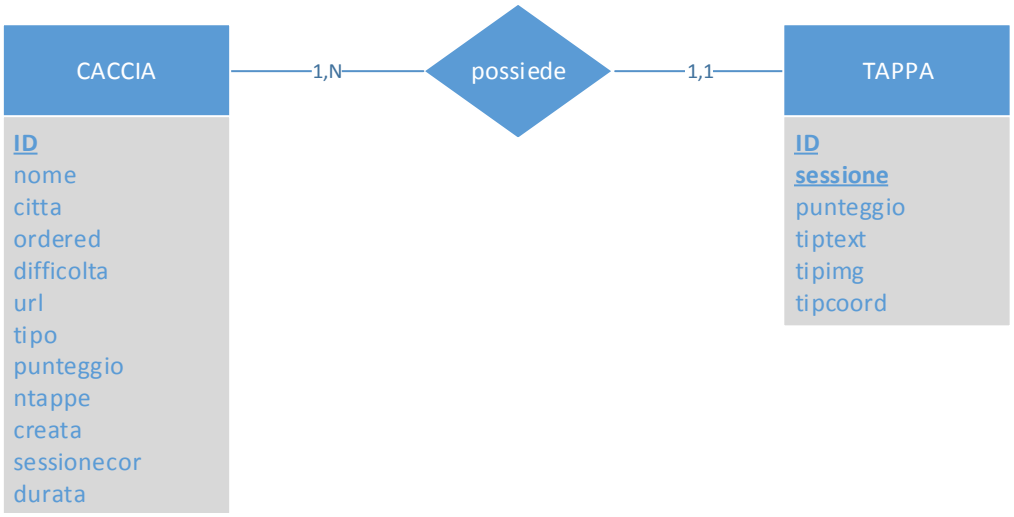
L'attributo "id" della classe CacciaAlTesoro conserverà il numero identificativo che il server ha associato alla caccia. Se la caccia viene creata dall'utente avrà assegnato come id il valore -1.

Conservare il numero identificativo della caccia al tesoro lato server sarà utile nel momento in cui l'utente vorrà pubblicare il punteggio ottenuto completando la caccia al tesoro.

Database locale SQLite

Come già detto si è deciso di adottare un database lato client per indicizzare le cacce al tesoro scaricate e create dall'utente. La base di dati locale ha anche il compito di memorizzare i punteggi delle cacce completate e gli aiuti extra utilizzati dall'utente in ogni tappa. Per la realizzazione della base di dati lato client si è scelto di utilizzare SQLite.

SQLite è una libreria software che implementa un SQL database engine indipendente il quale non richiede la presenza di un server e non necessita di una configurazione. SQLite è il SQL database engine più utilizzato al mondo. Il codice sorgente di SQLite è di pubblico dominio.



CACCIA(ID, nome, citta, ordered, difficolta, url, tipo, punteggio, ntappe, creata, sessionecor, durata)

TAPPA(ID, sessione, idCaccia:CACCIA, punteggio, tiptext, tipimg, tipcoord)

Figura 20 - Sopra schema Entity-Relationship del database locale, sotto schema logico del database locale

Il database locale [*Fig. 20*] è composto da due entità le quali hanno il compito di storicizzare tutte le informazioni adibite al gioco. Ogni tupla dell'entità caccia indicizza una caccia al tesoro presente nel dispositivo conservando le informazioni principali estratte dal file geoph come nome, città, tipologia della caccia (ordered), difficoltà ecc. Oltre alle informazioni estratte, ogni tupla contiene il path al file storicizzato nella memoria del device, la sessione di gioco corrente e il punteggio complessivo.

L'entità tappa storicizza le informazioni delle tappe giocate. ID e sessione compongono la chiave primaria dell'entità tappa. Sessione in chiave dà la possibilità di rigiocare una caccia al tesoro una volta completata.

Gli attributi tiptext, tipimg e tipcoord hanno il compito di memorizzare l'utilizzo degli aiuti extra. Ogni tupla di tappa contiene il punteggio parziale ottenuto completando la tappa.

Upload & download delle cacce

Per garantire il gioco offline è opportuno avere l'intera caccia al tesoro salvata nel proprio dispositivo. Una caccia al tesoro è composta da un numero variabile di file, questo comporta un problema rilevante nella fase di download e upload al server. Per porre rimedio al problema è stato deciso di comprimere in un singolo file zip i file che compongono la caccia. GeoPhotoHunt lato client è in grado di comprimere una caccia al tesoro in un singolo file zip e decomprimere, indicizzare e rendere giocabile una caccia al tesoro scaricata in un singolo file compresso.

3.3.2 Lato server

Il server ha la funzione di strumento di appoggio per la gestione della community online degli utenti iscritti a GeoPhotoHunt e delle cacce al tesoro che questi ultimi hanno deciso di condividere con gli altri.

Il server in particolare offre i servizi di:

- Registrazione e login
- Storicizzazione delle cacce: quando un utente decide di condividere una caccia che ha creato con gli altri giocatori potrà caricarla tramite il proprio smartphone. Una volta che il server riceve il file della caccia, descritto nei capitoli precedenti, gli assegna un "id" univoco, lo posiziona nella specifica directory dopo di che storicizza i dati essenziali della caccia aggiungendoli al database.
- Download delle cacce: allo stesso modo, quando un utente tramite smartphone cerca delle nuove cacce al tesoro da scaricare, interroga il database lato server. Selezionata una caccia può scaricarla effettuando così una richiesta al server.
- Salvataggio dei punteggi: quando un utente completa una caccia al tesoro può scegliere di condividere il proprio risultato con gli altri giocatori, scegliendo di caricare il proprio

punteggio il giocatore invia le informazioni necessarie al server il quale si occuperà di salvarle nella propria base di dati.

Per la realizzazione dei servizi sopracitati si è scelto di utilizzare le seguenti tecnologie:

- **MySQL***:* è un Relational database management system (RDBMS) disponibile sia per sistemi Unix che per Windows. MySQL è un software libero rilasciato a doppia licenza, compresa la GNU General Public License ed è sviluppato per essere il più possibile conforme agli standard ANSI SQL e ODBC SQL. I sistemi e i linguaggi di programmazione che supportano MySQL sono molteplici: ODBC, Java, Mono, .NET, PHP e altri. Per la realizzazione del database lato server si è deciso di utilizzare MySql in quanto è una tecnologia gratuita ma che offre un servizio di gestione dei dati sufficientemente potente per le nostre necessità. Esistono diversi MySQL manager, ovvero strumenti che rendono più agevole l'amministrazione, in questo caso abbiamo utilizzato MySQL WorkBench per facilitare la creazione della struttura del database lato server.

- **PHP***: (Hypertext Preprocessor)* è un linguaggio di programmazione interpretato, originariamente concepito per la programmazione di pagine web. L'interprete PHP è un software libero distribuito sotto la PHP License. Attualmente è principalmente utilizzato per sviluppare applicazioni web lato server ma può essere utilizzato anche per applicativi stand alone. Si è deciso di utilizzare questo linguaggio di programmazione in quanto è free ed è interfacciabile con MySQL.

- **JSON***:* (JavaScript Object Notation) è un formato per lo scambio di dati. È un metodo di codifica semplice da generare per le macchine e facilmente leggibile dalle persone. Si basa su un sottoinsieme del Linguaggio di Programmazione JavaScript. JSON è un formato di testo completamente indipendente dal linguaggio di programmazione ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, C#, Java, JavaScript, Perl e altri. JSON è basato su un insieme di coppie nome/valore per questo virtualmente tutti i linguaggi di programmazione moderni supportano questo formato. Un oggetto JSON è una serie non ordinata di nomi/valori. Un oggetto inizia con " { " e finisce con " } ". Ogni nome è seguito da " : " e la coppia di nome/valore sono separata da " , ". È stato scelto l'uso del formato JSON per lo scambio di informazioni tra client e server in quanto è un formato facilmente leggibile, ideale per la fase di debug, cross-platform e cross-language e per questo ideale per trasferire dati tra client e server.

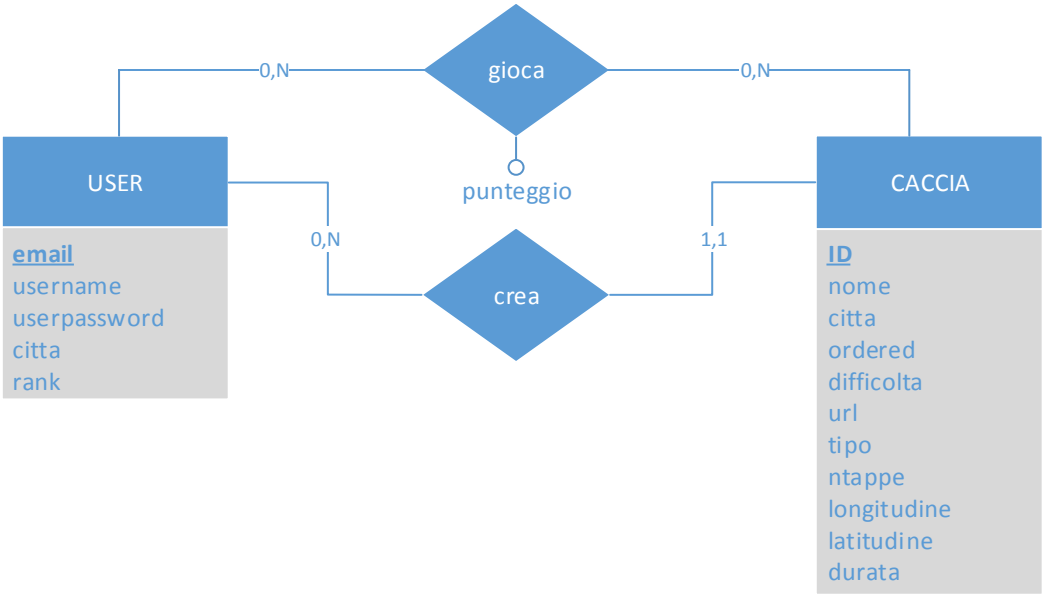
```
[
   {
      "id":"15",
      "nome":"Caccia gigante",
      "ntappe":"5",
      "ordered":"1",
      "tipo":"campagna",
      "citta":"cesena",
      "durata":"9",
      "url":"uploads\/08152014125251ncsc017kqk.zip",
      "difficolta":"HARD",
      "username":"Mario.Rossi",
      "longitudine":"12.3121584579349",
      "latitudine":"44.1939817211012"
   },
   {
      "id":"21",
      "nome":"Caccia del drago",
      "ntappe":"4",
      "ordered":"1",
      "tipo":"mare",
      "citta":"cervia",
      "durata":"121",
      "url":"uploads\/08282014164042elpby88xl0.zip",
      "difficolta":"MEDIUM",
      "username":"Luca.Bianchi",
      "longitudine":"12.3092757537961",
      "latitudine":"44.1980330738109"
   }
]
```


Figura 21 - Esempio di come vengono comunicate all'utente le cacce ricercate tramite la notazione JSON

Architettura del database lato server

Per la progettazione del database si è prima modellato lo schema Entity-Relationship. Osservando il diagramma [*Fig. 22*] si nota che ogni tupla dell'entità CACCIA non contiene solamente l'id della tappa e il path in cui risiede all'interno del server ma contiene anche le informazioni generali utilizzate in risposta ad una richiesta dell'utente durante la ricerca di cacce al tesoro da scaricare (esempio di invio delle informazioni generali [Fig. 22]. Per ogni caccia è inoltre storicizzata all'interno del database la posizione (longitudine e latitudine) della prima tappa in modo da far visualizzare al client le cacce più vicine alla sua posizione corrente.

Per quanto riguarda l'utente invece sono memorizzate le informazioni che vengono richieste durante la fase di registrazione e il rank del giocatore, ovvero che privilegi ha il determinato giocatore (inizialmente viene attribuito il rank 0 ovvero quello più basso, solo gli amministratori del sistema possono cambiare il rank di un utente). Ogni caccia al tesoro creata è legata al suo creatore. Nel momento in cui il giocatore pubblicherà un punteggio ottenuto portando a termine una caccia, questo verrà storicizzato nel database.



USER(email, username, userpassword, citta, rank)

CACCIA(ID, nome, citta, ordered, difficolta, url, tipo, ntappe, creatore:USER, longitudine, latitudine, durata)

PUNTEGGIO(caccia:CACCIA, user:USER, punti)

Figura 22 – Sopra schema Entity-Relationship del database lato server, sotto schema logico del database lato server

Script PHP

Gli script PHP sono pagine web che vengono richiamate dal client visitando l'indirizzo web della pagina stessa e fungono da tramite per la comunicazione tra client e server. Richiamando una determinata pagina, per esempio, il client può effettuare delle query per interrogare il database MySQL del server oppure effettuare il login ecc.

Le informazioni estratte tramite gli script PHP saranno poi trasformate in formato JSON per poi essere restituite al client che si occuperà della lettura.

Funzionamento dello script che fornisce il servizio di upload delle cacce al tesoro: quando la caccia viene caricata lo script prima verifica l'autenticazione dell'utente e controlla che la caccia non sia già presente nel database, di seguito scarica il file zip contenente le informazioni della caccia. Se questa operazione ha esito positivo procede con l'inserimento delle informazioni generali, fornite dal client in POST, della caccia all'interno del database. Al termine dell'operazione il server informa il client dell'esito dell'operazione.

3.4 Implementazione dell'algoritmo di image recognition

Sebbene la risposta fotografica fornita dall'utente, volta a completare una tappa di una caccia al tesoro, venga classificata con buone probabilità in modo corretto grazie alla posizione GPS e l'orientamento del dispositivo al momento di cattura dell'immagine, nel caso peggiore può succedere che nel momento di acquisizione della risposta fotografica, pur essendo nella posizione corretta e orientati verso l'obiettivo, il segnale GPS sia poco accurato e la bussola del dispositivo tarata male. In questo caso, basare l'intera classificazione della risposta su queste due informazioni porterebbe a classificare la risposta come incorretta pur essendo di natura corretta.

Per ridurre ulteriormente l'errore commesso nella classificazione delle risposte fotografiche, si è scelto di utilizzare un algoritmo di riconoscimento di immagini il quale, a partire dall'immagine risolutiva inserita nella tappa al momento di creazione della caccia al tesoro e la risposta fotografica fornita dal giocatore, restituisce un valore compreso tra 0 e 1, dove 1 indica che le immagini sono identiche e 0 la completa diversità.

La similarità tra due immagini non viene calcolata direttamente mettendo a confronto le due immagini per come l'occhio umano le percepisce; per ogni immagine viene estrapolato un vettore di informazioni ed il valore che identifica la similarità tra le due immagini, viene calcolato mettendo a confronto i vettori di informazioni estrapolati precedentemente dalle due immagini.

Più precisamente, per calcolare la similarità tra due immagini occorre implementare due algoritmi separati: "Features extractor" e "Distance calculator" dove il primo estrapola le informazioni dalle immagini ed il secondo mette a confronto le informazioni ottenute dal primo restituendo il valore di similarità tra le due immagini.

Da ora in poi sarà utilizzato il termine distanza assumendo che la similarità si calcoli:

$$\text{Similarità} = 1 - \text{distanza}$$

In seguito a test effettuati su diversi algoritmi di classificazione di immagini applicati su altrettanti dataset di immagini, si è scelto di utilizzare come features l'**istogramma HSV** dell'immagine e come distanza la **Chi Square Distance**.

Per un migliore approfondimento sui test effettuati e le motivazioni che hanno portato alla scelta degli algoritmi implementati si rimanda alla quarta capitolo.

Dato che l'istogramma HSV estratto occupa 240 byte in memoria, indifferentemente dall' immagine di partenza, per alleggerire il file finale della caccia al tesoro si è scelto di salvare l'istogramma al posto dell'intera immagine. Salvando il descrittore, diminuisce anche il tempo di download e il processo di verifica dato che sarà necessario estrarre solamente un istogramma HSV e non entrambi.

Inoltre si è scelto di dare maggiore importanza alla posizione GPS e all'orientamento piuttosto che al parametro di similarità ottenuto tra le due immagini, ovvero si è scelto di controllare per prima cosa la posizione e l'orientamento; nel caso non fossero corretti, la classificazione della risposta fotografica verrà definita dal solo parametro di similarità ottenuto tra le due immagini. In ogni caso il parametro di similarità agirà da moltiplicatore sul risultato finale.

3.4.1 HSV histogram extractor

L'estrattore di istogramma HSV, ricevendo come parametro l'immagine bitmap, deve per prima cosa convertire l'immagine in un vettore di interi contenenti i tre valori RGB per ogni pixel e di seguito convertirli in formato HSV. L'istogramma HSV è dato dalla concatenazione degli istogrammi della tinta (Hue), della saturazione (Saturation) e della luminosità (Value). Per ottenere migliori prestazioni si è deciso di dare più importanza all'istogramma della tinta che agli istogrammi della saturazione e della luminosità. Il campionamento dell'istogramma della tinta è in rapporto 4:1 ai restanti due istogrammi. L'istogramma è rappresentato da un vettore di double da 30 posizioni dove le prime 20 rappresentano l'istogramma della tinta, le successive 5 l'istogramma della saturazione e le restanti l'istogramma della luminosità. In fine l'istogramma HSV viene normalizzato dividendo ogni cella dell'array per il numero di pixel totali dell'immagine.

Si riporta di seguito il codice:

```
public static double[] getHSVHist(Bitmap src)
{
    int w = src.getWidth();
    int h = src.getHeight();

    int npixels = w*h;

    int[] mapSrcColor = new int[w * h];
    float[] pixelHSV = new float[3];

    double hHistTemp[] = new double[360];
    double hHist[] = new double[20];
    double sHist[] = new double[5];
    double vHist[] = new double[5];

    double hsvHist[] = new double[hHist.length+sHist.length+vHist.length];

     /*
      * pixelHSV[0] : Hue (0 .. 360)
      * pixelHSV[1] : Saturation (0...1)
      * pixelHSV[2] : Value (0...1)
      */

    src.getPixels(mapSrcColor, 0, w, 0, 0, w, h);
     /*
      * getPixels (int[] pixels, int offset, int stride, int x, int y, int width, int height)
      * - Ritorna un array di interi dove ogni intero rappresenta un colore RGB.
      *
      * pixels: array dove si riceveranno i colori di ogni pixel della bitmap
      * offset: Il primo indice da scrivere nell' array pixels[]
      * stride: Il numero di entries da saltare tra le righe (deve essere >= bitmap.width).
      * x:  la coordinata x del primo pixel da leggere nella bitmap
      * y:  la coordinata y del primo pixel da leggere nella bitmap
      * width: numero di pixel da leggere ad ogni riga
      * height: numero di righe da leggere
      *
      */

    int index = 0;
    for(int y = 0; y < h; ++y)
    {
        for(int x = 0; x < w; ++x)
        {

            //Converte da colore ad HSV
            Color.colorToHSV(mapSrcColor[index], pixelHSV);
            //Istogramma non campionato Hue
            hHistTemp[(int)pixelHSV[0]]++;
            //Campionamento istogramma Saturation
            if(pixelHSV[1]<=0.20)
                sHist[0]++;
            else if(pixelHSV[1]>0.20 && pixelHSV[1] <= 0.40)
                sHist[1]++;
            else if(pixelHSV[1]>0.40 && pixelHSV[1] <= 0.60)
                sHist[2]++;
            else if(pixelHSV[1]>0.60 && pixelHSV[1] <= 0.80)
                sHist[3]++;
            else
                sHist[4]++;
            //Campionamento istogramma Value
            if(pixelHSV[2]<=0.20)
                vHist[0]++;
            else if(pixelHSV[2]>0.20 && pixelHSV[2] <= 0.40)
                vHist[1]++;
            else if(pixelHSV[2]>0.40 && pixelHSV[2] <= 0.60)
                vHist[2]++;
            else if(pixelHSV[2]>0.60 && pixelHSV[2] <= 0.80)
                vHist[3]++;
            else
                vHist[4]++;
```

```
            index++;
        }
    }


    for(int i = 0; i<20 ; i++)
    {
        for(int j=i*18;j<i*18+18;j++)
        {
                hHist[i]+=hHistTemp[j];
        }
    }

    System.arraycopy(hHist, 0, hsvHist, 0, hHist.length);
    System.arraycopy(sHist, 0, hsvHist, hHist.length, sHist.length);
    System.arraycopy(vHist, 0, hsvHist, hHist.length+sHist.length, sHist.length);

    for(int i =0 ; i < hsvHist.length;i++)
    {
        hsvHist[i]=hsvHist[i]/(npixels);
    }

    return hsvHist;
}
```

3.4.2 Chi-Square distance calculator

L'implementazione dell'algoritmo di distanza scelto riceve come parametri due istogrammi HSV e restituisce un valore double compreso tra 0 e 1 dove 0 indica la congruenza tra i due istogrammi ed 1 la completa diversità. La distanza Chi-Square equivale alla sommatoria delle distanze di ogni coppia di bucket degli istogrammi. La similarità si ottiene come complementare della distanza. Valutare se il grado di similarità dei due istogrammi è sufficiente, implica la scelta di una soglia che determini se la foto scattata è abbastanza simile a quella richiesta. La scelta della soglia è stata effettuata sulla base dei risultati ottenuti nei test sperimentali descritti al capitolo 4. La soglia è stata fissata al valore di similarità di 0.78, acquisito calcolando la media dei valori minimi ottenuti da match genuine per ogni dataset; in questo modo si valuteranno corretti anche match incorretti, ottenendo quindi un'alta recall ma una bassa precisione: ciò non risulterà un problema dato che la valutazione è basata anche sulla posizione GPS e l'orientamento del dispositivo nel momento di acquisizione dell'immagine.

Si riporta di seguito il codice:

```
public static double getChiSquareDistance(double[] histA,double[] histB)
{
	double result = 0;

	for(int i = 0 ; i < histA.length ; i++)
	{
		double num= Math.pow((histA[i]-histB[i]), 2);
		double den = histA[i]+histB[i]+Float.MIN_VALUE;

		result+= (Math.pow((histA[i]-histB[i]), 2))
				/(histA[i]+histB[i]+Float.MIN_VALUE);
	}

	return 1-(result/2);
}
```

Capitolo 4: Test e analisi di algoritmi di Computer Vision

Al fine di ottimizzare le prestazioni del software di riconoscimento di immagini, prima dell'implementazione dell'algoritmo per dispositivi Android, si è scelto di eseguire una fase di test garantendo così l'indipendenza dei metodi che verranno utilizzati da uno specifico dataset di immagini.

In particolare per facilitare e accelerare la fase di test si è scelto di sfruttare il software Matlab, un ambiente per il calcolo numerico e l'analisi statistica che consente di manipolare matrici in modo veloce e fortemente ottimizzato. In questo modo, sfruttando i metodi nativi del software, si sono abbattuti notevolmente i costi computazionali. I risultati raccolti verranno poi rappresentati sotto forma tabulare o di grafico per poterne dare una valutazione più accurata confrontando più facilmente gli indicatori di prestazioni sia per dataset che per metodo utilizzato. Tra i vari indicatori si è scelto di adottare quelli più comunemente utilizzati in letteratura ovvero Area Under Curve (AUC) e l'Accuracy della classificazione ottenuta mediante K Nearest Neighbors Classifier (KNN).

Al fine di avere un panorama di confronto quanto più vario possibile, si è scelto di analizzare metodi adatti al nostro caso di studio di vario genere: sia metodi semplici e computazionalmente poco costosi che metodi più avanzati i quali sfruttano descrittori locali ma che comportano un aumento del costo computazionale abbastanza consistente. Una migliori descrizione degli algoritmi testati e degli indicatori di prestazioni utilizzati verrà fornita in seguito.

4.1 Dataset utilizzati

Per quanto riguarda invece la volontà di ottenere dei dati dataset-indipendent si è scelto di testare tutti gli algoritmi su quattro dataset di immagini molto usati in letteratura:

- **Catech buildings** [23]:

  Numero di edifici/monumenti fotografati: 50

  Numero di foto per ogni edificio: 5

  Risoluzione immagini: 2048x1536

  Caratteristiche: il dataset contiene 250 foto totali in risoluzione elevata di edifici del campus del California Institute of Technology. Le foto di ogni edificio sono scattate da diversi punti di vista garantendo così una prospettiva diversa dello stesso edificio. Le

foto sono state scattate più o meno tutte nello stesso lasso di tempo per questo ogni edificio ha sempre lo stesso tipo di illuminazione. In questo modo non è possibile fare paragoni con foto scattate in un altro momento della giornata o in giornate con condizioni atmosferiche diverse.

Figura 23 - Foto di esempio per Caltech buildings

- **ZuBud** [24]:

  Numero di edifici/monumenti fotografati: 201

  Numero di foto per ogni edificio: 5

  Risoluzione immagini: 640x480

  Caratteristiche: il dataset contiene 1005 foto totali di edifici della città di Zurigo. Il pregio di questo dataset è la quantità di immagini e la varietà di tipologie di edifici ritratti in esse, inoltre presenta gli stessi pregi e difetti del dataset Caltech buildings, ovvero: uno stesso edificio è stato fotografato da più angolazioni ma sempre con la stessa illuminazione.

Figura 24 – Foto di esempio per ZuBud

- **Sheffield Building** [25]:

  Numero di edifici/monumenti fotografati: 40

  Numero di foto per ogni edificio: 50 ~ 300+

  Risoluzione immagini: 160x120

  Caratteristiche: il dataset è composto da 4.178 file in tutto, le foto ritraggono edifici da diverse angolazioni e in differenti momenti della giornata infatti sono sottoposti a diverso tipo di illuminazione. Pur contenendo immagini in bassa risoluzione, il dataset, costituisce un ottimo supporto per i test di instance recognition in quanto propone una grossa varietà di edifici. Inoltre la fase di test è agevolata in quanto la bassa risoluzione delle immagini accorcia notevolmente i tempi di esecuzione degli algoritmi. L'eterogeneità delle foto è il punto di forza di questo dataset in quanto ci propone una situazione pseudo-reale in cui le foto sono scattate da diverse angolature, in diversi momenti della giornata, in diversi giorni ecc.

Figura 25 - Foto di esempio per Sheffield buildings

- **ZuBud Objects** [24]:

  Numero di oggetti fotografati: 53

  Numero di foto per ogni oggetto: 5

  Risoluzione immagini: 320x240

  Caratteristiche: si è deciso di includere in questa fase di test anche un piccolo dataset contenente foto di oggetti piuttosto che edifici al fine di verificare se i metodi testati sono adatti anche alla classificazione di oggetti, in questo modo si può pensare di estendere il concetto di caccia al tesoro fotografica non solo ad edifici ma anche a oggetti di uso comune. Si è comunque deciso di utilizzare un dataset di dimensioni

ridotte e contenente foto con bassa risoluzione al fine di non allungare troppo il tempo richiesto dai test.

Figura 26 - Foto di esempio per ZuBud Objects

4.2 Descrittori e relative distanze

4.2.1 Color Histogram Distance

Questo algoritmo si basa sulla distanza degli istogrammi di colore delle due immagini confrontate. L'istogramma di colore di un'immagine grayscale è un istogramma nel quale per ogni tono di grigio viene riportato il numero di pixel presenti nella foto. Per immagini a colori l'istogramma si basa appunto sui tre canali di colore del sistema con il quale si è scelto di rappresentare l'immagine stessa [*Fig. 27*]. Nel caso si scelga la rappresentazione RGB [*Fig. 28*] si ha un istogramma per il canale del rosso, uno per il canale del blu e uno per il canale del verde; nel caso si utilizzi la scala HSV [*Fig. 29*] si ottiene un istogramma per la tinta, uno per la saturazione ed uno per la luminosità. Nei diversi casi di studio a volte può essere utile raggruppare i valori dei toni di colore in classi, ovvero definendo dei bucket di diversa granularità per costruire l'istogramma. Maggiore sarà il numero di bucket dell'istogramma per un dato canale, maggiore sarà la sensibilità e l'importanza che si attribuisce a quel canale per

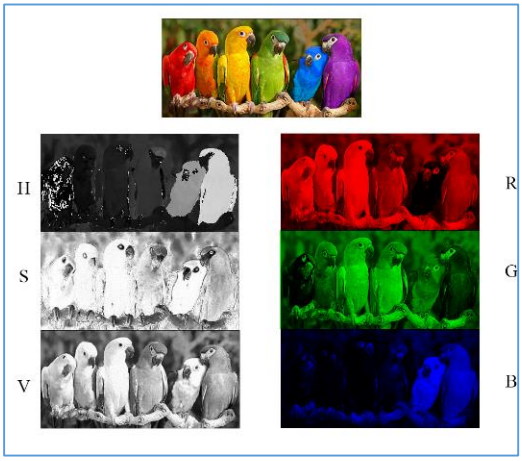


Figura 27 - A sinistra separazione dei canali HSV, a destra separazione dei canali RGB

calcolare la distanza tra le due immagini. Più in particolare, nel caso di un istogramma HSV, si può ad esempio pensare di raggruppare i valori dell'istogramma della tinta in 16 classi mentre quelli per la saturazione e per la luminosità in 4 classi ciascuna; in questo modo si ritiene più importante la gamma dei colori presenti nell'immagine, rispetto alla luminosità e alla saturazione, come parametro per attribuire un valore di similarità o distanza.

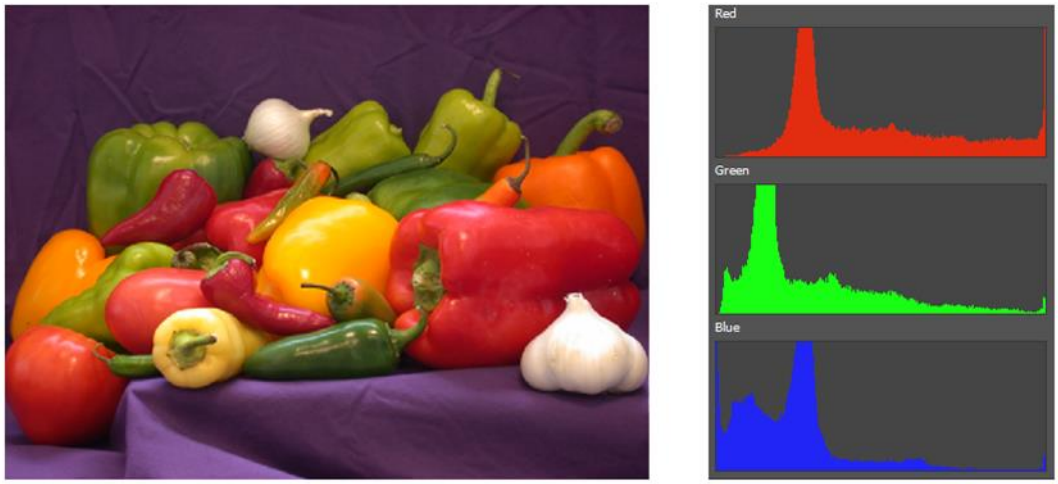


Figura 28 - Esempio di istogramma basato sulla scala cromatica RGB

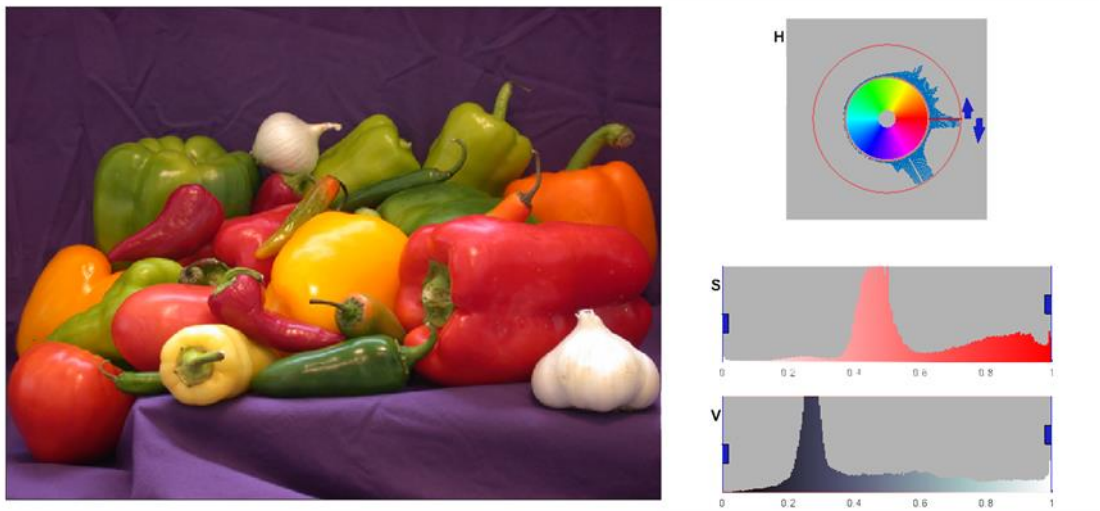


Figura 29 - Esempio di istogramma basato sulla scala cromatica HSV

Una volta calcolati gli istogrammi è semplice trovare una misura di distanza. Una molto può essere costituito dalla differenza tra i vettori degli istogrammi, questo tipo di distanza risulta essere però poco efficacie.

In letteratura sono presenti misure di distanza più elaborate che danno risultati migliori. In questi test si è scelto di adottare:

- Chi-Square Distance:

$$D(h_1, h_2) = \frac{1}{2}\sum_{k}\frac{h_1(k) - h_2(k)}{h_1(k) + h_2(k)}$$

- Intersection Distance:

$$D(h_1, h_2) = \sum_{k}\min(h_1(k), h_2(k))$$

- Bhattacharyya Distance:

$$D(h_1, h_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{h_1}\cdot\overline{h_2}\cdot n^2}}\sum_{k}\sqrt{h_1(k)\cdot h_2(k)}}$$

- Match Distance:

$$D(h_1, h_2) = \sum_{k}|H_{CS1}(k)| - |H_{CS2}(k)| \qquad \text{dove} \qquad H_{cs}(i) = \sum_{k=1}^{i} h(k)$$

Dove $h_1(k)$ rappresenta il valore del k-esimo bucket dell'istogramma costruito per la prima immagine e $h_2(k)$ rappresenta il valore del k-esimo bucket dell'istogramma calcolato per la seconda immagine. Mentre n è il numero di bucket degli istogrammi e H_{CS} sono gli istogrammi delle somme cumulative degli istogrammi di partenza.

4.2.2 Istogramma di gradienti orientati (HOG)

Sono un tipo di descrittori che contano le occorrenze dei valori di orientamento del gradiente in una porzione dell'immagine. La teoria su cui si basa questo metodo prevede che le sembianze e le forme degli oggetti locali di un'immagine possano essere descritti dalla distribuzione delle intensità dei gradienti o dei bordi all'interno dell'immagine stessa. L'implementazione [*Fig. 30*] di questo metodo prevede la segmentazione dell'immagine in più porzioni chiamate celle, per ogni cella viene calcolato l'istogramma delle direzioni dei gradienti; la combinazione degli istogrammi rappresenta il descrittore. Per aumentarne le prestazioni è opportuno regolare il contrasto delle immagini (operazione di contrast stratching).

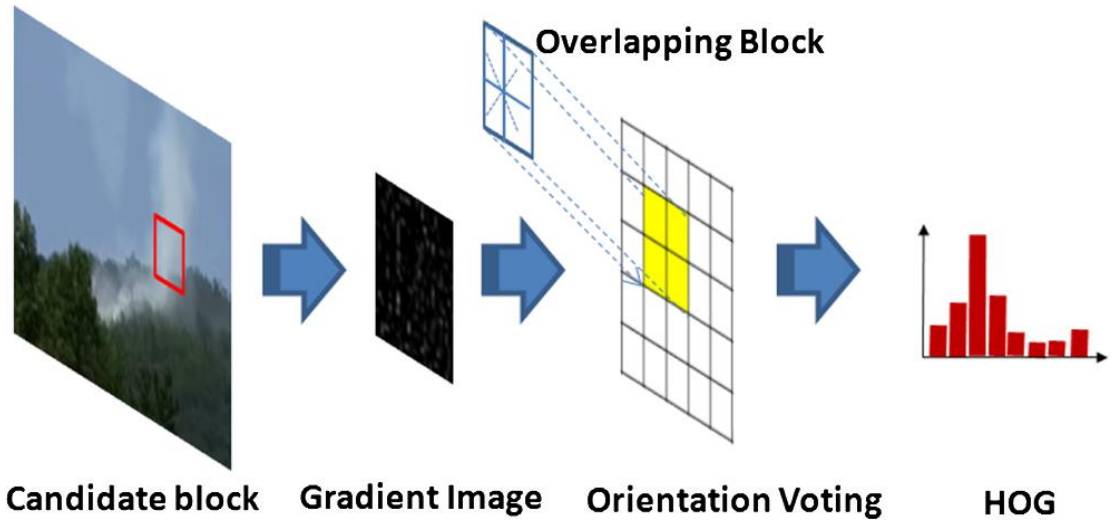


Figura 30 - Esempio di estrazione di istogramma HOG in una porzione di immagine

Per questo tipo di features non sono adatte le distanze tra istogrammi comunemente utilizzate, nel nostro caso si è scelto di utilizzare i seguenti metodi di distanza:

- Somma dei prodotti:

$$D(h_1, h_2) = \sum_k h_1(k) \cdot h_2(k)$$

- Somma delle differenze:

$$D(h_1, h_2) = \sum_k h_1(k) - h_2(k)$$

4.2.3 Scale-Invariant Features

Pubblicato da David Lowe [26] è un algoritmo che permette di individuare ed estrarre delle features locali in immagini [*Fig. 31*]. Affinché questo tipo di algoritmo sia efficacie è necessario che le features estratte dalle immagini siano individuabili anche se l'immagine viene esposta a cambiamenti di luminosità, scala e rotazione o sottoposta all'aggiunta di rumore. Per fare ciò si selezionano come punti salienti su cui operare l'estrazione dei descrittori regioni dell'immagine in cui ci sono forti variazioni di contrasto come ad esempio i bordi. Un altro fattore che permette all'algoritmo SIFT di cadere in errore è dato dal fatto che questo metodo individua ed estrae una grande quantità di descrittori; in questo modo viene ridotto il fattore di errore dovuto a cambiamenti locali delle immagini. Queste sue caratteristiche rendono SIFT un algoritmo robusto per variazioni di scala e rotazione e in misura minore per trasformazioni affini. L'algoritmo SIFT essenzialmente trasforma un'immagine in un grande insieme di vettori di features che hanno proprietà simili ai neuroni della corteccia temporale inferiore, i quali sono impiegati nel riconoscimento degli oggetti nei processi cognitivi dei primati.

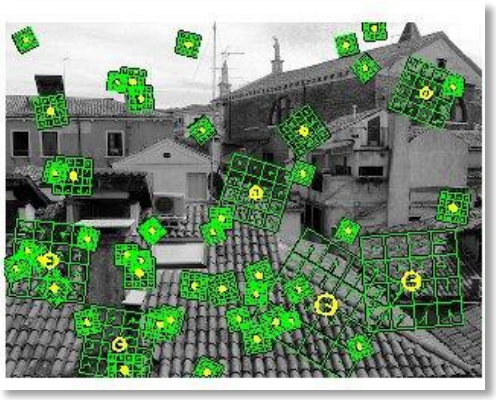


Figura 31 - Rappresentazione grafica di come vengono estratti i descrittori SIFT da un'immagine

L'algoritmo si divide in due fasi:

- **Fase 1 Keypoints detection ed estrazione dei feature vector*:*** le locazioni dei punti salienti sono definite come minimo e massimo del risultato della differenza di una funzione Gaussiana applicata, in uno spazio scalare, ad una serie di immagini ricampionate. Le regioni di basso contrasto vengono scartate e le regioni con orientamento del gradiente dominante vengono individuate come key point. Quando i punti salienti sono stati individuati vengono considerati pixel nel raggio attorno al key point e per questi si calcola la mappatura dell'orientamento dei gradienti; l'insieme di questi costituisce i features vectors.
- **Fase 2 Features matching e indicizzazione*:*** la fase di indicizzazione consiste nel raccogliere i feature vectors confrontando ogni singolo descrittore con quelli della nuova immagine per determinare quali di questi faranno match. Lowe ha utilizzato una variante dell'algoritmo euristico k-d-tree chiamato Best-bin-first search method che è in grado di identificare il nearest neighbors (ovvero il vicino più simile) con una

probabilità di successo molto alta utilizzando solo un numero limitato di computazioni. La distanza tra i vettori dei descrittori è calcolata con la distanza Euclidea. L'affidabilità che un match sia corretto può essere determinata prendendo un valore di soglia per il rapporto tra le distanze del primo miglior match e il secondo miglior match. Lowe ha stimato che utilizzando una soglia maggiore di 0.8 vengono eliminati il 90% dei falsi match perdendo solamente il 5% dei match corretti.

Il test che è stato condotto per questo algoritmo si è avvalso dell'uso della libreria VLFeat per matlab che implementa efficientemente questo metodo in particolare:

La funzione vl_sift individua i key point tramite la tecnica di Harris corner detection; questa tecnica non rispecchia del tutto le specifiche proposte da Lowe ma può essere considerata del tutto compatibile infatti la stragrande maggioranza dei keypoints corrisponde esattamente, la posizione del centro dell'89% dei keypoint individuati con i due metodi è identica considerando una precisione di 0.01 pixel [*Fig. 32*].

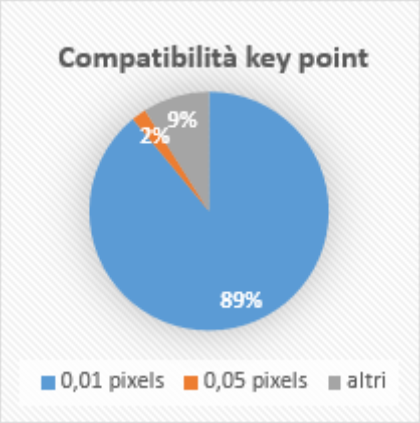


Figura 32 - L'areogramma mostra la similarità tra i keypoint individuati con l'algoritmo di Lowe e con quelli individuati con l'algoritmo implementato nella libreria VLFeat

La stessa cosa vale anche per i descrittori che risultano essere quasi altrettanto simili. Meno del 60% dei descrittori differisce per meno del 10% [*Fig. 33*].

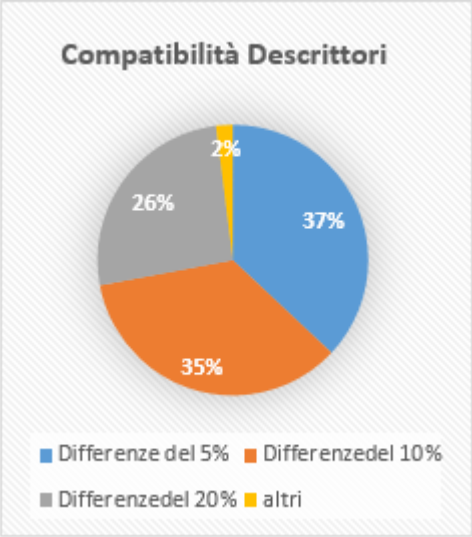


Figura 33 - L'areogramma raffigura la le percentuali per le quali i descrittori estratti con il metodo di Lowe ed il metodo utilizzato nella libreria VLFeat differiscono

Il metodo vl_ubcmatch invece implementa un modo fare il match tra due insiemi di SIFT descriptor vectors, l'algoritmo utilizza la distanza Euclidea quadratica per fornire un valore numerico di distanza. Per ogni descrittore vl_ubcmatch individua quello a lui più vicino nella seconda immagine quindi lo salva in un vettore. I match (ovvero le coppie di punti "corrispondenti") possono anche essere filtrati passando un parametro alla funzione che specifica una soglia per la quale il match viene scartato o meno (di default il valore di soglia è impostato a 1.5). Il metodo di filtraggio implementato è quello suggerito da Lowe: un descrittore D1 fa match con un

descrittore D2 solo se la distanza tra D1 e D2 moltiplicata per il valore di soglia non è maggiore della distanza tra D1 e tutti gli altri descrittori.

L'algoritmo vl_ubcmatch restituisce un vettore che contiene le distanze euclidee quadratiche per ogni coppia di key points che hanno fatto match, si tratta di una tecnica di matching locale che non fornisce di per sé un valore di similarità tra le immagini di partenza. E' quindi necessario definire una misura di distanza che possa quantificare numericamente la distanza tra le immagini, in questo studio si sono testati diversi metodi:

- Un primo metodo riordina il vettore delle distanze euclidee in modo crescente (mettendo le coppie di match migliori all'inizio del vettore) e considera la media dei primi n match dove n è un valore parametrizzabile che deve essere ottimizzato.

$$AVGdistance = \frac{1}{n}\sum_{i=1}^{n} matches_i$$

- Un secondo metodo calcola il rapporto tra il numero di match ottenuti e il minimo tra il numero di key point della prima immagine e il numero di keypoint della seconda immagine. In questo caso il parametro da ottimizzare è la soglia che viene impostata nella fase di match per determinare se un match è ambiguo o meno.

$$NMATCHdistance = \frac{|matches|}{min(|keypoints_a|, |keypoints_b|)}$$

4.2.4 RANdom Sample Consensus (RANSAC)

Il criterio di matching considerato sopra non tiene conto della posizione dei keypoint in fase di matching, ma considera solo il loro descrittore. Un criterio più evoluto di confronto consiste nel cercare la trasformazione affine che permette di sovrapporre i 2 insiemi di punti che fanno match, minimizzando un funzione di costo al variare dei parametri di roto-traslazione che determinano tale trasformazione. Per ottimizzare i parametri di tale modello si è usato l'algoritmo RANSAC, ovvero un metodo iterativo per stimare i parametri di un modello matematico da un insieme di dati che contiene degli outliers[1]. RANSAC è un algoritmo non deterministico in quanto produce un risultato corretto solo con una certa probabilità. L'algoritmo è stato pubblicato per la prima volta da Fischler e Bolles.

[1] Per outliers di un inseme numerico si intende un sottoinsieme di valori sporadici che si discostando dal trand generale dei valori dell'insieme. Viceversa gli inliers sono i valori che non si discostano dal trand.

Secondo il metodo RANSAC un insieme di valori può essere quindi diviso in due sottoinsiemi distinti: il sottoinsieme degli inliers, dati la cui distribuzione può essere spiegata da un modello parametrico, e gli outliers, dati che non rientrano nel modello e possono essere stati generati da errori di vario genere.

L'algoritmo procede nel modo seguente:

i. Seleziona un sottoinsieme random dei dati iniziali chiamato *sottoinsieme degli inliers ipotetici*.
ii. Viene generato un modello per mappare il sottoinsieme del punto 1.
iii. Tutti gli altri dati sono testati per verificare se sono conformi al modello. Ogni punto che risulta essere conforme al modello viene anch'esso considerato inlier ipotetico.
iv. Il modello viene quindi ricalibrato considerando il nuovo insieme di inliers ipotetici.
v. Il modello viene valutato attraverso una stima dell'errore che si commette associando agli inliers ipotetici il modello.

Il metodo viene reiterato un numero predefinito di volte, ogni volta si genere un nuovo modello ed un nuovo gruppo di ipotetici inliers, nel caso in cui il valore di valutazione sia migliore di quelli considerati in precedenza il modello viene considerato ottimo e gli altri vengono scartati [*Fig. 34*].

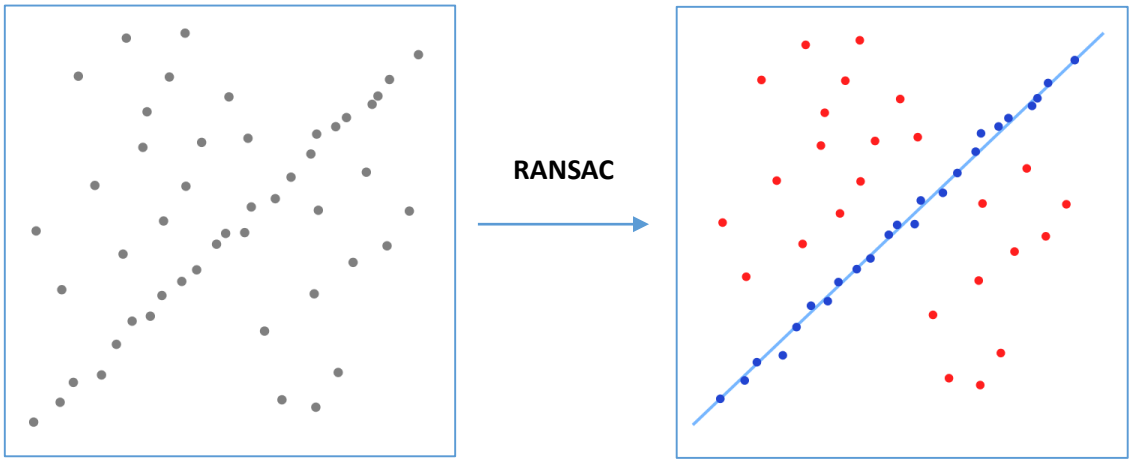


Figura 34 - L'esempio riportato qui di sopra rappresenta graficamente come opera l'algoritmo RANSAC in un insieme di dati in uno spazio bidimensionale. Dopo un numero n di iterazioni il sottoinsieme degli inliers ipotetici sarà costituito dai pallini blu che ovviamente costituiscono un'ottima valutazione del modello che quindi verrà considerato come ottimo

RANSAC risulta essere quindi un metodo efficace e adatto a filtrare i match generati da un algoritmo di riconoscimento di immagini basato su descrittori locali. Nel nostro caso viene considerato l'insieme dei match generati dall'algoritmo SIFT [Fig. 36].

In particolare si consideri come insieme di partenza l'insieme di tutti i match prodotti dalla funzione vl_ubcmatch:

i. L'algoritmo seleziona un sottoinsieme random di quello di partenza, appunto il sottoinsieme di ipotetici inliers.
ii. Viene stimata una trasformazione affine della seconda immagine per la quale si tenta di sovrapporre nel migliore dei modi i key point che fanno match appartenenti al sottoinsieme degli inliers ipotetici.
iii. Quindi si verifica se la restante parte dei match, che non fanno parte dell'insieme degli inliers ipotetici, è conforme alla trasformazione affine generata nel punto precedente. Nel caso siano conformi il sottoinsieme viene arricchito con le nuove coppie di key points che fanno match.
iv. La trasformazione affine viene quindi ricalcolata basandosi sul nuovo sottoinsieme di match.
v. Viene fornita una valutazione numerica qualitativa della trasformazione affine prodotta nel punto precedente. Nel caso il valore sia migliore la trasformazione affine viene considerata ottima e quelle calcolate in precedenza vengono scartate.

Essendo un algoritmo non deterministico dove la scelta del sottoinsieme dei match è casuale, il processo viene ripetuto K volte al fine di aumentare la probabilità di ottenere la trasformazione affine migliore.

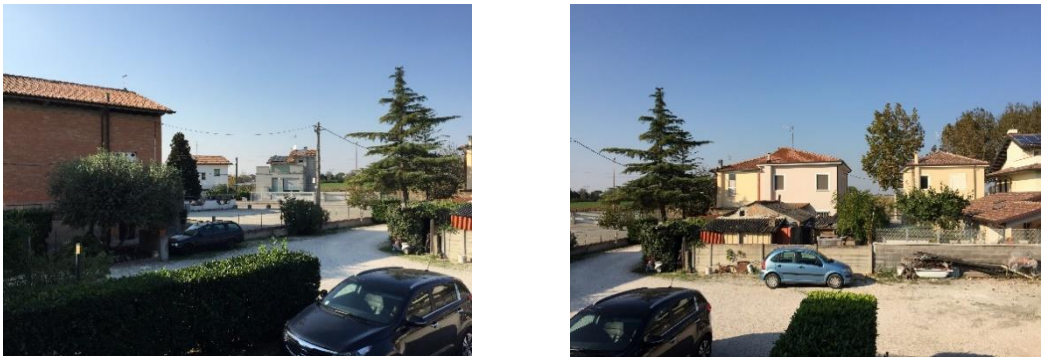


Figura 35 - Immagini di partenza per l'esecuzione dell'algoritmo RANSAC

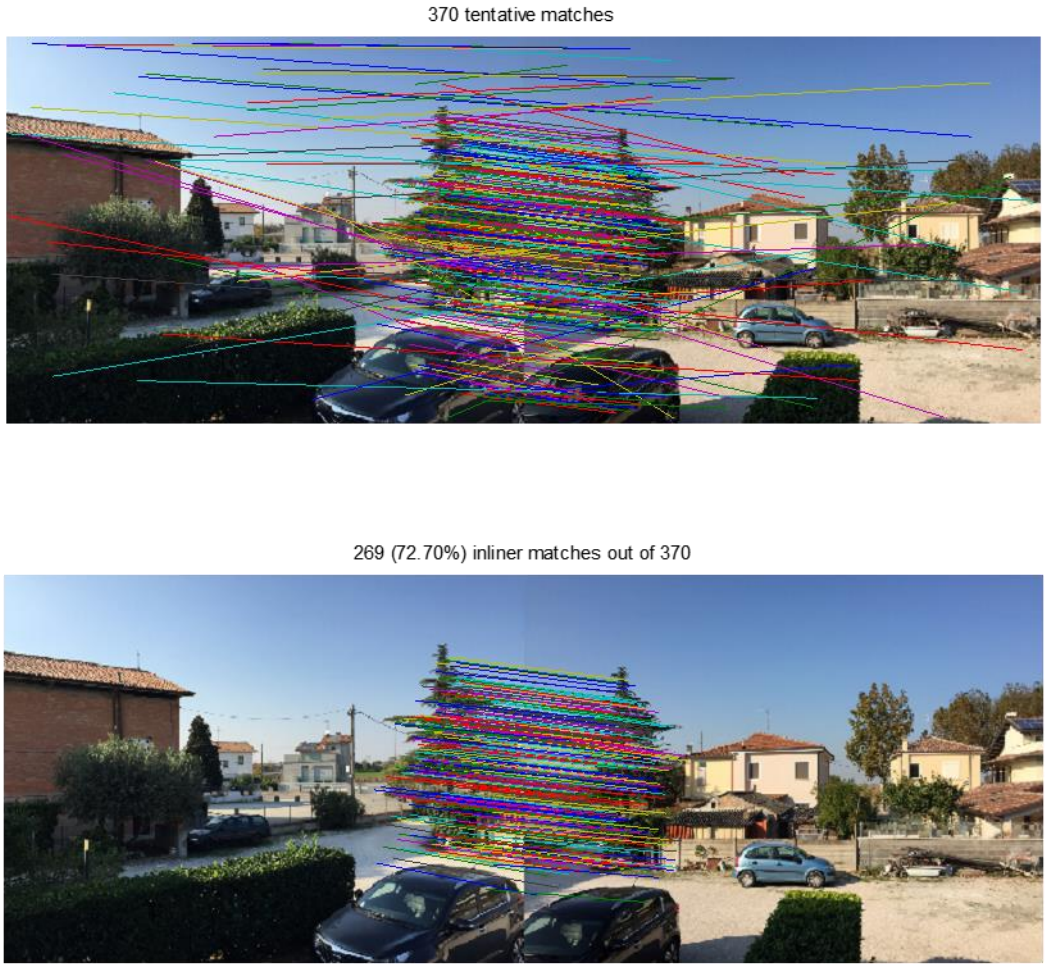


Figura 36 - Nella figura sopra è possibile vedere il risultato dei match generati grazie ai SIFT descriptors tramite la funzione vl_ubcmatch, nella figura sotto è possibile notare come dopo la computazione dell'algoritmo RANSAC siano stati rimossi tutti i match outlier, si noti che la percentuale dei match rimossi è circa del 30%

Ora è necessario poter quantificare numericamente quanto sono simili le due immagini dopo la computazione dell'algoritmo RANSAC; per fare questo sono stati valutati 3 diversi metodi di distanza:

- Si calcola il rapporto tra il numero di match che fanno parte dell'insieme finale degli inliers ipotetici prodotto dopo K computazioni di RANSAC e il numero di match prodotti dall'algoritmo SIFT implementato dalla funzione vl_ubcmatch.

$$N\_INLIERSdistance = \frac{|\text{match}_{\text{inliers}}|}{|\text{match}_{\text{SIFT}}|}$$

- Si considera la media delle distanze euclidee quadratiche dei match che fanno parte del sottoinsieme degli inliers ipotetici prodotto da RANSAC dopo K computazioni.

$$AVG\_INLIERSdistance = \frac{\sum \text{inliers}}{|\text{inliers}|}$$

- Partendo dal presupposto che le immagini sulle quali viene calcolato l'algoritmo abbiano lo stesso orientamento (come avviene nei dataset che sono stati utilizzati); un terzo tipo di distanza che si è deciso di calcolare nasce dal fatto che, per due immagini simili, l'insieme delle linee che congiungono le coppie di key points che hanno fatto match non dovrebbero intersecarsi. Questo dovrebbe essere vero soprattutto se dall'insieme dei match vengono scartati tutti i match outlier. Per questo risulta essere ragionevole la distanza che calcola il rapporto tra numero di intersezioni tra le linee che si ottengono e numero di intersezioni possibili.

$$INTERSECTIONdistance = \frac{|\text{intersections}|}{|\text{possible intersections}|} = \frac{|\text{intersections}|}{\frac{(|\text{inliers}| - 1)|\text{inliers}|}{2}}$$

$$= 2\frac{|\text{intersections}|}{(|\text{inliers}| - 1)|\text{inliers}|}$$

4.3 Indicatori di prestazione

4.3.1 Accuracy usando K-Nearest-Neighbors

Il k-nearest neighbors (KNN) è un algoritmo che classifica un oggetto basandosi sulle caratteristiche dei K oggetti classificati come più simili ad esso.

L'oggetto in esame viene classificato in base alla classe che appare maggiormente tra i K oggetti più simili ad esso [*Fig. 37*]. La scelta del parametro K dipenderà dalle caratteristiche dei dati presi in esame. Nel nostro caso il parametro varierà da dataset a dataset e sarà pari al numero di immagini per classe.

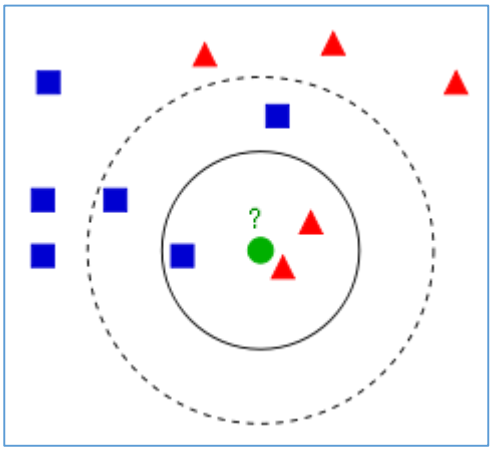


Figura 37 - - Esempio grafico della classificazione tramite KNN, per k=3, l'oggetto preso in esame sarà classificato come triangolo vista la superiorità della classe negli oggetti a lui più vicini mentre per k=5 l'oggetto sarà classificato come quadrato

Nel nostro caso si utilizza l'accuracy, ovvero la percentuale di immagini correttamente classificate, per valutare le performance del classificatore KNN.

4.3.2 Receiver Operator Characteristic (ROC)

La curva Receiver Operator Characteristic (ROC) è una rappresentazione grafica della qualità di un classificatore binario. Il metodo è molto intuitivo ed espressivo.

Immaginiamo di trovarci nella situazione graficamente rappresentata [*Fig. 38*]. Classifichiamo i due insiemi A e B rispettivamente insieme dei malati e insieme dei sani e prendiamo in esame la popolazione U composta dai due insiemi.

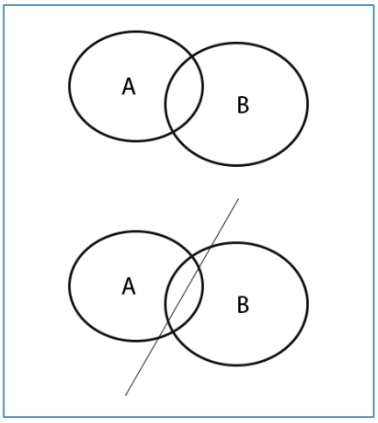


Figura 38 - Rappresentazione grafica dell'intersezione delle caratteristiche di due insiemi di classi opposte

Abbiamo a disposizione un test che taglia in due parti la popolazione U cercando di separare i malati dai sani.

Purtroppo questa operazione nella maggior parte dei casi, come nel nostro, non sarà in grado di dividere l'insieme A dall' insieme B perché le "feature" osservate non sono sufficientemente tipiche del fenomeno che si vuole distinguere. Sarà quindi opportuno introdurre nel classificatore un errore causato dalla sovrapposizione nello spazio dei due insiemi delle due classi.

Stabilita una soglia s, i casi che possono presentarsi sono 4:

i. x>s e il paziente è sano. Questo caso si chiama "vero positivo" (TP).
ii. x>s e il paziente è malato. Questo caso si chiama "falso positivo" (FP).
iii. x<s e il paziente è malato. Questo caso si chiama "vero negativo" (TN).
iv. x<s e il paziente è sano. Questo caso si chiama "falso negativo" (FN).

Per poter condurre un'analisi appropriata è comune normalizzare le popolazioni A e B dividendo ogni valore ottenuto per il numero totale di elementi del sottoinsieme di appartenenza.

	Situazione Reale	
	A	**B**
x<s	1900	50
x>s	100	250

→

	Situazione Normalizzata	
	A	**B**
x<s	0.95	0.16
x>s	0.05	0.84

Figura 39 - Normalizzazione dei risultati ottenuti con soglia s

Dai valori normalizzati è ora possibile calcolare diversi indicatori quali:

Precisione = $\frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FP})}$

Sensibilità = $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$

Accuratezza = $\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FN}+\mathrm{FP}} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{U}}$

Specificità = $\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$

Supponiamo ora di scegliere una nuova soglia s'>s.

In questo caso il valore normalizzato TN salirebbe possibilmente fino ad 1 avendo una massima specificità.

Contemporaneamente però aumenterebbe il valore FN.

Scegliendo invece un ulteriore nuovo valore di soglia s''<s si otterrebbe l'effetto contrario.

	Soglia s'	
	A	**B**
x<s	1	0.24
x>s	0	0.76

	Situazione Normalizzata	
	A	**B**
x<s	0.85	0
x>s	0.15	1

Figura 40 - Valori ottenibili variando le soglie

La curva ROC [*Fig. 41*] si ottiene "graficando" le performance dello schema di classificazione su un diagramma bidimensionale al variare della soglia selezionata.

L'asse delle ascisse x corrisponde al valore (1-Accuratezza) e l'asse delle ordinate y al valore di sensibilità.

In altri termini ogni punto (x,y) della curva ROC ottenuta è calcolato come segue:

$$(x, y) = \left(1 - \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}, \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}\right)$$

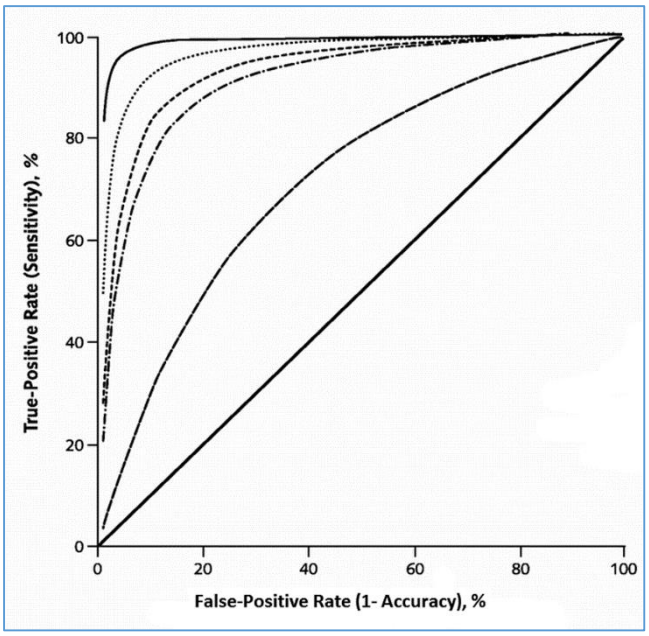


Figura 41 - ROC curve

Per ogni valore di s quindi corrisponderà un punto nel grafico.

La curva risultante da tutti i possibili valori di soglia scelti si dice ROC del sistema dati/classificatore.

Un parametro di rilevante importanza stabilito dalla curva ROC è ottenibile calcolando l'area sottesa dalla curva (AUC). L'area ottenuta rappresenta la probabilità di un elemento a caso tratto dalla popolazione di essere classificato correttamente.
Notare che ogni soglia identifica la sua probabilità di classificare correttamente un elemento preso a caso dalla popolazione [*Fig. 42*].
L'area è calcolabile scomponendo in due triangoli di base 1 il quadrilatero ottenuto:

$$Area = \frac{1}{2}\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} + \frac{1}{2}\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$$

Questa osservazione ci consente di dire che il valore di soglia migliore è identificato dal punto della curva ROC più distante alla diagonale principale [*Fig. 42*].

Se il punto si trova sulla diagonale principale, l'area sarà pari a $\frac{1}{2}$ e quindi, data la natura binaria del classificatore, la sua performance sarà equivalente ad una scelta casuale.

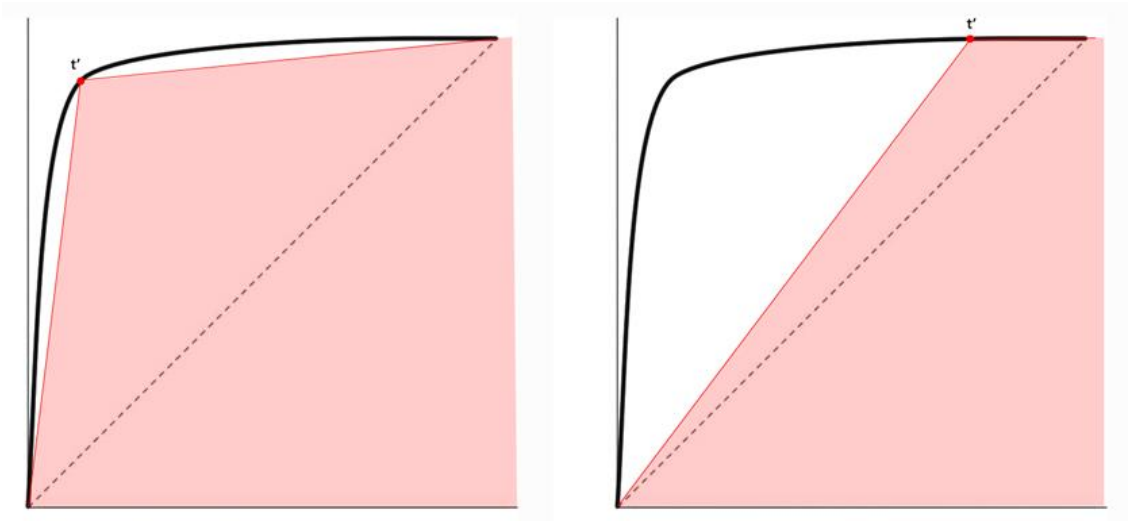


Figura 42 - Differenza di probabilità di corretta classificazione al variare della soglia t. Più t è distante dalla diagonale principale, più la probabilità di classificare correttamente un elemento della popolazione è maggiore.

4.3.3 Grafico genuine/impostor

Il grafico genuine/impostor rappresenta sullo stesso piano la densità delle distanze ottenute da match genuine e la densità delle distanze ottenute da match impostor.

Per distanza ottenuta da un match tra due immagini si intende uno scalare compreso tra 0 e 1 dove 1 indica che le due immagini sono congruenti e 0 incongruenti. Per match genuine si intende un confronto su due immagini della stessa classe di appartenenza mentre per match impostor un confronto su due immagini di diversa classe di appartenenza.

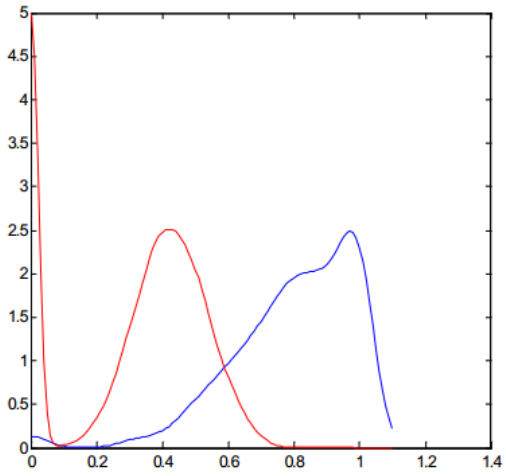


Figura 43 - In rosso la densità dei match impostor e in blu la densità dei match genuine

Il grafico genuine/impostor rappresenta i risultati della classificazione in modo intuitivo ed espressivo individuando l'intersezione delle due densità. Per massimizzare l'AUC del sistema la soglia dovrà esser scelta nell'intorno del punto di intersezione delle due curve, ma sono possibili altre scelte dipendenti dai requisiti di "sicurezza" del sistema. La scelta della soglia dipende dalla natura dell'applicazione del fenomeno che si vuole classificare, per esempio nel caso di classificazione di impronte digitali, solitamente si usa tenere la soglia molto alta in modo da scartare ogni possibile impostor classificando quindi come impostor anche casi genuine al fine di garantire la massima sicurezza.

Prendendo in esame il grafico genuine/impostor illustrato [*Fig. 43*] nel caso si scegliesse come soglia 0.6, si classificherebbero come genuine una piccola parte dei match impostor ma si classificherebbero impostor una elevata quantità di match che in realtà sarebbero genuine.

4.4 Risultati Sperimentali

Qui di seguito verranno proposti i dati raccolti testando i metodi di instance recognition presi in esame sui dataset di immagini descritti in precedenza. Al fine di consentire una miglior analisi di performance ed efficienza, i dati stessi, verranno proposti sotto forma tabulare o di grafico in modo da mettere a confronto i vari metodi e stimare per quali parametri alcuni di essi sono più robusti di altri come ad esempio variazione di luminosità, contrasto, punti di prospettiva differenti, ecc.

È inoltre doveroso fare presente che i dati verranno analizzati facendo una suddivisione per dataset, in quanto, in questo modo, si potranno fare valutazioni in base alle caratteristiche stesse delle immagini raccolte.

AUC		DATASET			
Descrittore	Distanza	Caltech Buildings	ZuBud	Sheffield Buildings	ZuBud Objects
HSV Histogram	Chi-Square	0,9322	0,9680	0,8102	0,8454
	Bhattacharyya	0,9362	0,9672	0,8107	0,8456
	Match	0,9147	0,9452	0,7294	0,8427
	Intersection	0,9189	0,9675	0,8050	0,8439
HOG Histogram	Somma Prodotti	0,8122	0,8608	0,8169	0,7537
	Somma Differenze	0,8652	0,8928	0,8389	0,7671
SIFT	AvgMatch [n=5]	--	--	0,8789	--
	AvgMatch [n=10]	--	--	0,7931	0,7540
	AvgMatch [n=15]	--	--	0,7145	0,7736
	AvgMatch [n=20]	--	--	0,6574	0,7825
	AvgMatch [n=25]	--	--	0,6173	0,774
	AvgMatch [n=30]	--	--	--	0,7550
	AvgMatch [n=35]	--	--	--	--
	AvgMatch [n=40]	0,9449	0,9724	--	--
	AvgMatch [n=45]	0,9438	0,9731	--	--
	AvgMatch [n=50]	0,9394	0,9738	--	--
	AvgMatch [n=55]	0,9319	0,9730	--	--
	AvgMatch [n=60]	0,9210	0,9705	--	--
	NMatch/ NKeypoints	0,9266	0,9631	0,8640	0,8001
SIFT+RANSAC	AvgInliers	0,8242	0,8763	--	0,6511
	NInlier/Nmatch	0,5018	0,3753	--	0,6405
	NIntersections	0,9087	0,9382	--	0,6686

Tabella 1 - Valori di AUC ottenuti

Accuracy with K-NN		DATASET			
Descrittore	Distanza	Caltech Buildings	ZuBud	Sheffield Buildings	ZuBud Objects
HSV Histogram	Chi-Square	68,80%	75,82%	96,86%	53,96%
	Bhattacharyya	67,60%	75,52%	96,52%	55,85%
	Match	56,80%	54,03%	93,99%	55,85%
	Intersection	58,80%	71,54%	96,10%	49,81%
HOG Histogram	Somma Prodotti	32%	46,67%	95,48%	16,98%
	Somma Differenze	48%	58,61%	96,86%	21,51%
SIFT	AvgMatch [n=5]	--	--	99,14%	--
	AvgMatch [n=10]	--	--	98,99%	10,57%
	AvgMatch [n=15]	--	--	98,49%	10,19%
	AvgMatch [n=20]	--	--	97,10%	13,58%
	AvgMatch [n=25]	--	--	93,51%	19,25%
	AvgMatch [n=30]	--	--	--	27,17%
	AvgMatch [n=35]	--	--	--	--
	AvgMatch [n=40]	75,60%	78,21%	--	--
	AvgMatch [n=45]	78,00%	80,10%	--	--
	AvgMatch [n=50]	77,20%	81,59%	--	--
	AvgMatch [n=55]	76,80%	82,99%	--	--
	AvgMatch [n=60]	77,60%	84,18%	--	--
	NMatch/ NKeypoints	74,40%	87,46%	93,51%	43,02%
SIFT+RANSAC	AvgInliers	10,80%	11,74%	--	2,26%
	NInlier/Nmatch	14,40%	7,76%	--	7,55%
	NIntersections	87,60%	89,65%	--	34,64%

Tabella 2 - Valori di accuracy usando l'algoritmo K-NN con k=5

Nelle tabelle sono riportati i valori di AUC e di accuracy calcolata tramite K-Nearest Neighbors con valore di K pari a 5; si è scelto di utilizzare questo valore in quanto la maggioranza dei dataset presi in esame avevano esattamente 5 foto per ogni "classe", di conseguenza utilizzare un valore più elevato di 5 è concettualmente errato in quanto non si potranno mai classificare correttamente più di 5 immagini.

Fatte queste considerazioni è opportuno sottolineare che, per il dataset Sheffield Building dove la cardinalità delle classi di immagini è variabile e dove questa non è mai inferiore a 60 circa, il valore dell'accuracy sarà molto più elevato in quanto per ogni classe sono presenti un parte immagini molto simili tra di loro, dove l'angolo di inquadratura varia di pochi gradi.

Si consideri inoltre che per quanto riguarda la distanza *AvgMatch,* ovvero la distanza ottenuta come media dei primi n valori di score tramite match di SIFT features, si è scelto di

rappresentare per ogni dataset un giusto range di valori di n in funzione della risoluzione delle immagini del dataset stesso; l'algoritmo SIFT estrae più features quanto più le immagini sono grandi, per questo fare la media dei primi 50 match su immagini piccole come quelle di Sheffield Buildings risulta essere privo di significato.

In fine, data la complessità computazionale dell'algoritmo RANSAC, si è deciso di non calcolare queste distanze per il dataset di immagini Sheffield building in quanto avrebbe richiesto un tempo elevato data la potenza di calcolo in possesso e la grossa mole di dati da analizzare. Si rimanda a studi futuri il calcolo e la valutazione dei suddetti valori.

4.5 Valutazione dei dati sperimentali

Qui di seguito sono riportati i risultati sotto forma di grafici separandoli per i vari dataset su cui sono stati testati gli algoritmi. Al fine di rendere più leggibili le informazioni, per quanto riguarda il confronto tra le curve ROC, si è scelto di fare una selezione delle curve più significative per ogni algoritmo di classificazione di immagini. In particolare si è deciso di avere un ampio spettro di visione tra le diverse misure di distanza: per quanto riguarda le features basate sugli istogrammi si è scelto di rappresentare graficamente una sola misura di distanza (ovvero la Chi-Square distance per l'istogramma HSV e la somma dei prodotti per HOG) in quanto i valori ottenuti con le diverse distanze sono abbastanza equivalenti. Per quanto riguarda invece i risultati ottenuti con le diverse distanze basate su descrittori locali, si è scelto di fare una minore selezione in quanto le curve ottenute con i diversi metodi sono molto eterogenee (l'unica eccezione è stata fatta per la distanza ottenuta come media dei valori con le SIFT features, in questo caso si è scelto infatti di mantenere la curva che rappresentava il valore di AUC migliore).

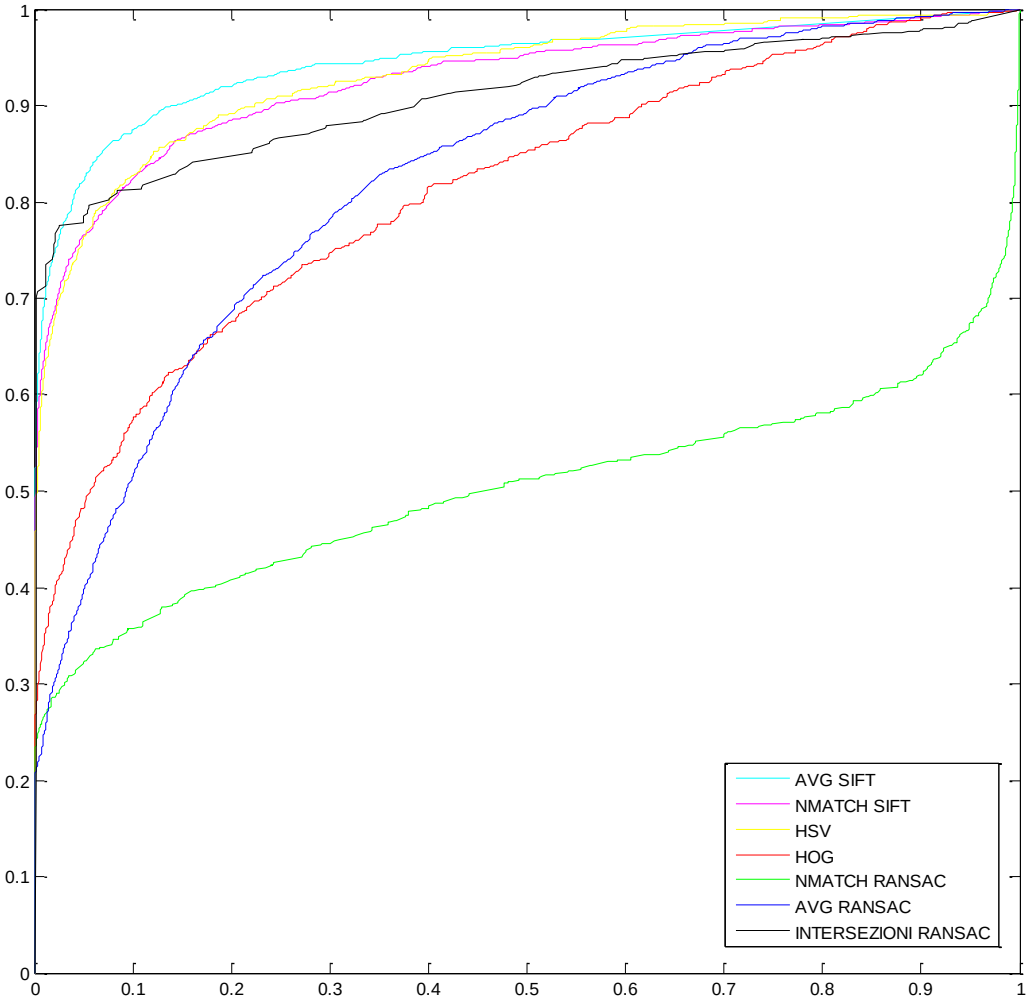


Figura 44 - Rappresentazione delle curve ROC a confronto ottenute sul dataset Caltech Buildings

In questo grafico sono rappresentate le curve ROC più significative per ogni metodo testato sul dataset di immagini Caltech Buildings in modo da facilitarne il confronto. Si intuisce che la curva ROC migliore è rappresentata dalla distanza ottenuta con il metodo di classificazione basato sulle SIFT features (curva azzurra) che utilizza come misura di distanza la media dei primi *n* valori degli score di match. In questo caso la misura migliore viene ottenuta facendo la media dei primi 40 valori di score.

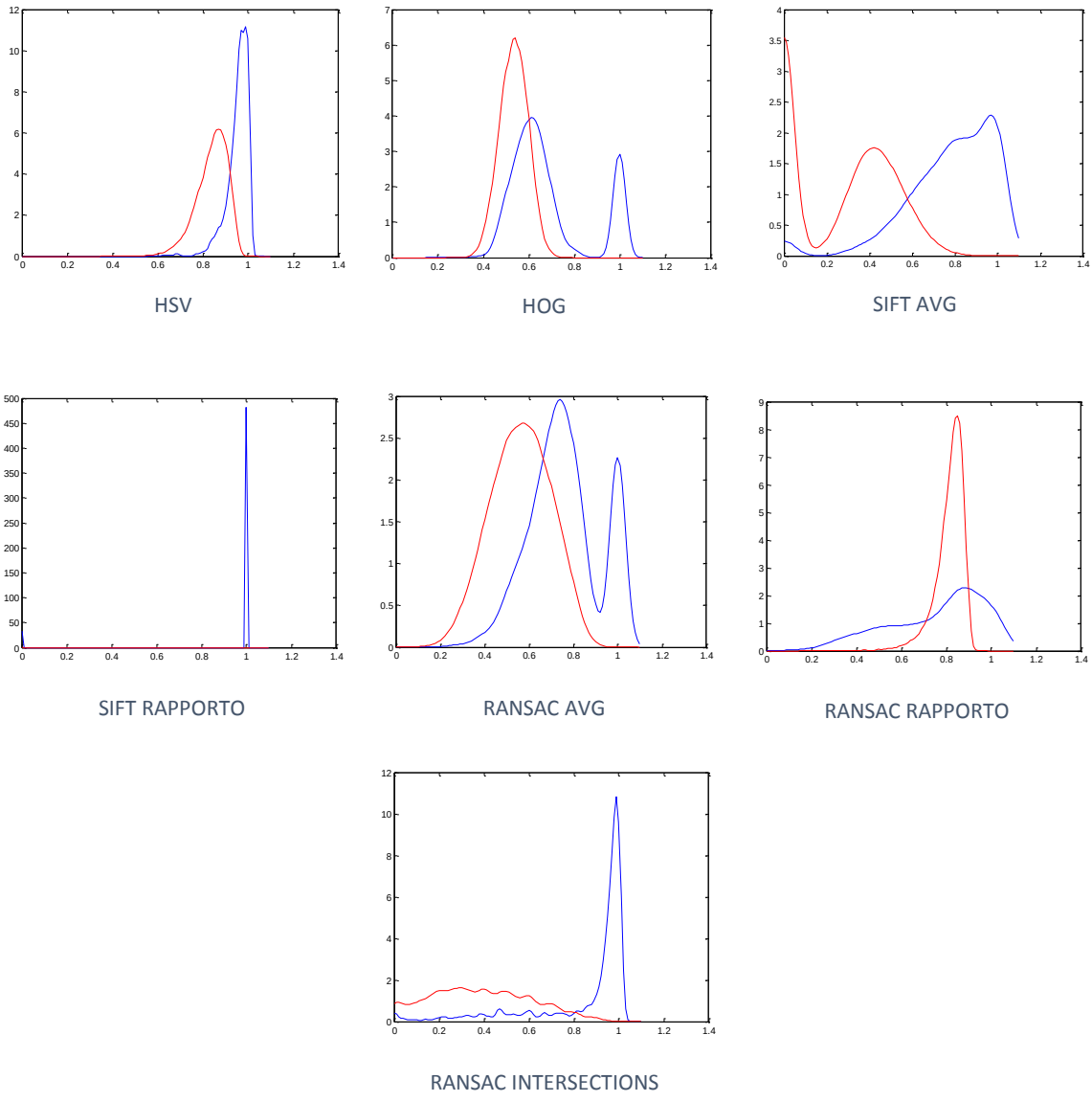


Figura 45 - Rappresentazione dei grafici Genuine/Impostor per il dataset Caltech Buildings

Apparentemente secondo questi grafici risultano essere delle buone misure di distanza quelle ottenute tramite il rapporto tra intersezioni ottenute e intersezioni possibili utilizzando le SIFT features raffinate tramite RANSAC. Per quanto riguarda le misure di distanza ottenute con algoritmi basati su features globali come gli istogrammi si nota che la Chi-Square distance fornisce risultati nettamente migliori rispetto alle distanze ottenute con HOG.

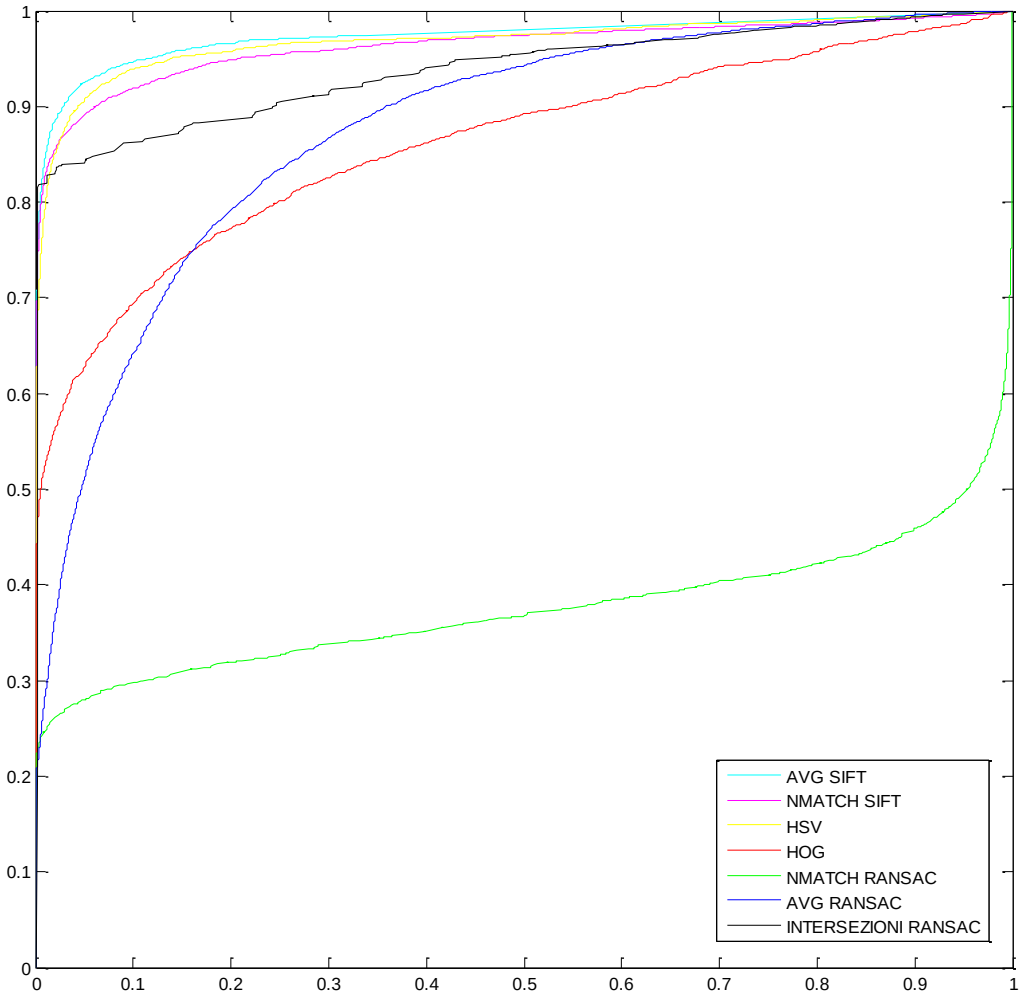


Figura 46 - Rappresentazione delle curve ROC a confronto ottenute sul dataset Zubud

Anche in questo caso il risultato migliore è ottenuto tramite la media degli score dei primi n match usando le SIFT features (in questo caso il risultato migliore è dato da *n* pari a 55) anche se la Chi-Square distance presenta risultati molto simili. Questo netto miglioramento della precisione ottenuta mediante misure basate su istogrammi di colore è motivabile nel seguente modo: Zubud è un dataset molto "facile" in quanto le immagini presenti hanno contrasto e illuminazione molto simili il che porta ad avere istogrammi di colore molto uniformi, di conseguenza le distanze tra questi risultano essere più efficienti.

È possibile notare un ulteriore peggioramento della misura di distanza ottenuta come rapporto tra numero di inliers e numero di match usando RANSAC +SIFT.

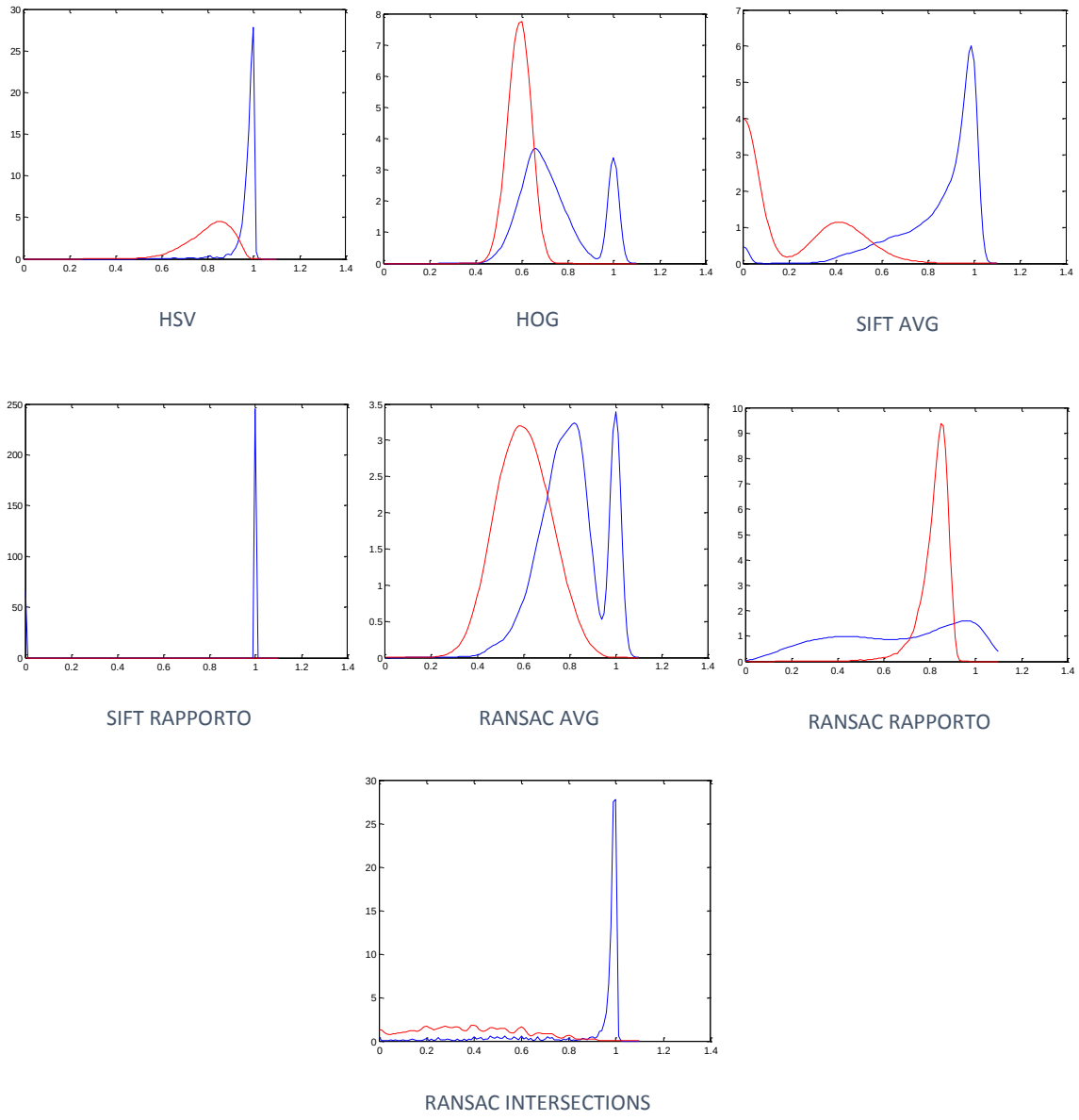


Figura 47 - Rappresentazione dei grafici Genuine/Impostor per il dataset Zubud

In questi grafici è possibile apprezzare più facilmente il significativo miglioramento di prestazioni che si è verificato per i metodi basati su *histograms features* anche se i metodi basati sulla media degli score con SIFT risultano comunque ottimi.

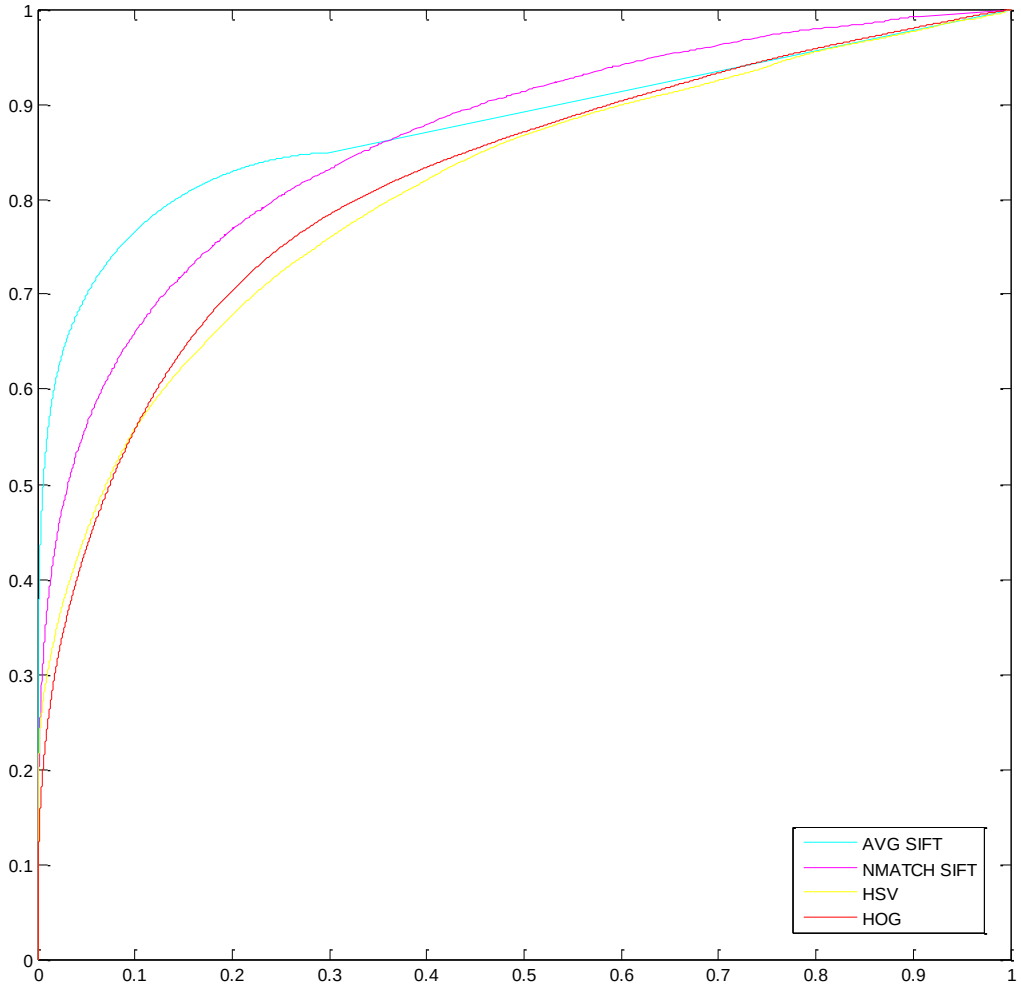


Figura 48 - Rappresentazione delle curve ROC a confronto ottenute sul dataset Sheffield Buildings

Come già affermato in precedenza per il dataset Sheffield building sono assenti i valori riguardati le distanze che fanno uso dell'algoritmo RANSAC a causa dell'elevata complessità computazionale dello stesso.

È possibile notare come le prestazioni degli algoritmi testati abbiano subito un piccolo peggioramento, questo è sicuramente dovuto al fatto che il dataset preso in esame contiene immagini che ritraggono lo stesso edificio in diversi momenti della giornata e quindi soggetto a diversi tipi di illuminazione. Fatta questa considerazione affermiamo che il peggioramento più significativo si riscontra nelle prestazione della distanza basata su istogrammi HSV, in quanto, come già enunciato in precedenza, questo algoritmo non risulta essere robusto nella classificazione di immagini che subiscono un drastico cambiamento di luminosità e contrasto. Un miglioramento delle prestazioni si potrebbe avere facendo uno studio sulla parametrizzazione della costruzione dell'istogramma HSV stesso. Per quanto riguarda le altre

feature possiamo affermare che HOG si dimostra essere più robusto rispetto all'istogramma HSV ma comunque anche in questo caso la distanza SVG SIFT risulta essere la migliore.

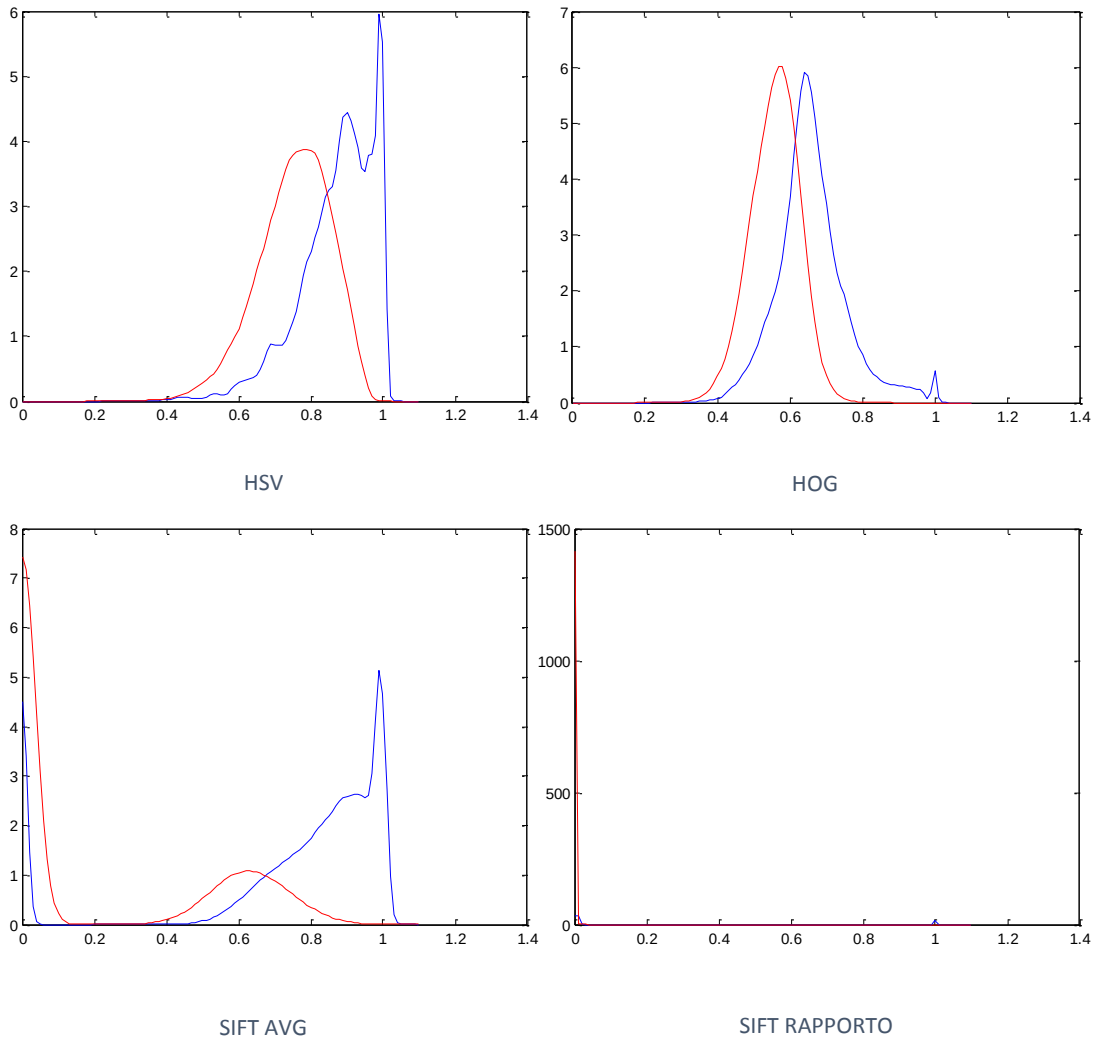


Figura 49 - Rappresentazione dei grafici Genuine/Impostor per il dataset Sheffield Buildings

Anche da questo tipo di rappresentazione grafica si evince un peggioramento delle prestazioni. Le curve genuine (in blu) e impostor (in rosso) risultano essere nettamente sovrapposte, risulterà quindi più complicato trovare un valore di soglia per distinguere nel modo più efficiente il match corretti da quelli errati.

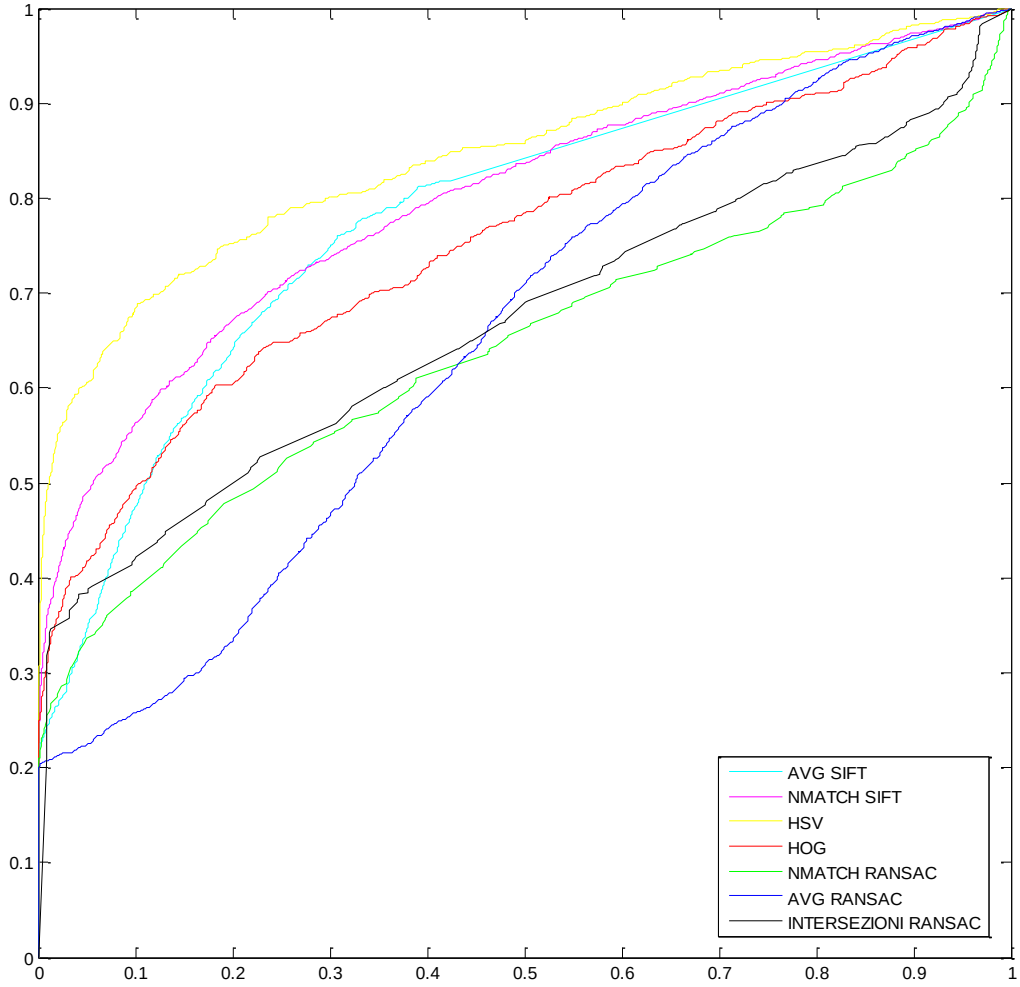


Figura 50 - Rappresentazione delle curve ROC a confronto ottenute sul dataset Zubud

In questo grafico si evince un drastico peggioramento degli algoritmi di classificazione che sfruttano features locali. Il migliore risultato in questo caso è stato ottenuto dalla misura basata su istogrammi HSV; anche in questo caso è possibile dare una spiegazione ai valori ottenuti:

Una prima considerazione va fatta sulla dimensione delle immagini del dataset: essendo queste molto piccole, come già affermato in precedenza, è inevitabile che le prestazioni degli algoritmi che sfruttano le SIFT feature crollino in quanto vengono individuati un numero inferiore di punti salienti e di conseguenza risulta più difficile individuare un match corretto. In secondo luogo le immagini del dataset Zubud Objects ritraggono oggetti molto colorati e pieni di dettagli, non edifici quindi risultano molto efficienti le misure che fanno un forte affidamento sui colori presenti sulle immagini.

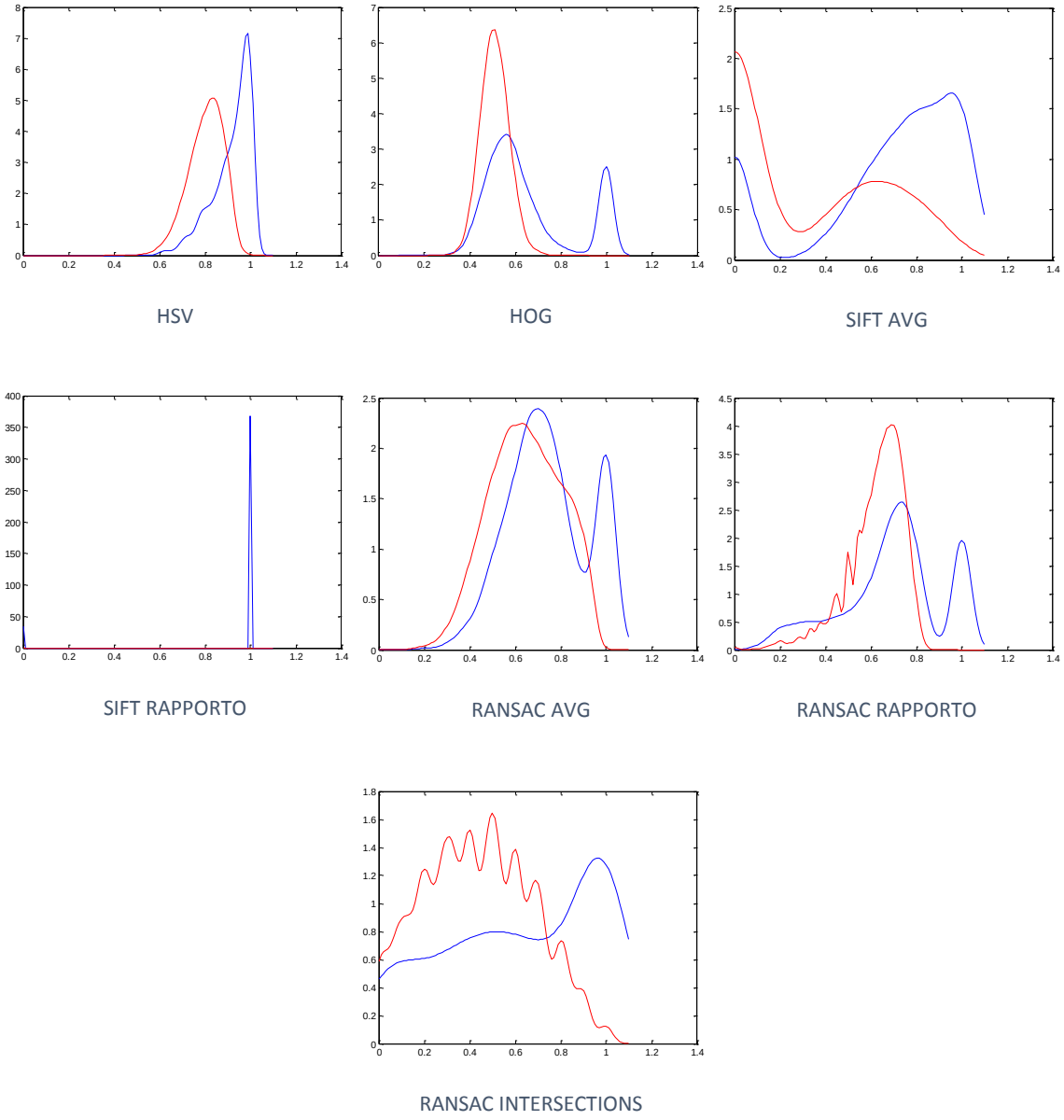


Figura 51 - Rappresentazione dei grafici Genuine/Impostor per il dataset ZuBud Objects

Risulta qui ancora più immediato riscontrare che le prestazioni degli algoritmi basati su istogrammi rimangano più o meno costanti mentre quelli basati su descrittori locali subiscano un grosso peggioramento.

4.6 Analisi dei risultati

Possiamo affermare che, nel complesso, l'algoritmo che permette una classificazione delle immagini nel modo più corretto è sicuramente quello che si basa sulla media dei primi *n* migliori match ottenuti confrontando le SIFT features. SIFT permette di ottenere ottimi risultati nella maggior parte dei casi grazie alla loro natura di descrittori locali e come tali poco soggette alla variazione di luminosità e contrasto, la robustezza di questi descrittori a variazioni di illuminazione rende l'algoritmo di instance recognition un ottimo candidato per l'implementazione del nostro caso d'uso. Le uniche debolezze che si possono riscontrare in questo algoritmo dipendono da principalmente due fattori: il primo è la dimensione dell'immagine, per immagini molto piccole le prestazioni dell'algoritmo peggiorano notevolmente; questo di per sé non risulta essere un grosso problema per lo studio in quanto la maggior parte degli smartphone di nuova generazione sono dotati di una fotocamera sufficientemente potente in grado di scattare fotografie con risoluzione adatta all'estrazione di SIFT. Nel caso la risoluzione dell'immagine sia invece troppo elevata e quindi implichi un'eccessiva dilatazione del tempo di calcolo per l'estrazione dei descrittori sarà sufficiente applicare un ridimensionamento dell'immagine. Un secondo problema è il soggetto ritratto nell'immagine, i descrittori locali sono ottimi per fare classificazione di edifici ma risultano essere meno efficienti per classificare oggetti, probabilmente questo dipende dal fatto che oggetti di piccole dimensioni contengono troppi dettagli come ad esempio delle scritte, in questi casi i descrittori SIFT estratti sono spesso molto simili tra di loro e quindi risulta essere più probabile fare match incorretti.

Per quanto riguarda invece descrittori globali, quello che si dimostra essere più accurato è quello basato sull'istogramma HSV anche se, come già affermato in precedenza, risulta poco robusto per cambi di luminosità contrasto e tono di colore, nonostante ciò rimane comunque un ottimo metodo di classificazione in quanto si è dimostrato adatto anche a identificare foto di oggetti e non solo di edifici. Al fine di migliorare ulteriormente questo metodo di matching si può pensare di parametrizzare la costruzione degli istogrammi stessi. Nei test qui presentati si è scelto di mantenere valori statici per la dimensione dei bucket dell'istogramma: per il canale della tinta si è suddiviso il range dei valori in 20 bucket, per la saturazione e per la luminosità 5 bucket; si è fatta questa scelta in quanto era nostra intenzione dare una maggior rilevanza alla tinta piuttosto che alla luminosità o alla saturazione in quanto è il canale che, al variare dell'illuminazione e contrasto, rimane pressoché costante. Parametrizzando la dimensione dei bucket di ogni canale si può pensare di individuare un'ottimizzazione dell'algoritmo al variare dei suddetti parametri. Un ulteriore miglioramento per questo tipo di algoritmi può essere

effettuato eseguendo operazioni di pre-processing come ad esempio: operazioni di contrast-stretching, normalizzazione dell'istogramma di colore dell'intera immagine o altro. Adottando questi accorgimenti si ottiene un bilanciamento del contrasto e della luminosità dell'immagine, in questo modo due immagini simili, soggette a illuminazione diversa, saranno confrontate più accuratamente.

Alla luce dei dati sopra elencati e delle considerazioni qui effettuate è possibile scegliere quale algoritmo di image recognition adottare per l'implementazione sulla piattaforma Android:

L' algoritmo basato su SIFT features rappresenta un ottimo candidato per l'implementazione in quanto si è dimostrato robusto ed efficiente nella maggioranza dei casi. È comunque un algoritmo con costo computazionale elevato; questo comporterebbe un peggioramento della user experience: immaginiamo che un giocatore dotato di un dispositivo mobile obsoleto scatti una foto per rispondere ad un quesito di una tappa e debba attendere a lungo per la verifica della risposta; questa attesa risulterebbe essere troppo elevata rendendo non piacevole l'esperienza di gioco. Non bisogna comunque trascurare il fatto che questo tipo di descrittori secondo i test sopra riportati è risultato essere meno efficiente per identificare oggetti, questo pone delle limitazioni qualora si volesse estendere il concetto di caccia al tesoro non solo ad edifici o monumenti ma anche ad oggetti. Un ultimo aspetto da considerare lo spazio su disco occupato dai descrittori estratti: la dimensione dei descrittori SIFT è variabile, dipende infatti dal numero di punti salienti individuati nell'immagine, questo comporta nel caso si costruiscano cacce al tesoro con un numero elevato di tappe si potrebbe verificare un'eccessiva dilatazione della dimensione del file che l'utente deve scaricare.

Un secondo ottimo candidato all'implementazione è l'algoritmo basato su istogrammi di colore in scala cromatica HSV; si è scelto di adottare questo metodo in quanto pur non essendo robusto e performante come quello basato su SIFT rappresenta una valida alternativa per il nostro caso d'uso. In particolare la scelta si è basata sui seguenti fattori:

- La dimensione del descrittore estratto è costante e contenuta, indipendentemente dalla tipologia e dalla risoluzione dell'immagine.
- È un algoritmo caratterizzato da una complessità computazionale non troppo elevata e largamente supportata dai dispositivi mobili.
- Si ipotizza che la maggior parte delle cacce al tesoro vengano effettuate di giorno quando l'illuminazione è sempre presente e non dovrebbe comportare troppi problemi per questo tipo di matching; d'altro canto la qualità delle foto di interi edifici scattate durante le ore notturne, quando l'illuminazione scarseggia, con uno smartphone risulta essere scarsa e

quindi comporterebbe in ogni caso un problema per qualsiasi algoritmo di riconoscimento di immagini.

- Le prestazioni dell'algoritmo risultano essere più efficienti anche per la classificazione di oggetti piuttosto che di edifici o monumenti.
- I dispositivi smartphone delle nuove generazioni sono sempre dotati di GPS e giroscopio, con questi dispositivi è già possibile stabilire se la risposta che viene data al quesito della tappa è corretta; l'algoritmo implementato risulta quindi essere un ulteriore supporto di verifica che non comporta la necessità di avere garanzie assolute per affermare se la risposta è corretta o meno.

In conclusione l'algoritmo implementato su piattaforma Android è un ottimo compromesso tra esperienza di gioco ed efficienza; si ritengono quindi soddisfatti i parametri che si erano proposti in fase di pianificazione dello studio.

Conclusioni

Lo scopo di questo studio è rappresentato dalla realizzazione di un'app, GeoPhotoHunt, per dispositivi Android che consenta di creare, giocare e condividere cacce al tesoro fotografiche. In secondo luogo si propone di implementare un CBIR System su piattaforma mobile che consenta di analizzare le foto scattate dagli utenti durante l'esperienza di gioco in modo da affermare se la foto fornita in risposta al quesito formulato per ciascuna tappa della caccia al tesoro sia corretta. Un requisito importante del sistema è il processo di elaborazione e confronto delle immagini deve avvenire sul client in modo da consentire l'utilizzo dell'applicazione anche in assenza di una connessione alla rete.

Al fine di soddisfare gli obiettivi sopra elencati si è sviluppata una prima versione dell'applicazione per dispositivi mobili e il relativo interfacciamento con un servizio server remoto nonché si è effettuato uno studio su algoritmo di riconoscimento di immagini presenti in letteratura al fine di poter valutare quale fosse il metodo migliore da implementare nella stessa app Android.

Dallo studio è emerso che le prestazioni migliori si hanno nel caso si sfruttino algoritmi basati su descrittori di immagini locali, come SIFT, data la loro robustezza nei confronti di cambiamenti di luminosità contrasto e variazioni di inquadratura. Purtroppo questi descrittori comportano un costo computazionale abbastanza elevato e una potenza di calcolo sufficientemente supportato da molto dispositivi al giorno d'oggi commercializzati. In base a queste considerazioni si è quindi scelto di ripiegare su un algoritmo meno performante ma con costo computazionale decisamente più contenuto ovvero quello basato sulla distanza tra istogrammi di colore sulla scala HSV.

Il prototipo realizzato è perfettamente funzionante, ma certamente non ottimizzato e include alcune lacune che potranno essere colmate in possibili sviluppi futuri ovvero:

- Implementare sul server un'interfaccia web che permetta di creare o consultare le cacce al tesoro piuttosto che controllare i punteggi totalizzati dagli amici o lasciare commenti sulla qualità delle cacce create dagli altri utenti.
- Integrare la possibilità di interfacciare l'applicazione a social network, supportando ad esempio il login con credenziali già utilizzate nei servizi dei suddetti social network.
- Migliorare l'algoritmo di riconoscimento di immagini integrando operazioni di pre-processing oppure ampliando gli studi condotti sui vari metodi presenti in letteratura.

- Effettuare il porting dell'applicazione anche su dispositivi iOS e Windows Phone.
- Integrare, al fine di rendere l'applicazione ancor più indipendente dalla connessione internet, un sistema di supporto delle mappe in sostituzione a GoogleMaps.
- Ampliare l'insieme dei dataset su cui testare gli algoritmi.
- Testare nuovi algoritmi di computer vision.

Bibliografia

[1] «GEOCACHING,» [Online]. Available: http://www.geocaching.com/.

[2] «OPENCACHING,» [Online]. Available: http://www.opencaching.com/it.

[3] «c:geo - GooglePlay,» [Online]. Available: https://play.google.com/store/apps/details?id=cgeo.geocaching&hl=it.

[4] «VISITO Tuscany,» [Online]. Available: https://play.google.com/store/apps/details?id=it.visitotuscany&hl=it.

[5] «WHAIWHAI,» [Online]. Available: http://www.whaiwhai.com/.

[6] «Picture Geo Hunt - GooglePlay,» [Online]. Available: https://play.google.com/store/apps/details?id=pk.games.PictoGeoHunt.

[7] «Scavify,» [Online]. Available: https://www.scavify.com/.

[8] «Mobilogue - GooglePlay,» [Online]. Available: https://play.google.com/store/apps/details?id=info.collide.android.mobilogue&hl=it.

[9] A. Giemza, N. Malzahn and U. Hoppe, "Mobilogue: Creating and Conducting Mobile Learning Scenarios in Informal Settings," *Proceedings of the 21st International Conference on Computers in Education. Indonesia: Asia-Pacific Society for Computers in Education,* 2013.

[10] T.-J. Chin, Y. You, C. Coutrix, J.-H. Lim, J.-P. Chevallet e L. Nigay, «Mobile phone-based mixed reality: the Snap2Play game,» *The Visual Computer.*

[11] D. Kohen-Vacs, M. Ronen e S. Cohen, «Mobile Treasure Hunt Games for Outdoor Learning,» *Bulletin of the IEEE Technical Committee on Learning Technology,* vol. 14, n. 4, 2012.

[12] D. Grüntjens , S. Groß , D. Arndt e S. Müller , «Fast Authoring for Mobile Gamebased City Tours,» *Procedia Computer Science,* vol. 25, pp. 41-51, 2013.

[13] C. Neustaedter e T. K. Judge, «See It: A Scalable Location-Based Game for Promoting Physical Activity,» *Interactive Poster,* 2012.

[14] P. Santos, D. Hernández-Leo e J. Blat, «To be or not to be in situ outdoors, and other implications for design and implementation, in geolocated mobile learning,» *Elsevier,* 2013.

[15] K. Lin, J. Jia, W. Huang e C. Fang, «Smartphone Landmark Image Retrieval Based on Lucene and GPS,» *The 9th International Conference on Computer Science & Education,* 2014.

[16] K. D. Joshi , S. Bhavsar e R. Rajesh C. Sanghvi , «Image Retrieval System Using Intuitive Descriptors,» *Elseiver,* 2014.

[17] V. Tyagi, A. Pandya,, A. Agarwal e B. Alhalabi, «Validation of Object Recognition Framework on Android Mobile Platform,» *IEEE 13th International Symposium on High-Assurance Systems Engineering,* 2011.

[18] T. Deselaers, D. Keysers e H. Ney, «Features for Image Retrieval: An Experimental Comparison,» 2007.

[19] K. D. Joshi, S. N. Bhavsar e R. C. Sanghvia, «Image Retrieval System using Intuitive Descriptors,» *Elseiver,* pp. 535 - 542, 2014.

[20] L. Nanni, A. Lumini e S. Brahnam, «Ensemble of different local descriptors, codebook generation methods and subwindow configurations for building a reliable computer vision system,» *Journal of King Saud University,* 2014.

[21] J. Wu e J. Rehg, «CENTRIST:A Visual Descriptor for Scene Categorization,» *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* 2011.

[22] J. Wu e J. M. Rehg, «Where am I: Place instance and category recognition using spatial PACT».

[23] «Caltech Buildings Dataset,» [Online]. Available: http://www.vision.caltech.edu/malaa/datasets/caltech-buildings/.

[24] «Zubud Dataset,» [Online]. Available: http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html.

[25] «Sheffield Buildings dataset,» [Online]. Available: https://www.sheffield.ac.uk/eee/research/iel/research.

[26] S. Hua, G. Chen, H. Wei e Q. Jiang, «Similarity measure for image resizing using SIFT,» *EURASIP Journal in Image and Video Processing,* 2012.