UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Interrogazione di Immagini in DBMS Multimediali: analisi e sperimentazione

Relatore

Chiar.mo Prof. Sonia Bergamaschi

Contro Relatore

Chiar.mo Prof. Paolo Tiberio

Tesi di Laurea di

Marco Torricelli

Anno Accademico 1998 – 1999

Parole chiave:
Database Multimediali
Content Based Visual Query
Feature extraction
Distance Metrics
QBIC

Alla mia famiglia

RINGRAZIAMENTI

Ringrazio la Professoressa Sonia Bergamaschi per l'aiuto fornito alla realizzazione della presente tesi.
Un ringraziamento particolare va inoltre al Dott. Marco Patella dell'Università degli Studi di Bologna per i preziosi consigli e la costante disponibilità riguardo l'installazione del motore di ricerca WindSurf.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

L'evoluzione tecnologica degli ultimi anni ha permesso l'acquisizione, la memorizzazione e la distribuzione a costi ridotti di sorgenti di informazione innovative. Immagini, filmati, file audio, frammenti di testo scritto a mano, rappresentano gli esempi più noti di queste nuove tipologie di dati.

Le nuove tecnologie hanno portato alla riduzione dei costi necessari alla gestione ed all'acquisizione di queste nuove sorgenti. Basti pensare agli scanner per le immagini e alle telecamere per i filmati. Questi, grazie allo sviluppo della tecnologia CCD, hanno beneficiato della riduzione delle dimensioni, della semplificazione dei circuiti di interfaccia e della conseguente riduzione dei costi.

L'aumento delle dimensioni, a parità di costo, dei dispositivi di memorizzazione primaria (memorie RAM), secondaria (Hard Disc) e terziaria (Cd-Rom) ha consentito l'elaborazione e l'archiviazione di file di dimensioni ed in quantità sempre maggiore.

D'altro canto l'aumento delle capacità di tutti i computer ha consentito anche alla maggior parte degli utenti di personal computer operazioni prima consentite solo agli utilizzatori di mainframe.

L'evoluzione della rete Internet e la disponibilità di modem sempre più veloci ha favorito la distribuzione di immagini, filmati e suoni, sia separatamente che integrati tra loro sotto forma di documenti multimediali, ad una platea di utenti crescente in maniera esponenziale.

A questa crescente disponibilità di sorgenti Multimediali ed all'evoluzione dell'Hardware necessario alla loro gestione non è però seguita un'uguale evoluzione degli strumenti Software forniti agli utenti per recuperare nel minor tempo possibile le informazioni possibilmente conformi alle loro richieste.

I Database che memorizzano dati classici come numeri interi o floating point, stringhe di testo e campi data si avvalgono, per il recupero delle informazioni memorizzate al loro interno, di potenti linguaggi di interrogazione come SQL e di strumenti per la gestione di Basi di Dati Relazionali (RDBMS) per ottimizzare i tempi di recupero.

Questi tipi di dati sono noti dall'introduzione dei database ed è ovvio che le

tecniche di recupero siano molto più evolute di quelle sui nuovi dati multimediali, inoltre questi sono intrinsecamente più complessi da analizzare rispetto a semplici stringhe o date di nascita.

Questa tesi analizza le problematiche relative alle interrogazioni di basi di dati multimediali ed in particolare nella prima parte si concentra sulla determinazione della similitudine tra immagini.

Lo schema della prima parte della tesi è il seguente:

- nel Capitolo 1 si introducono: le architetture di riferimento per i database multimediali e per quelli di immagini in particolare e si analizzano le modalità di interrogazione disponibili e le tecniche di valutazione dei risultati delle medesime;
- nel Capitolo 2 si illustra il panorama delle caratteristiche (feature) disponibili per caratterizzare il contenuto non semantico delle immagini da recuperare;
- nel Capitolo 3 vengono valutate le metriche disponibili per valutare la similitudine delle feature estratte dalle immagini.
- nel Capitolo 4 vengono illustrati gli algoritmi di segmentazione che permettono di migliorare la precisione nel recupero delle immagini associando informazioni sulla posizione ai sottoinsiemi delle feature estratte nel Capitolo 2.
- Nel Capitolo 5 si introducono i nuovi indici per i dati multimediali e si illustrano alcune tecniche per la riduzione delle dimensioni delle feature;
- nel Capitolo 6 vengono analizzate le potenzialità e le soluzioni tecniche implementate da una serie di motori di ricerca per immagini disponibili in rete. In particolare vengono analizzati: il sistema QBIC sviluppato presso i laboratori IBM e WindSurf sviluppato presso il DEIS dell'Università di Bologna.

Nella seconda parte i Capitoli 7, 8 e 9 illustrano l'implementazione di un'applicazione per la gestione di un database di immagini in QBIC. In particolare:

- nel Capitolo 7 viene illustrata la libreria di classi implementata da QBIC, che permette di includere le funzionalità di recupero delle immagini all'interno dei programmi sviluppati dagli utenti;
- nel Capitolo 8 vengono descritte le caratteristiche ed il funzionamento di una applicazione sviluppata per facilitare l'inserimento delle immagini e

le interrogazioni sul contenuto in un Database di immagini implementato tramite QBIC;

- infine nel capitolo 9 si confrontano le prestazioni nel recupero di gruppi di immagini appartenenti al Database, per tutte le opzioni di query disponibili, per i sistemi QBIC e WindSurf di cui sono disponibili la versione sviluppata in questa tesi e quella fornita dall'Università di Bologna.

Capitolo 1

I Database Multimediali

Fin dalla loro introduzione l'utilizzo principale, in campo industriale, dei computer è consistito nell'elaborazione di insiemi di simboli.

Questi sistemi selezionano simboli appartenenti ad uno specifico alfabeto d'ingresso e producono un diverso set, frutto di adeguate elaborazioni, come uscita. Questo è il modello a cui si ispirano i sistemi di elaborazione teorizzati da Turing nell'omonima macchina.

Negli ultimi anni si è verificato il crescente bisogno di realizzare sistemi in grado di interrogare ed elaborare vaste quantità di dati che non sempre sono facilmente descrivibili attraverso semplici simboli. Alcuni esempi di queste nuovi tipologie di dati consistono in:

immagini, filmati, suoni, testi strutturati, appunti scritti a mano, ecc... Queste nuove tipologie di dati sono solamente una classificazione parziale di una realtà in continua espansione e ciascun tipo è ulteriormente frammentato in una miriade di formati con strutture anche molto differenti:

gif, jpeg, png, ecc... per le immagini, avi, mpeg, ecc... per i filmati, waw,mp3, ecc... per i file audio, doc, txt, html, ecc... per i file di testo strutturato.

Una definizione informale definisce un sistema per la gestione di un database multimediale MMDBMS (Multimedia Database Management System)come un ambiente che organizza differenti tipi di dati, che possono essere rappresentati in una moltitudine di formati e risiedere fisicamente su media diversi.

Per poter funzionare in maniera efficiente un MMDBMS deve soddisfare le seguenti caratteristiche:

- Deve avere la possibilità di interrogare in maniera uniforme i dati (immagini, testi, ecc.) memorizzati in diversi formati.

  Ad esempio un MMDBMS deve fornire la possibilità di interrogare ed integrare in maniera uniforme i dati provenienti da database relazionali, file di

dati non strutturati oppure dati memorizzati in DBMS di tipo object oriented o spaziali.

In particolare, interrogazioni che spaziano su sorgenti memorizzate in DBMS diversi devono essere in grado di fondere i risultati e presentarli in maniera uniforme

- In maniera analoga deve essere possibile interrogare dati appartenenti a media diversi.

  Ad esempio un MMDBMS deve essere in grado di interrogare contemporaneamente sia un Database di immagini, uno audio e uno relazionale ed integrare adeguatamente i risultati.

- Un MMDBMS deve rendere possibile il recupero di oggetti appartenenti a media diversi, memorizzati localmente, con continuità senza ritardi evidenti.

  Siccome oggetti multimediali come i video spesso occupano grosse quantità di memoria, nell'ordine di decine di gigabyte, anche in formato altamente compresso bisogna tenere in considerazione il fatto che questi dati possono essere memorizzati su sistemi di memorizzazione secondaria (hard disk) e terziaria(CD-ROM o nastri) che presentano prestazioni diverse.

  Quindi sistemi per il recupero di informazioni multimediali devono tenere conto delle variazioni dei tempi di recupero e renderli il più possibile simili.

- Un MMDBMS deve, partendo dalla risposta di una query, che è una struttura matematica di qualche tipo, sviluppare una presentazione di questi risultati in maniera audiovisiva.

  Indipendentemente dal fatto che l'interfaccia ed il contenuto di ciascuna presentazione possa variare da un applicazione all'altra, l'utente deve avere la possibilità di specificare la struttura ed il contenuto della risposta che vuole ottenere dal sistema.

- Infine quando la presentazione è stata delineata secondo le specifiche del punto precedente deve essere possibile distribuirla in maniera che venga rispettata la qualità di trasmissione del mezzo impiegato.

  Ad esempio se al punto precedente il sistema che sovrintende alla presentazione ha deciso che un filmato ed un file sonoro devono essere presentati in parallelo l'MMDBMS deve essere in grado di garantire che la presentazione non subisca rallentamenti o sfasamenti dovuti ai requisiti diversi che i due flussi di dati devono soddisfare.

Inoltre, poiché i risultati delle query vengono trasmessi attraverso i nodi della rete, per raggiungere i vari utilizzatori il sistema di gestione della base multimediale deve garantire che la rete presenti sempre una banda adeguata e la disponibilità dei buffer necessaria a garantire una qualità di servizio tale da trasmettere senza interruzioni la presentazione dei risultati prodotti dal sistema.

Negli ultimi decenni sono stati sviluppati i linguaggi di interrogazione, le tecniche di indicizzazione, gli algoritmi di recupero e i metodi di aggiornamento impiegati nelle basi di dati relazionali, orientate agli oggetti, spaziali e temporali ed altri tipi di database.

Ognuno di questi sistemi ha esteso i linguaggi, gli algoritmi e le conoscenze precedenti allo scopo di incorporare nuovi ed importanti tipi di dati.

In questa ottica i dati multimediali non sono differenti dai precedenti e le novità devono integrarsi nel panorama esistente senza richiedere di ricreare tutto di nuovo.

Ad esempio un linguaggio di interrogazione per le immagini deve tenere conto di oltre 30 anni di ricerca sull'esecuzione e sull'ottimizzazione delle query senza ripartire necessariamente da zero.

1.1 Architetture per Database Multimediali (MMDBMS).

Quando si realizza un database multimediale che rappresenta una vasta varietà di tipi appartenenti a media diversi dobbiamo confrontarci con diversi aspetti riguardanti l'organizzazione del loro contenuto informativo.

In questo studio analizziamo tre architetture per l'organizzazione del contenuto di un database multimediale che adottano i seguenti principi:

Principio di autonomia: possiamo raggruppare insieme tutte le immagini, tutti i video e tutti i documenti e creare un indice per ciascun raggruppamento in modo da ottenere la massima efficienza possibile per la rispettiva sorgente.

Parliamo di autonomia poiché ciascuna sorgente mantiene una organizzazione totalmente separata dalle altre, in questa maniera non è necessario scendere a compromessi ed è possibile adottare la soluzione migliore ed ottimizzare le prestazioni. La figura 1.1 illustra un diagramma concettuale di diversi tipi di dati multimediali organizzati secondo il principio di autonomia.

Principio di uniformità: al contrario della soluzione precedente possiamo utilizzare una singola struttura astratta che viene impiegata per tutti i tipi di media supportati dal sistema e creare un indice unico che indirizza tutte le entità del sistema indipendentemente dalla loro natura.

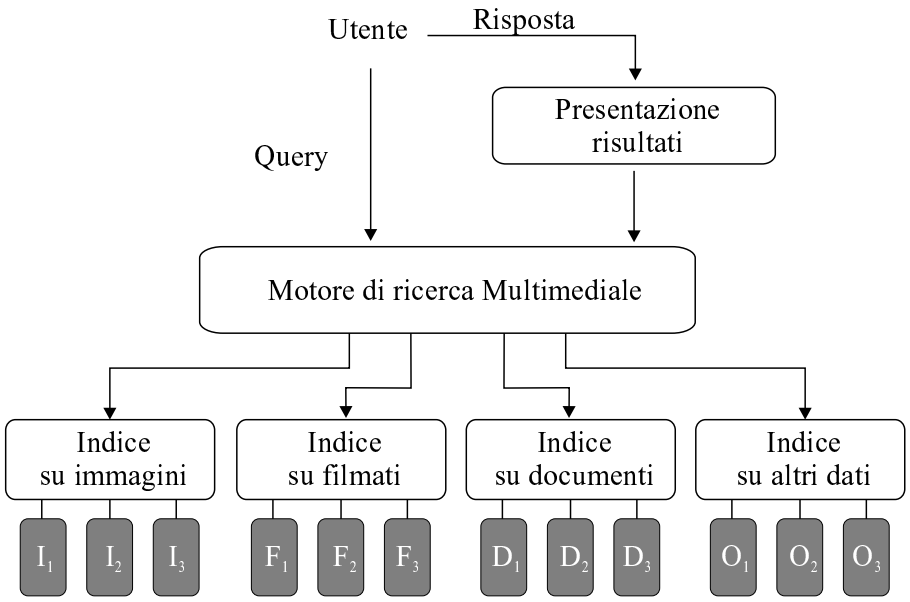


Figura 1.1: Architettura di un database multimediale secondo il principio di autonomia.

In altre parole rappresentiamo il contenuto dei differenti dati multimediali all'interno di un'unica struttura e poi sviluppiamo algoritmi per interrogare questa nuova struttura. La figura 1.2 illustra un diagramma concettuale di diversi tipi di dati multimediali organizzati secondo il principio di uniformità.

Principio di organizzazione ibrida: la terza possibilità è quella di adottare una soluzione ibrida in cui alcuni tipi hanno la loro struttura specifica secondo l'approccio autonomo, mentre altri invece, in accordo con l'approccio omogeneo, adottano una struttura unificata.

La scelta di quei dati che devono essere trattati in maniera omogenea e viceversa di quelli che devono essere trattati in maniera autonoma dipende dalle scelte progettuali che analizzano le caratteristiche dell'organizzazione dei dati che si vogliono privilegiare. La figura 1.3 illustra un diagramma concettuale di diversi tipi di dati multimediali organizzati secondo il principio di organizzazione ibrida.

Ognuna delle tre rappresentazioni illustrate presenta vantaggi e svantaggi.

Le architetture basate sul principio dell'autonomia richiedono la creazione di algoritmi e strutture dati per ogni tipo di media supportato, inoltre sono necessarie tecniche per eseguire i join necessari per riunificare i risultati parziali ottenuti interrogando le singole strutture dati.

D'altro canto creare strutture specializzate che possano accedere efficientemente a ciascun tipo di dato permette di ridurre drasticamente i tempi di accesso

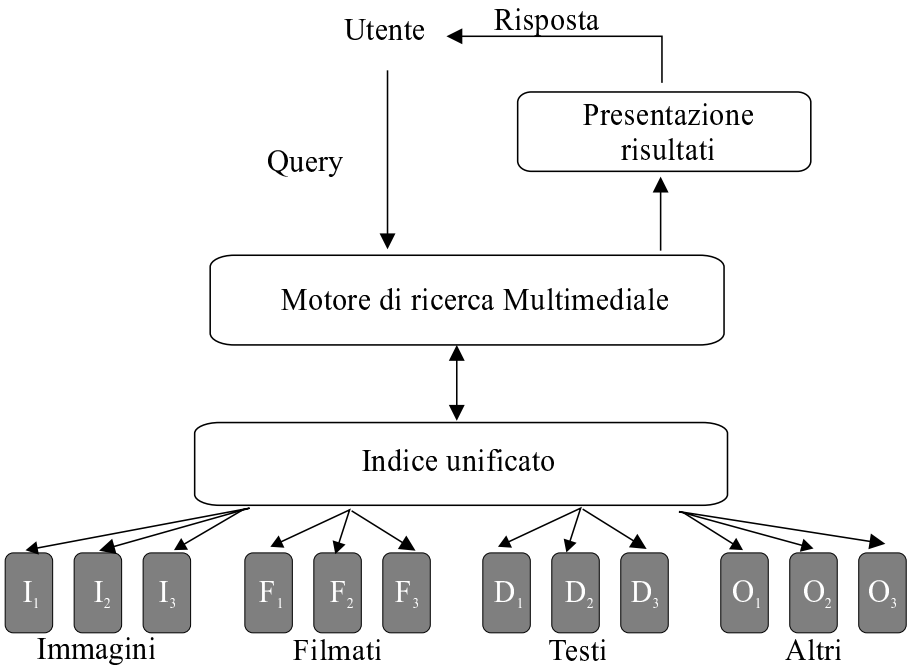


Figura 1.2: Architettura di un database multimediale secondo il principio di uniformità.

e di risposta alle interrogazioni.

Oltretutto quando occorre integrare basi di dati esistenti con strutture dati ad algoritmi ereditati il principio di autonomia ne permette l'utilizzo immediato senza sviluppare ulteriore codice. I sistemi orientati agli oggetti implementano il principio di autonomia trattando ciascun tipo di dato come un oggetto i cui metodi sono accessibili dal MMDBMS globale.

In contrasto con il principio di autonomia quello di uniformità richiede la determinazione di una struttura comune che possa contenere le informazioni relative a immagini, filmati, suoni e così via.

Questa architettura richiede di rappresentare quello che tutti i media hanno in comune con gli altri e di creare un unico indice su questi dati. In pratica il principio di uniformità viene realizzato attraverso i concetti di annotazioni o "metadata" dove le informazioni sul contenuto di ciascun elemento multimediale vengono espresse attraverso un metalinguaggio comune. Il vantaggio principale dell'adozione del principio di uniformità sta nella facilità di implementazione dei relativi algoritmi e nella assenza di join sui risultati parziali.

Lo svantaggio consiste nella modalità con cui vengono create le annotazioni che può essere manuale od automatica.

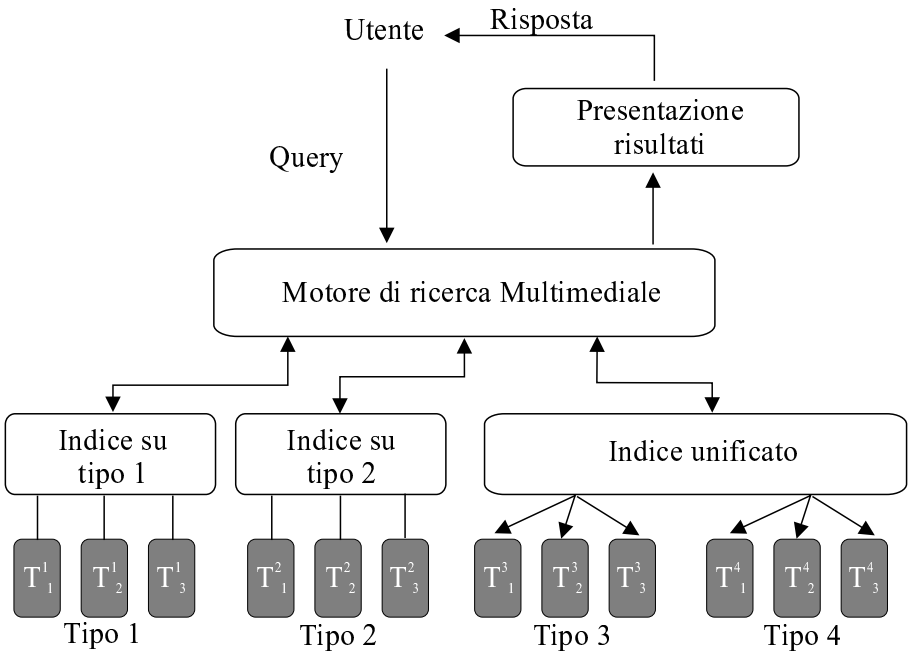


Figura 1.3: Architettura di un database multimediale secondo il principio di organizzazione ibrida.

L'operazione di annotazione manuale può risultare molto costosa in termini di tempo e di risorse impiegate ed inoltre in questo caso le annotazioni risultano dipendenti da chi le ha inserite.

L'annotazione automatica invece rischia di essere troppo superficiale trascurando informazioni importanti per la caratterizzazione della sorgente. Ad esempio strumenti per l'annotazione automatica delle immagini possono perdere informazioni sulla tessitura di gruppi di pixel, in maniera analoga uno strumento per l'annotazione di suoni può perdere informazioni sull'ampiezza o sulla frequenza in alcuni punti strategici del segnale.

Il principio dell'organizzazione ibrida si avvantaggia dei punti positivi di entrambe le architetture precedenti cercando contemporaneamente di attenuare molti aspetti negativi.

Supponiamo di creare un database multimediale formato dai tipi $(M_1, M_2, \ldots, M_n)$ iniziamo dividendo l'insieme in due parti:

1^a **parte:** Quei media che sono ereditati da database già esistenti che hanno già i loro indici e le loro procedure di interrogazione; in questo caso sfruttiamo il codice esistente incorporandolo all'interno del MMDBMS adottando l'approccio indipendente.

2^a **parte:** Quei media non ereditati che non hanno indici già esistenti ed algoritmi per manipolarli; in questo caso utilizziamo l'approccio uniforme creando un unico indice per il gruppo tollerando una leggera perdita di informazione comunque non critica per l'applicazione.

Per completare il sistema è necessario fornire il codice per eseguire i join tra le varie sorgenti utilizzando gli indici esistenti e quelli creati. Questo approccio sfrutta al massimo il lavoro già sviluppato minimizzando gli sforzi addizionali per integrare gli elementi specifici della particolare applicazione.

1.2 Database di immagini IDB.

I database di immagini IDB sono estensioni dei tradizionali DB in grado di archiviare, modificare e recuperare informazioni visuali.

Nei database tradizionali le unità di base delle informazioni, cioè gli attributi, hanno valori ben definiti, nelle immagini invece l'informazione di base è memorizzata nei pixel che rappresentano il valore di misurazioni di proprietà sensoriali in una regione dello spazio.

Possiamo rappresentare un'immagine, indipendentemente dal particolare formato di memorizzazione (gif, jpeg, tiff, tga, ecc ...), attraverso una matrice bidimensionale di elementi che mantiene le informazioni sul valore dei singoli pixel. I pixel contengono le informazioni sul colore di ogni punto dell'immagine.

Siccome esistono diversi spazi per la rappresentazione del colore (vedi Capitolo 2) la struttura dati impiegata per la memorizzazione del pixel dipende dallo spazio utilizzato, in particolare sia lo spazio RGB che quello HSV utilizzano tre valori interi per la rappresentazioni del colore, mentre nelle immagini a toni di grigio basta un solo valore intero.

Sebbene il valore del pixel abbia un chiaro significato quantitativo è ambigua la modalità con cui va a formare gli oggetti.

Da ciò si deduce quanto sia distante la conoscenza del valore dei pixel dalla determinazione del contenuto dell'immagine.

Una definizione informale descrive il contenuto di un'immagine come l'insieme di tutti gli oggetti che vengono considerati "interessanti" secondo il punto di vista dell'applicazione.

Per ottenere una rappresentazione più significativa del contenuto dell'immagine, che non sia il valore del pixel, associamo ad ogni oggetto all'interno dell'immagine un insieme di proprietà o "feature" del seguente tipo:

- **Descrittore di forma:** descrive la forma e la posizione della regione all'interno della quale è localizzato l'oggetto dell'immagine.

Un esempio di questi tipi di descrittore possono essere gli MBB (Minimum Bounding Box) che rappresentano un rettangolo minimo che contiene la regione ed è orientato secondo gli assi dell'immagine;

- **Descrittore di proprietà:** descrive una caratteristica di un pixel o di un gruppo di pixel dell'immagine.

  Esempi di proprietà di questo tipo sono l'istogramma dei colori o la caratterizzazione delle tessiture.

Siccome il calcolo delle proprietà richiede un notevole sforzo computazionale queste devono essere calcolate una volta per tutte ed archiviate all'interno di un Database.

I primi IDB, al fine di razionalizzare l'occupazione di memoria, suddividono l'immagine in celle di dimensione fissa in modo che le proprietà vengano calcolate solo a livello di cella. Come caso estremo è possibile definire un'unica cella calcolando proprietà valide solo per l'intera immagine.

Nei sistemi più moderni, la suddivisione in celle non è più statica, ma le proprietà vengono definite a livello di regioni estratte dall'immagine tramite algoritmi di estrazione delle regioni (Vedi Capitolo 4).

Un descrittore di forma può essere definito da un rettangolo, un poligono od un insieme di rettangoli.

Un descrittore di proprietà è una coppia di valori del tipo (PropName, PropValue).

La griglia viene definita specificando due costanti intere M,N che rappresentano la granularità dell'analisi:

l'immagine viene divisa in gruppi di M x N pixel dalla funzione $(xdivM)$, $(ydivN)$.

Un Database di immagini IDB viene definito dalla terna (GI, Prop, Rec) dove:

1. GI è un insieme di immagini con griglia nella forma (Immagine, M, N);
2. Prop è un insieme di proprietà di cella o di regione;
3. Rec è una relazione che associa ad ogni immagine un insieme di oggetti sotto forma di regioni rettangolari.

Le implementazioni correnti dei DBMS che supportano la gestione delle immagini si basano sulla tecnologia ad oggetti.

Si assume che ogni immagine sia un oggetto nel senso dei database orientati ad oggetti. Per la classe degli oggetti immagine vengono definiti i seguenti metodi.

- Metodi per l'editing delle immagini:

  permettono di aprire e visualizzare le immagini in formati diversi, eseguire operazioni di rotazione, ingrandimento, inversione e modifica di colori.

- Metodi per l'estrazione di feature:

  si applicano alle immagini e permettono di estrarre informazioni sulle caratteristiche visuali come il colore, la tessiture e le forme (Vedi Capitolo 2).

  Ad esempio, per il colore si estrae l'istogramma dei colori all'interno di un determinato spazio dei colori; per la tessitura si estraggono una serie di proprietà caratteristiche come la rugosità e la direzionalità e il contrasto, per le forme si impiegano descrittori di forma e momenti invarianti.

- Metodi di estrazioni delle regioni:

  questi metodi permettono, fissati dei criteri di omogeneità, di raggruppare i pixel dell'immagine in regioni in base a questi criteri (Vedi Capitolo 4).

Gli IDB, Figura 1.4, per essere considerati tali presentano le seguenti caratteristiche:

1. Possono essere comparate le immagini nella loro interezza oppure le regioni estratte tramite algoritmi di segmentazione.

2. Data un'immagine (regione) gli IDB vi associano una serie di proprietà salienti che devono essere in grado di discriminare fra immagini rilevanti e non rilevanti: non è necessario che estraggano tutte le immagini "interessanti", ma che scartino quelle meno importanti in modo da alleviare i compiti dell'utente; inoltre la loro estrazione deve risultare computazionalmente poco onerosa. Esempi di queste feature possono essere: il colore, la tessitura e la forma. Le N feature estratte vengono memorizzate in un vettore con N campi all'interno del database.

3. Nel Database viene creato un indice N-dimensionale, cosicché ogni immagine (regione) rappresenta un punto V_i nello spazio N-dimensionale. Questo indice può essere un'estensione multidimensionale di un R-tree o di un Point QuadTree.

4. Quando l'utente invoca l'esecuzione di una query nella forma: "trova tutte le immagini simili all'immagine d'esempio" il sistema recupera tutti vettori V_i e le relative immagini I la cui distanza dal vettore V_q è inferiore alla soglia fissata.

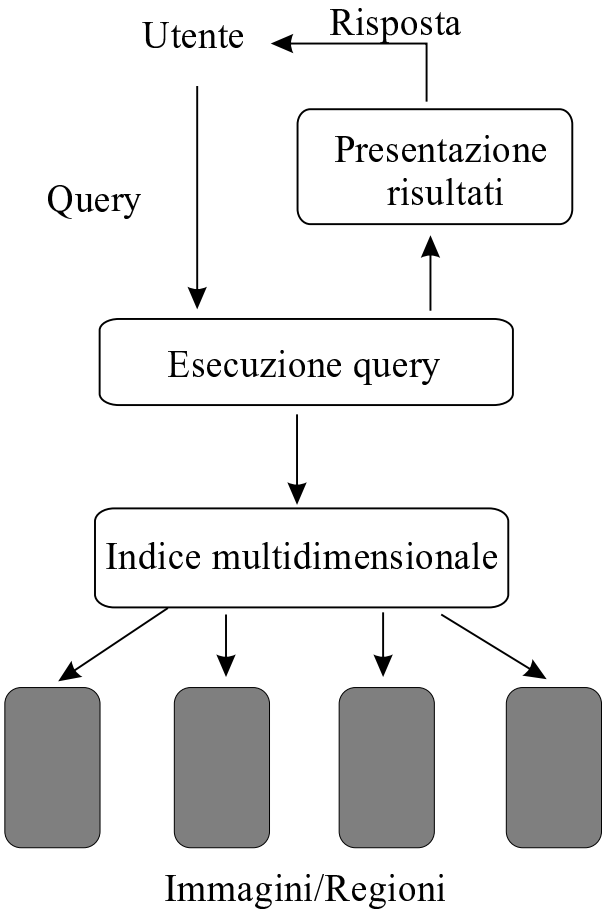


Figura 1.4: Architettura generale di un DB di immagini.

1.3 Query sul contenuto visuale delle immagini.

Le query basate sul contenuto visuale (Content Based Visual Query, CBVQ) forniscono uno strumento potente ed innovativo in grado di recuperare immagini e filmati all'interno di un database multimediale. Le CBVQ permettono, in mancanza di informazioni note a priori su ciò che è rappresentato all'interno dell'immagine, di specificare una serie di proprietà che si vuole risultino verificate dai risultati della query.

Gli studi più recenti sulle CBVQ hanno evidenziato il fatto che le proprietà che meglio si adattano a descrivere il contenuto delle immagini sono quelle sul colore, sulla tessitura, sulla forma e sul movimento se si tratta di sequenze di immagini in movimento.

I sistemi a cui si applicano le query visuali archiviano informazioni visu-

ali, immagini e filmati, che non appartengono a nessun dominio applicativo particolare.

Il fatto di non porre nessun tipo di filtro implica che non possano essere impiegati i classici algoritmi di "pattern recognition", che si applicano a domini applicativi dove le categorie di soggetti sono di un solo tipo.

Sviluppare sistemi che permettano di analizzare, archiviare e recuperare elementi visuali è una necessità pressante a causa della continua crescita del volume dei dati che viene prodotto e distribuito nelle forme più svariate. Ogni giorno nuove forme di media visuali come trasmissioni televisive, fotografie, disegni e animazioni sono rese accessibili in forma digitale ad una vasta comunità di utilizzatori.

Ad esempio sono milioni le immagini ed i filmati, rappresentanti i più disparati soggetti, che in maniera estremamente volatile vengono pubblicate sul WEB oppure su CD-ROM multimediali.

Queste tecniche sono comunque applicabili, con leggere modifiche, a particolari domini applicativi come immagini acquisite da satelliti, immagini mediche, ambientali e geografiche.

In questo studio supponiamo che le immagini da ricercare siano memorizzate in forma digitale all'interno di un Database e che gli utenti del sistema per la ricerca ed il recupero, collegati ad una rete, locale, intranet o Internet, eseguano query per recuperare le immagini con le caratteristiche desiderate.

Nell'implementazione più semplice possibile di un DBMS di immagini annotazioni e commenti testuali vengono associati in maniera manuale da chi archivia l'immagine.

Esempi di annotazioni testuali o parole chiave possono includere la descrizione dei soggetti principali contenuti all'interno dell'immagine, il nome dell'autore, la data e altre informazioni. Questi sistemi, che si appoggiano ad un DB di tipo relazionale o ad oggetti, presentano diversi problemi:

le descrizioni delle immagini rappresentano interpretazioni personali di chi le inserisce, possono essere incomplete oppure errate e ciò porta ad una scarsa capacità di recuperare le immagini basando le query solo sul confronto di stringhe testuali.

Le descrizioni devono essere inserite manualmente e se l'archivio è molto vasto questa attività risulta molto dispendiosa in termini di tempo e di risorse umane.

Per migliorare questa situazione i sistemi CBVQ fanno uso di query visuali che si basano sul confronto di feature visuali. Le feature visuali come il colore, la tessitura o le forme sono calcolate per ogni immagine del database, memorizzate all'interno dello stesso ed organizzate tramite indici multidimensionali per un più facile recupero.

Il sistema di CBVQ fornisce all'utente una serie di strumenti visuali tramite i

quali definire i criteri che le immagini devono soddisfare per far parte dei risultati della query.

La definizione dei criteri avviene attraverso un'interfaccia grafica che consente di specificare le linee guida sui colori o le forme da ricercare o di immettere i riferimenti ad immagini di esempio da usare come termini di paragone allo scopo di recuperare tutte le immagini simili agli esempi forniti.

Il concetto di similitudine tra immagini viene introdotto definendo delle metriche cioè delle funzioni di distanza che confrontano le feature estratte dall'immagine d'esempio con quelle memorizzate all'interno del database: più la distanza è alta più le immagini sono diverse e viceversa.

Le CBVQ possono recuperare le K, numero intero definito dall'utente, immagini più simili cioè con i punteggi più alti oppure recuperare tutte le immagini la cui distanza da quella d'esempio risulta minore di un valore sempre definito dall'utente.

I sistemi commerciali più famosi che mettono a disposizione strumenti di CBVQ sono:

QBIC [**?**] della IBM, il PhotoBook [**?**, **?**] del MIT, Chabot dell'università di Berkeley [**?**] e Virage [1].

QBIC elabora feature sul colore sulla tessitura e sulla forma mentre PhotoBook lavora sulle tessiture, sulle forme e sul riconoscimento di volti.

Un'ulteriore evoluzione delle CBVQ sono le query sul contenuto delle regioni (Content Based Region Query CBRQ) che prendono in considerazione le regioni dell'immagine con caratteristiche comuni.

La definizione di feature globale può risultare riduttiva poiché si perdono le particolarità dell'immagine. Ad esempio due immagini totalmente diverse possono avere la stessa distribuzione dei colori cioè lo stesso istogramma.

Alle immagini vengono applicati algoritmi di estrazione di regioni che partizionano i pixel dell'immagine in gruppi omogenei rispetto ad una certa proprietà.

Il partizionamento genera una serie di regioni disgiunte la cui unione rappresenta l'intera immagine con la proprietà che ogni pixel della regione ha valori della proprietà simili.

Nelle CBRQ non sono le feature globali che vengono confrontate, ma quelle estratte per ogni regione.

Una CBRQ può essere scomposta in una serie di CBVQ, una per ogni regione presente all'interno dell'immagine d'esempio:

ogni regione viene confrontata, nella stessa maniera in cui prima venivano confrontate le immagini intere, con le regioni estratte dalle immagini memorizzate all'interno del database, i punteggi ottenuti vengono poi valutati da una funzione

[1] **Reperibile all'indirizzo Internet http://www.virage.com**

che restituisce una misura globale della distanza. Sistemi commerciali che implementano le CBRQ sono il Visualseek [**?**] e il Webseek [**?**] della Università della Columbia e Windsurf dell'Università di Bologna [**?**]

Un'ulteriore evoluzione delle possibilità di query consente di prendere in considerazione le informazioni sulla disposizione spaziale delle regioni estratte secondo i criteri precedenti.

Siccome l'elaborazione delle disposizioni spaziali è computazionalmente molto complessa le informazioni spaziali vengono valutate solo in un secondo tempo.

In particolare prima vengono applicate la CBRQ sulle regioni coinvolte nella query:

vengono considerate solo le immagini che contengono le regioni coinvolte nella query indipendentemente della posizione, poi su questo risultato intermedio vengono applicate le analisi sulle disposizioni spaziali scartando le immagini con regioni in posizione diversa da quella specificata.

Possiamo distinguere tra due tipi di query sulle disposizioni spaziali assolute e relative:

le prime richiedono che le regioni delle immagini appartenenti ai risultati siano nella stessa identica posizione delle regioni nell'immagine d'esempio, le seconde invece sono meno vincolate e richiedono soltanto che siano rispettate le mutue disposizioni delle regioni, indipendentemente dalla loro effettiva posizione.

Un sistema che implementa tutte queste possibilità di query è il sistema SaFe sviluppato dalla Università della Columbia.

1.4 Efficacia ed Efficienza.

I giudizi ed i confronti in merito alla bontà di un sistema CBVQ vengono espressi considerando i parametri di Efficacia (Recall) e di Efficienza (Precision) che valutano le modalità con cui le immagini vengono estratte da una query visuale sul Database.

Supponiamo che I sia un insieme finito di immagini, che Alg sia un algoritmo che accetta come ingresso una definizione di una query visuale Q e ritorna in uscita un sottoinsieme dell'insieme delle immagini I.

Supponiamo pure di poter definire un predicato di rilevanza $Rel(I, Q)$, sulle immagini del Database, rispetto alla query Q, che estrae l'insieme di tutte e sole le immagini da ritenersi valide ai fini della determinazione del risultato.

Il predicato consiste nel definire l'insieme delle immagini che un utente imparziale selezionerebbe manualmente come risultato ottimo per una data query [**?**]. Sia A l'insieme delle immagini ritenute rilevanti e B l'insieme di quelle che

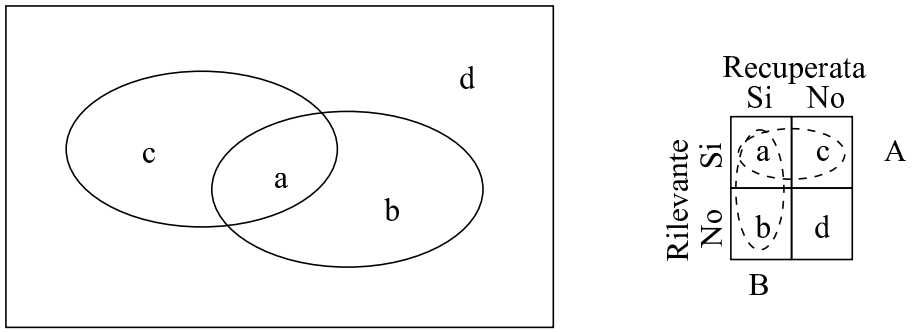


Figura 1.5: Rappresentazione insiemistica delle relazioni a, b, c e d.

vengono recuperate dalla query basata sul contenuto, allora possiamo definire i seguenti raggruppamenti dell'insieme I:

1. a= Elementi recuperati e rilevanti (detection);
2. b= Recuperati ma non rilevanti (Falso positivo);
3. c= Non recuperato ma rilevante (Falso negativo);
4. d= Non recuperato e non rilevante.

In particolare esprimendo queste definizioni attraverso il linguaggio degli insiemi 1.5 si ottiene che:

$$a = A \cap B, \quad b = B - a, \quad c = A - a, \quad d = I - (A + B) \tag{1.1}$$

Per motivi di praticità le analisi delle prestazioni di un motore di ricerca vengono valutate su un "Test Set" rappresentativo di immagini di dimensioni notevolmente minori rispetto alla popolazione I dell'intero Database.

L'efficacia di un algoritmo Alg, rispetto ad un dato predicato di rilevanza Rel ed ad un insieme di immagini di prova (Test Set), viene misurata determinando il rapporto tra il numero di immagini recuperate dall'algoritmo di ricerca che sono effettivamente valide e la globalità delle immagini valide:

$$\text{Efficacia (Recall)} = \frac{card(A \cap B)}{card(A)} = \frac{a}{a + c} \tag{1.2}$$

L'efficienza di un algoritmo di ricerca Alg, rispetto ad un dato predicato di rilevanza Rel ed ad un insieme di immagini di prova (Test Set) viene misurata determinando il rapporto tra il numero di immagini recuperate dall'algoritmo di ricerca

che sono effettivamente valide e l'insieme delle immagini recuperate:

$$\text{Efficienza (Precision)} = \frac{card(A \cap B)}{card(B)} = \frac{a}{a+b} \tag{1.3}$$

In altre parole l'Efficacia risponde alla domanda "Quante immagini che dovevano essere recuperate dall'algoritmo Alg vengono effettivamente recuperate?", mentre l'Efficienza risponde alla domanda "Quante immagini recuperate da Alg sono corrette?".

Un sistema ideale per valori di Efficacia ed Efficienza normalizzati sull'intervallo $[0, 1]$ dovrebbe rispondere alla condizione di Efficacia=Efficienza=1; in realtà non è possibile ottimizzare entrambe le grandezze e si rende necessario trovare un compromesso accettabile tra la precisione del sistema e la quantità di immagini restituite all'utente in risposta ad una data query [**?**].

Capitolo 2

Caratteristiche (feature) di immagini

Lo scopo dell'estrazione delle feature consiste nella determinazione degli attributi necessari a caratterizzare il contenuto visuale dell'immagine.

In particolare le uniche informazioni che si ottengono senza ulteriori elaborazioni delle immagini consistono nel colore dei pixel.

Sono quindi necessarie procedure in grado di estrarre proprietà di più alto livello in modo da ottenere una migliore caratterizzazione del contenuto dell'immagine. Le feature possono essere calcolate a livello di immagine o di regione.

Nel primo caso si ottiene una descrizione molto compatta, ma poco accurata dato che immagini anche molto diverse posso presentare le medesime feature poiché si perdono le informazioni sulla localizzazione nello spazio. Invece nel secondo caso, a fronte di una maggiore occupazione di memoria, si ha una rappresentazione delle feature a livello locale che mantiene le informazione sulla loro posizione.

È stato dimostrato che il contenuto visuale di un'immagine è adeguatamente descritto da una terna di feature che rappresentano le caratteristiche dell'immagine riguardanti il colore, la tessitura e la forma.

La scelta delle tipologie di feature da estrarre influenza le interrogazioni effettuabili sulla base di dati che memorizza le immagini poiché i risultati della query sono determinati confrontando le feature memorizzate con quelle estratte dall'immagine di query.

Descriviamo nei prossimi paragrafi una serie di metodologie impiegate per l'estrazione delle feature a partire dalle immagini non ancora elaborate.

2.1 Feature di Colore.

La radiazione elettromagnetica $f(\lambda)$ nell'intervallo della luce visibile (λ compreso tra 380nm e 780nm) viene percepita dall'occhio umano come colore o luce colorata.

È stato verificato sperimentalmente che il colore è percepito, all'interno dell'occhio, da tre recettori indipendenti che hanno picchi di risposta in corrispondenza delle lunghezze d'onda associate ai colori rosso (**R**ed), verde (**G**reen) e blu (**B**lue). Una generica emissione luminosa può essere rappresentata da una somma opportunamente pesata delle tre componenti precedenti.

Il processo di percezione dei colori è un procedimento complesso per cui i sistemi automatici analizzano il colore a livello di pixel ignorando il colore di quelli adiacenti al pixel analizzato.

Lo spazio rappresentato dai valori RGB è anche quello utilizzato all'interno dei più comuni formati di memorizzazione delle immagini dove, a ciascun canale, è associato lo spazio di memoria di un byte, che permette di rappresentare più di un milione di colori differenti come combinazione dei 256 valori assegnabili ad ogni canale. Rappresentare ogni immagine $I[x, y]$ in questo spazio è necessario ai fini della visualizzazione, mentre ai fini dell'estrazione delle feature basta uno spazio di dimensione minore, per cui lo spazio iniziale viene trasformato dalla funzione T_c in uno spazio più adeguato e quantizzato, dalla funzione Q_c, in un numero minore di valori. Le proprietà che lo spazio trasformato e quantizzato deve verificare per essere adeguato alle operazioni di estrazione delle feature sono le seguenti:

1. **Uniformità:** le misure di distanza tra colori sono strettamente legate alla loro similitudine ottica.

   Come sappiamo infatti la similarità dei colori delle immagini vengono calcolate in base a funzioni di distanza per cui la trasformazione che porta al nuovo spazio dei colori deve essere tale da rendere semplice il calcolo della similitudine. Un modo per ottenere ciò è far si che la distanza tra due colori non sia funzione della loro posizione assoluta, ma solo di quella relativa.

2. **Completezza:** lo spazio dei colori deve contenere tutti i colori percettivamente distinti. Devono essere rappresentati tutti i colori che visivamente risultano diversi.

3. **Compattezza:** ogni colore nello spazio deve essere percettivamente distinto dagli altri colori. Il nuovo spazio, risultante dalla trasformazione e dalla quantizzazione, non deve essere ridondante in modo da ridurre la complessità della rappresentazione delle feature sui colori ed il calcolo delle dis-

tanze. Queste due condizioni fanno si che il nuovo spazio contenga solo i colori necessari e sufficienti all'adeguata rappresentazione delle feature.

4. **Naturalezza:** lo spazio dei colori deve fornire una naturale suddivisione dei colori nei tre principali attributi luminosità, tinta e saturazione:

   - **Luminosità:** attributo che rappresenta la sensazione fisica associata alla percezione di una maggiore o minore quantità di luce.
   - **Tinta:** attributo che rappresenta la similitudine cromatica del colore dato con uno o due dei quattro colori principali: rosso, verde, giallo e blu.
   - **Saturazione:** attributo che rappresenta la sensazione fisica associata alla quantità di tinta presente all'interno di un dato colore. La saturazione permette di esprimere un giudizio sul grado in cui una luce colorata differisce da una non colorata indipendentemente dalle rispettive luminosità.

Lo spazio RGB soddisfa solamente la proprietà di completezza, mentre non valgono quelle di uniformità, compattezza e naturalezza.

Definiamo una serie di trasformazioni che permettono di ottenere a partire da un punto $\vec{v_c}$ nello spazio RGB un punto $\vec{w_c}$ in un nuovo spazio che cerca di soddisfare il maggior numero di proprietà tra quelle precedentemente elencate.

2.1.1 Lo spazio di colore OPP.

Un nuovo spazio, con una trasformazione molto semplice da definire, è lo spazio di colore opponente OPP che, tramite una trasformazione lineare, mette in luce il fatto che all'interno dell'occhio umano i singoli canali luminosi vengono combinati in due canali composti.

$$\begin{aligned} WB &= R + G + B \\ BY &= -R - G + 2B \\ RG &= R - G + 2B \end{aligned} \tag{2.1}$$

La prima equazione rappresenta la luminosità, mentre le altre due rappresentano le informazioni cromatiche che non corrispondono a tinta e saturazione, ma più a blu *vs* giallo e rosso *vs* verde.

I vantaggi dello spazio OPP sono la semplicità e la completezza della trasformazione, ma lo spazio non è nè uniforme nè naturale e non è possibile definire metriche robuste per calcolare la similarità tra i colori.

Ad esempio Swain e Ballard [**?**] utilizzano questo sistema quantizzato a 2048 elementi per il loro sistema di ricerca di immagini.

2.1.2 Lo spazio di colore di Munsell.

Un'altra trasformazione è quella che permette di ottenere lo spazio di colore di Munsell.

Munsell ha realizzato una raccolta di 1200 piastrine colorate ad ognuna delle quali ha assegnato un valore di luminosità (Hue), saturazione (Saturation) e tinta (Chroma). Le varie piastrine sono disposte in modo che la distanze tra due colori adiacenti sia sempre la stessa e unitaria.

Lo spazio di colore di Munsell (HSC) è sicuramente compatto, completo e naturale, ma non è uniforme. Il problema con questo spazio è che Munsell non ha definito quali equazioni per la trasformazione o quantizzazione sono necessarie per ottenere il suo spazio partendo da quello RGB.

Tramite la MTM (Trasformata Matematica su Munsell) viene definita una procedura non lineare e non invertibile che permette di approssimare lo spazio di Munsell (HSC) a partire da quello RGB [**?**].

Ad esempio QBIC [**?**] in un primo momento quantizza lo spazio RGB a 4096 colori, poi ogni colore è trasformato tramite la MTM nello spazio di Munsell che viene poi ulteriormente ridotto determinando i 64 o 256 colori più rappresentativi attraverso un adeguato algoritmo di clustering.

2.1.3 Lo spazio di colore HSV.

Un'ulteriore trasformazione permette di passare allo spazio HSV che codifica i colori tramite i valori di tinta (Hue), saturazione (Saturation) e luminosità (Value).

Anche questa trasformazione è non lineare, ma è anche facilmente invertibile. Lo spazio HSV è naturale ed approssimativamente uniforme. Tramite un'adeguata quantizzazione risulta anche compatto e completo. La trasformazione T_c che permette di ottenere il vettore $\vec{w_c} = (h, s, v)$ a partire da $\vec{v_c} = (r, g, b)$ è definita dalle seguenti equazioni:

Trasformazione da RGB a HSV:

$$\begin{array}{ll} v = \max(r,g,b) & s = \frac{v-\min(r,g,b)}{v} \\ r' = \frac{v-r}{v-\min(r,g,b)} \quad g' = \frac{v-g}{v-\min(r,g,b)} & b' = \frac{v-b}{v-\min(r,g,b)} \end{array} \tag{2.2}$$

$$6h = \begin{cases} 5 + b' & se\ r = \max(r,g,b)\ e\ g = \min(r,g,b) \\ 1 - g' & se\ r = \max(r,g,b)\ e\ g \neq \min(r,g,b) \\ 1 + r' & se\ g = \max(r,g,b)\ e\ b = \min(r,g,b) \\ 3 - b' & se\ g = \max(r,g,b)\ e\ b \neq \min(r,g,b) \\ 3 + g' & se\ b = \max(r,g,b)\ e\ r = \min(r,g,b) \\ 5 - r' & altrimenti \end{cases} \tag{2.3}$$

Trasformazione inversa da HSV a RGB:

$$\begin{array}{ll} \alpha = 6h/round(6h) & \omega_1 = v*(1-s) \\ \omega_2 = v*((1-s)*\alpha) & \omega_3 = v*(1-(s*(1-\alpha))) \end{array} \tag{2.4}$$

$$r = \begin{cases} v & se\ \alpha = 0\ o\ \alpha = 5 \\ \omega_1 & se\ \alpha = 2\ o\ \alpha = 3 \\ \omega_2 & se\ \alpha = 1 \\ \omega_3 & se\ \alpha = 4 \end{cases} \quad g = \begin{cases} v & se\ \alpha = 1\ o\ \alpha = 2 \\ \omega_1 & se\ \alpha = 4\ o\ \alpha = 5 \\ \omega_2 & se\ \alpha = 3 \\ \omega_3 & se\ \alpha = 0 \end{cases} \quad b = \begin{cases} v & se\ \alpha = 3\ o\ \alpha = 4 \\ \omega_1 & se\ \alpha = 0\ o\ \alpha = 1 \\ \omega_2 & se\ \alpha = 5 \\ \omega_3 & se\ \alpha = 2 \end{cases} \tag{2.5}$$

Lo spazio HSV può essere agevolmente rappresentato da un cilindro dove la quota sull'asse verticale che attraversa le due basi rappresenta la luminosità (Value) cioè la quantità di bianco o nero presente nel colore. La distanza dall'asse verticale rappresenta la saturazione (Saturation) cioè la quantità di colore presente. L'angolo di rotazione rispetto ad un dato piano che giace sull'asse verticale rappresenta la tinta (Hue) dove i tre colori principali R,G,B distano tra di loro di 120.

Lo spazio HSV è impiegato in VisualSEEk [**?**] con una quantizzazione Q_c a 166 colori ottenuti assegnando alla tinta 18 possibili valori, mentre alla saturazione e alla luminosità vengono assegnati solamente 3 valori. Questa quantizzazione permette di ottenere $18 * 3 * 3 = 162$ possibili tonalità di colore a cui vengono aggiunte anche 4 tonalità di grigio per un totale di 166 elementi.

2.1.4 Istogramma dei colori.

Le feature sul colore più conosciuta ed impiegata da quasi tutti i sistemi è l'istogramma dei colori che consiste in un vettore di K interi, dove K è il numero di colori ottenuti dopo la trasformazione e la quantizzazione dello spazio iniziale (Figura 2.1).

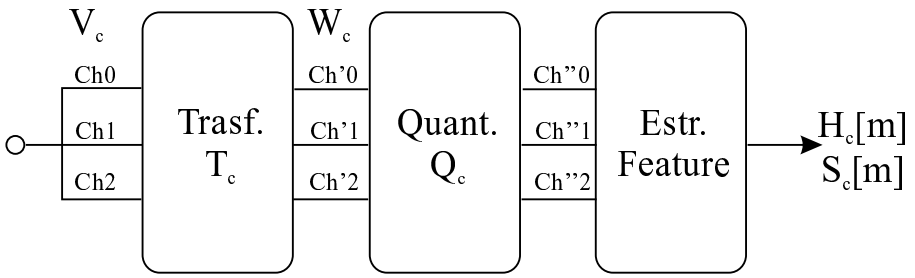


Figura 2.1: Schema a blocchi dei passi di trasformazione T_c e quantizzazione Q_c necessari all'estrazione dell'istogramma dei colori.

L'istogramma dell'immagine viene calcolato contando le occorrenze di ognuno dei K colori contenuti all'interno dell'immagine. Data un'immagine nello spazio RGB $I[x, y]$ applichiamo la trasformazione T_c e la quantizzazione Q_c ottenendo uno spazio a K colori, allora l'm-esimo elemento dell'istogramma $\vec{H}_c[0, \ldots, (K-1)]$ è determinato nella maniera seguente:

$$H_c[m] = \sum_{i=0}^{NR-1} \sum_{j=0}^{NC-1} \begin{cases} 1 & se\ Q_c(T_c(I[i,j])) = m \\ 0 & altrimenti \end{cases} \tag{2.6}$$

Un'ulteriore elaborazione dell'istogramma dei colori permette di ottenere una nuova feature ancora più compatta detta Color Set. Il Color Set è implementato attraverso un vettore di K elementi binari, per cui è molto più compatto, dove ad ogni elemento viene associato il valore 1 se il corrispondente elemento nell'istogramma contiene più di τ_m elementi, con τ_m fissato dal sistema, e 0 in caso contrario.

$$S_c[m] = \begin{cases} 1 & se\ H_c[m] \geq \tau_m \\ 0 & altrimenti \end{cases} \tag{2.7}$$

I Color set vengono impiegati insieme ad algoritmi di estrazione delle regioni all'interno di VisualSEEk [**?**]. Infatti i Color set vengono utilizzati per rappresentare il colore predominante delle regioni scartando tutti quelli che non sono adeguatamente rappresentati.

	Spazi di colore			
Proprietà	RGB	OPP	Munsell	HSV
Uniformità	NO	NO	NO	SÌ
Compattezza	NO	NO	SÌ	SÌ
Completezza	SÌ	SÌ	SÌ	SÌ
Naturalezza	NO	NO	SÌ	SÌ
Trasformazione	N/A	LINEARE	NON LINEARE	NON LINEARE

Tabella 2.1: Confronto delle caratteristiche peculiari dei diversi spazi di colore descritti.

2.2 Feature di tessitura.

Con il termine tessitura (texture) si definiscono quei raggruppamenti di pixel che con la loro disposizione spaziale o per il loro motivo uniforme non sono sufficientemente rappresentati dal solo attributo colore.

Le tessiture possono consistere nella disposizione periodica o casuale di elementi fondamentali, ma possono anche non presentare alcuna strutturazione. Modellare analiticamente una tessitura è uno dei compiti più difficili della visione artificiale.

Le tessiture sono uno degli elementi più importanti nella definizione di una scena, in particolare forniscono indizi sulla profondità della scena oppure sull'orientamento nello spazio delle superfici; inoltre vengono efficacemente impiegate, nelle immagini da satellite, per riconoscere la natura della superficie terrestre. È quindi possibile distinguere monti, pianure, tipi di raccolti e agglomerati urbani.

I sistemi di visualizzazione grafica ottengono risultati di elevato realismo quando sulle superfici tridimensionali vengono proiettate le rispettive texture. Infine le tessiture descrivono il contenuto di immagini reali come nuvole, foglie, mattoni e tessuti. Un ulteriore problema consiste nel riconoscere le tessiture indipendentemente dalle distorsioni prospettiche e dalle variazioni di luminosità.

2.2.1 Caratterizzazione diretta.

Uno dei primi approcci nel riconoscimento delle tessiture implementato nel sistema QBIC consiste nell'estrarre una serie di parametri calcolati sul canale di luminosità dello spazio di colore dell'immagine: in particolare il sistema estrae una versione modificata di alcuni dei parametri introdotti da Tamura in [**?**] come rugosità (Coarseness), contrasto (Contrast) e direzionalità (Directionality).

- La rugosità misura la scala della tessitura ed è facilmente determinata attraverso una serie di finestre mobili di diverse dimensioni.
- Il contrasto determina le variazioni di luminosità dell'immagine ed è calcolato attraverso la varianza dell'istogramma a toni di grigio.
- La direzionalità determina l'esistenza di una direzione preferenziale oppure l'eventuale omogeneità ed è derivata dall'analisi della distribuzione della direzione del gradiente dell'immagine in una serie di punti equamente distribuiti.

2.2.2 La trasformata Wavelet.

Una recente evoluzione nell'analisi delle tessiture introduce una trasformazione dello spazio dei punti dell'immagine che porta in un nuovo spazio dove la determinazione delle proprietà salienti è più evidente. Questa trasformazione prende il nome di trasformata Wavelet.

Attraverso la trasformata Wavelet proprietà in apparenza diverse come il colore, la tessitura e la forma possono essere efficacemente caratterizzate in maniera

omogenea. Questo approccio fa si che non debbano essere definite procedure diverse per l'esecuzione di query sulle feature estratte, procedure che presentano metriche di confronto diverse, prestazioni diverse nel recupero delle immagini ed in particolare impediscono la definizione di query che specificano nei criteri di recupero contemporaneamente sia informazioni sul colore che sulla tessitura o sulla forma.

La trasformata wawelet è una tecnica di analisi tempo/frequenza detta anche analisi multirisoluzione che trova applicazione in problemi di compressione, riconoscimento di bordi ed analisi di elementi di tessitura di immagini. Le wawelet sono funzioni matematiche in grado di fornire una rappresentazione multirisoluzione del segnale molto utile nell'analisi del contenuto dell'immagine; sono infatti in grado di decomporre il segnale in diverse componenti di frequenza e studiare ogni componente con diversa risoluzione e scala.

Forniscono inoltre robuste informazioni relativamente all'intensità luminosa e sono in grado di catturare efficientemente proprietà di tessitura e forma. L'idea alla base delle wawelet è la stessa presente nella trasformata di Fourier, cioè approssimare attraverso la sovrapposizione di funzioni. Il problema principale legato all'analisi di Fourier è che le funzioni base impiegate nell'approssimazione sono seno e coseno che per la loro natura non sono locali, in quanto si estendono su tutto l'asse reale e non sono pertanto in grado di approssimare dettagli fini.

L'innovazione portata dalle wawelet è tale per cui le funzioni approssimanti sono contenute in un dominio finito che le rende particolarmente idonee alla rappresentazione di segnali contenenti forti discontinuità.

Esistono diversi tipi di WT (Wavelet Trasfomation); la principale distinzione riguarda:

- Trasformata di Wavelet continua (CWT)
- Trasformata di Wavelet discreta (DWT)

La DWT è particolarmente utile nel caso di analisi, computazione e compressione di immagini.

La CTW decompone un segnale 1D, $f(x)$, in un insieme di funzioni Wavelet base:

$$W_a^b(f) = \int f(x)\psi_{a,b}^*(x)dx \tag{2.8}$$

Tale base, che normalmente è completa e ortogonale, è ottenuta traslando e dilatando una singola "mother Wavelet" $\psi(x)$:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right) \tag{2.9}$$

La “mother Wavelet” $\psi(x)$ è localizzata sia nel dominio spaziale che nel dominio delle frequenze e deve soddisfare la condizione di avere media nulla:

$$\int \psi(x)dx = 0 \tag{2.10}$$

Un esempio di “mother Wavelet” è la funzione base di Harr (Figura 2.2) che definisce la trasformata di Harr ed è il più semplice ed antico esempio di Wavelet ortonormale a supporto compatto.

La funzione base di Harr può essere espressa nella seguente formulazione:

$$\psi(x) = \begin{cases} 1 & 0 \leq x < 1/2 \\ -1 & 1/2 \leq x < 1 \\ 0 & \text{altrimenti} \end{cases} \tag{2.11}$$

L’algoritmo di trasformazione adotta un procedimento di scomposizione che

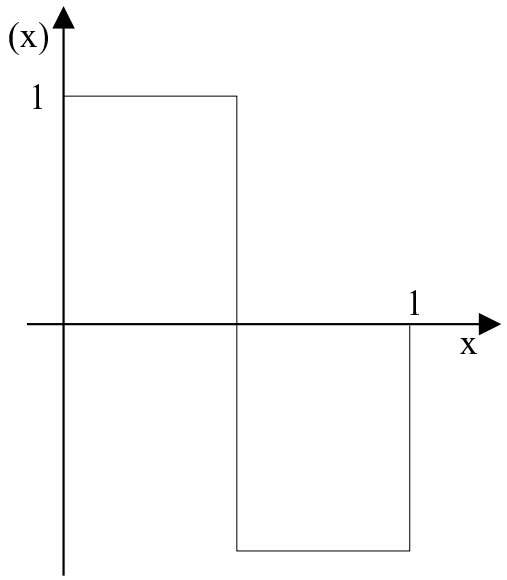


Figura 2.2: Funzione base di Harr.

utilizza due sottobande in maniera iterativa.

Dopo il primo passo di codifica in sottobande (costituito da una fase di filtraggio ed una fase di sottocampionamento), dal quale si ottengono G_0 e G_1, la componente a bassa frequenza, G_0, è nuovamente filtrata e sottocampionata; l’iterazione continua mantenendo le componenti ad alta frequenza e codificando quelle a bassa frequenza fino ad ottenere un unico segnale a bassa frequenza.

Considerando un segnale di input discreto $x = (x_0, x_1, \ldots, x_{N-1})$ avente lunghezza $N = 2^L$ il processo di trasformazione è compiuto attraverso i seguenti passi:

Livello 1: si considerino i primi due punti campione x_0, x_1 e siano a_0^1 la loro somma e d_0^1 la loro differenza; si proceda allo stesso modo per tutte le restanti coppie di punti (x_{2i}, x_{2i+1}). Da tali punti si otterranno i valori:

$$\begin{aligned} a_i^1 &= C(x_{2i} + x_{2i+1}) \\ d_i^1 &= C(x_{2i} - x_{2i+1}) \end{aligned} \tag{2.12}$$

dove C è una costante di proporzionalità.

I valori $a_i^1 (0 \leq i < N/2)$ rappresentano la parte a bassa frequenza del segnale ('smooth') mentre i valori d_i^1 costituiscono l'alta frequenza dello stesso.

Livello 2: si considerino i valori a_i^1 (parte smooth del segnale) e su di essa si procede come nel passo precedente; si ottengono allo stesso modo, i valori a_i^2, componente a bassa frequenza del segnale, e $d_i^2 (0 \leq i < N/4)$.

... si ripeta ricorsivamente il procedimento fino ad avere un unico segnale smooth di lunghezza unitaria.

La trasformata di Harr del segnale originale x è data dall'insieme dei valori di 'differenza' d_i^l ad ogni livello l più la componente 'smooth' a_i^l dall'ultimo livello $L = \log_2 N$.

Per chiarire meglio cosa succede consideriamo un segnale discreto x di dimensione $N = 2^L$ campionato nell'intervallo Δt. Al primo passo di trasformazione la Wavelet spezza x in due sottovettori $\vec{a_1}, \vec{d_1}$ ognuno di lunghezza N/2 sull'intervallo di campionamento $2\Delta t$. Il vettore $\vec{d_1}$ è mantenuto e rappresenta la prima porzione della trasformata di x, il processo di suddivisione continua su $\vec{a_1}$ producendo $\vec{a_2}, \vec{d_2}$; la suddivisione continua sino all'L-esimo passo nel quale $\vec{a_L}, \vec{d_L}$ consistono in un solo punto ed il vettore $\vec{x}$ può essere rimpiazzato dai vettori $(d_1, d_2, \ldots, d_L, a_L)$. Ad ogni passo la somma delle lunghezze di due sotto vettori ottenuti è pari alla lunghezza del vettore padre, così la somma degli L vettori $(d_1, d_2, \ldots, d_L)$ più il punto finale a_L è pari a N lunghezza del segnale x. L'insieme dei valori indicati con a è essenzialmente la media (averaging) dei dati del precedente livello (Figura 2.3). Nel dominio delle frequenze ciò equivale ad applicare un filtro passa basso (low-pass filter) alle informazioni del precedente livello. L'insieme dei valori indicati con d è la differenza dei dati del passo precedente e la media dei dati al passo attuale. Nel dominio delle frequenze questa operazione equivale all'applicazione di un filtro passa-alto (high-pass filter) al livello precedente.

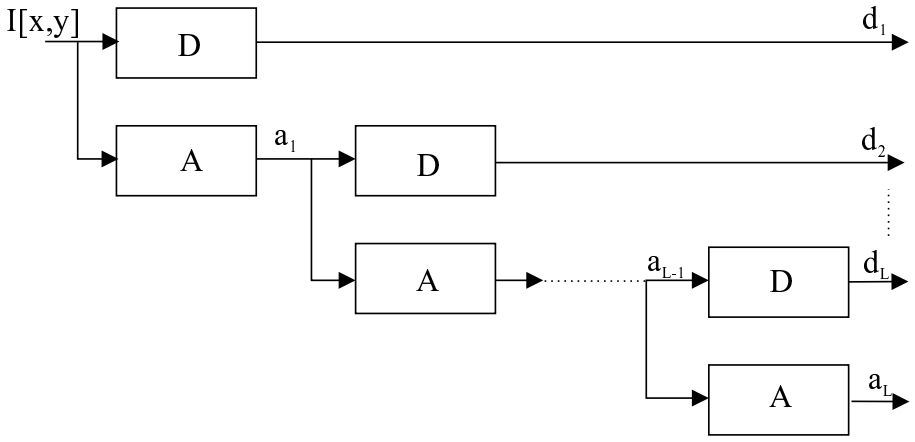


Figura 2.3: Rappresentazione Filter Bank FB di una decomposizione Wavelet 1D.

L'operazione di trasformazione può essere vista come l'applicazione in cascata di una serie di moduli di Filter Bank (FB). Un Filter Bank è un insieme di filtri passa banda che consente di decomporre il segnale in componenti di diversa frequenza. Il segnale di input $f(x)$ è fatto passare in parallelo attraverso tali filtri e le funzioni $g_i(x)$ ottenute in uscita da ogni singolo blocco sono tali che la loro sommatoria permette di riottenere la funzione $f(x)$ di ingresso.

L'idea alla base della trasformata Wavelet è quella di suddividere il piano delle frequenze in un insieme di intervalli disgiunti ed usare tali suddivisioni per definire una serie di funzioni passa-banda che permettano di esaltare le componenti di interesse.

La funzione di Harr ha il principale svantaggio di essere discontinua, ciò la rende inadeguata all'approssimazione di funzioni smooth; una delle più popolari trasformate Wavelet, nota come Wavelet di Daubechies è in grado di risolvere questi problemi.

Le Wavelet di Daubechies forniscono importanti risultati nell'analisi e nella sintesi di immagini; tali trasformate infatti sono simili ad una media pesata che meglio preserva le informazioni memorizzate nel segnale se si considera solo la componente a bassa frequenza.

Le immagini possono essere rappresentate come segnali 2D che vengono gestiti, in maniera analoga, attraverso la trasformata Wavelet bidimensionale che ne fornisce una rappresentazione in sottobande di frequenza.

Nel caso 2D il dominio temporale è rappresentato dalla locazione spaziale dei pixel, mentre il dominio delle frequenze è rappresentato dalla variazione di colore tra i punti. La costruzione di una base ortonormale per uno spazio bidimensionale prende come punto di partenza il caso monodimensionale, per mezzo del quale

è possibile definire la funzione prototipo come il prodotto di due funzioni base monodimensionali:

$$\psi_{i_1,j_1;i_2,j_2}(x_1, x_2) = \psi_{i_1,j_1}(x_1)\psi_{i_2,j_2}(x_2) \tag{2.13}$$

La famiglia di funzioni ottenute variando i e k nell'insieme dei numeri relativi $\mathcal{Z}$ definisce una base ortonormale in $L^2(\mathcal{R}^2)$, nella quale le due variabili x_i sono dilatate separatamente.

Data un'immagine avente dimensione (N x M), dove M ed N sono potenze di 2, ogni passo di trasformazione decompone il segnale in quattro sotto-immagini di dimensioni (N/2 x M/2), ognuna delle quali rappresenta una parte del piano delle frequenze. Ogni sotto-immagine è ottenuta applicando i passi di trasformazione, sopra descritti, alle righe ed alle colonne dell'immagine di input.

In particolare indichiamo con:

- LL: Low–Low nella direzione orizzontale e verticale;
- LH: Low nella direzione verticale High nella direzione orizzontale;
- HL: High nella direzione verticale Low nella direzione orizzontale;
- HH: High–High nella direzione orizzontale e verticale.

Dove Low e High stanno per l'applicazione di un filtro passa-basso cioè una somma di elementi adiacenti e di un filtro passa-alto cioè una differenza di elementi adiacenti rispettivamente e le direzioni verticali ed orizzontali corrispondono alla direzione di righe e colonne su cui si considera l'adiacenza.

Il successivo livello di trasformazione considera la sotto-immagine formata dalla banda LL e la scompone allo stesso modo in ulteriori quatto bande ottenendo immagini di dimensione (N/4 x M/4).

Il processo di trasformazione può essere eseguito per un numero massimo di livelli pari a $L \leq \log_2(\min(N, M))$.

Sia le proprietà di colore che di tessitura in Windsurf [**?**] possono essere caratterizzate in maniera efficace attraverso la trasformata Wavelet, come di seguito descritto.

La WT contiene informazione relative ad ogni canale di colore: infatti, data un'immagine in input definita su di un particolare spazio di colore, il processo di trasformazione è tale per cui ogni canale trasformato risulta essere diviso in sottobande a frequenza diversa.

Si consideri un'immagine definita, ad esempio, sullo spazio HSV. Se si suppone di applicare la WT al livello 1 si otterranno quattro sottobande di frequenza per ogni canale. In particolare si avrà una rappresentazione di ogni sottobanda (LL,LH,HL,HH) per ogni canale H, S e V.

Se si suppone inoltre di trasformare fino al secondo livello, allora la sottobanda LL del livello 1, di ogni canale, sarà ulteriormente suddivisa nelle quattro componenti di frequenza e così via (Figura 2.4). Quindi ogni coefficiente dell'immagine trasformata altro non è che la rappresentazione di un particolare canale di colore e in una particolare banda di frequenza di un particolare punto dell'immagine.

Indichiamo il j-esimo coefficiente Wavelet della sottobanda S ($S \in \{LL, LH, HL, HH\}$) del livello l ($l \leq L$) di trasformata con il simbolo:

$$w_j^{l;S} = (w_{0j}^{l;S}, w_{1j}^{l;S}, w_{2j}^{l;S}) \tag{2.14}$$

dove $w_{c,j}^{l;S}$ rappresenta il coefficiente relativo al canale c ($c \in \{0, 1, 2\}$).

Le caratteristiche riguardanti le tessiture delle immagini sono ottenute misurando l'energia dei coefficienti della Wavelet nelle diverse sottobande di frequenza. Questo tipo di approccio è lo stesso utilizzato da Smith [**?**] nella rappresentazione della tessitura ed è noto essere il metodo più robusto per la rappresentazione delle feature di tessitura.

L'idea di base di questo approccio è quella di poter rappresentare congiuntamente informazioni di colore e di tessitura, per tale ragione si fornisce un approccio che utilizza una formulazione in grado di contenere informazioni sia sull'energia di un punto definito su di un particolare canale di colore che proprietà di correlazione fra punti relativi a canali non omonimi. Definiamo quindi l'energia sui canali c, d ($c, d \in \{0, 1, 2\}$) di un punto j di una sottobanda S di un dato livello di trasformata l come:

$$e_{cd,j}^{l;S} = w_{c,j}^{l;S}\ w_{d,j}^{l;S} \tag{2.15}$$

In particolare per $c = d$ tale formulazione rappresenta l'energia del canale, per $c \neq d$, viceversa, si parla di energia di correlazione o cross-energy tra i canali c e d.

Definiamo pertanto:

$$e_j^{l;S} = \left(e_{00,j}^{l;S}, e_{01,j}^{l;S}, e_{02,j}^{l;S}, e_{11,j}^{l;S}, e_{12,j}^{l;S}, e_{22,j}^{l;S}\right) \tag{2.16}$$

il vettore contenente le energie e le cross-energy sui diversi canali di colore.

2.2.3 Istogramma della tessitura.

VisualSEEk [**?**] invece va oltre la trasformata Wavelet introducendo ulteriori elaborazioni che permettono di caratterizzare le tessiture in uno spazio a 9 dimensioni.

In particolare ogni immagine è elaborata attraverso la trasformata Wavelet iterata tre volte sulla banda LL (Figura 2.5). Il passo seguente consiste nell'ap-

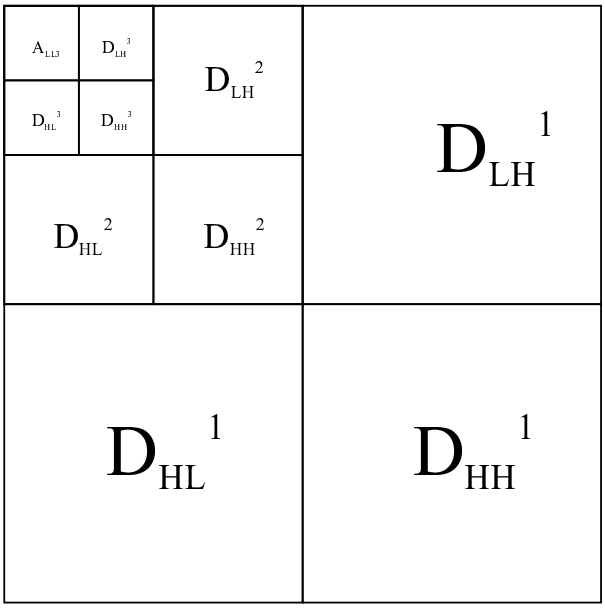


Figura 2.4: Disposizione delle sottobande di frequenza di un'immagine trasformata attraverso la WT iterata fino al terzo livello.

plicazione del generatore di canali di tessitura TGC (Texture Channel Generator) che per ognuna delle nove sottobande ottenute attraverso la trasformata Wavelet determina l'energia, esegue il sovracampionamento dei pixel portando ciascuna sottobanda alle dimensioni dell'immagine originale ed infine esegue un filtraggio globale.

I valori delle energie sono sovracampionati alla loro dimensione originaria inserendo degli zeri. I punti mancanti sono reinseriti utilizzando i filtri di blocco che eseguono una semplice replicazione dei pixel ottenendo i canali di tessitura S_k.

Al termine di questa operazione otteniamo un valore dell'energia per ogni canale di tessitura ottenendo una rappresentazione analoga a quella del colore.

Mentre per caratterizzare il colore di un'immagine, ad esempio nello spazio HSV, si utilizzano tre matrici di numeri interi che memorizzano i tre canali H, S e V, per le tessiture ogni punto dell'immagine è rappresentato da un vettore nello spazio a 9 dimensioni. Un punto di tessitura è del tipo:

$$\vec{v_t} = \{S_0, S_1, \ldots, S_{K-1}\} \tag{2.17}$$

Dove S_k rappresenta l'energia del k-esimo canale ottenuto iterando tre volte la trasformata Wavelet sulla sottobanda LL. La fase successiva, come nel caso del colore, consiste nell'applicazione di un passo di trasformazione che permette di

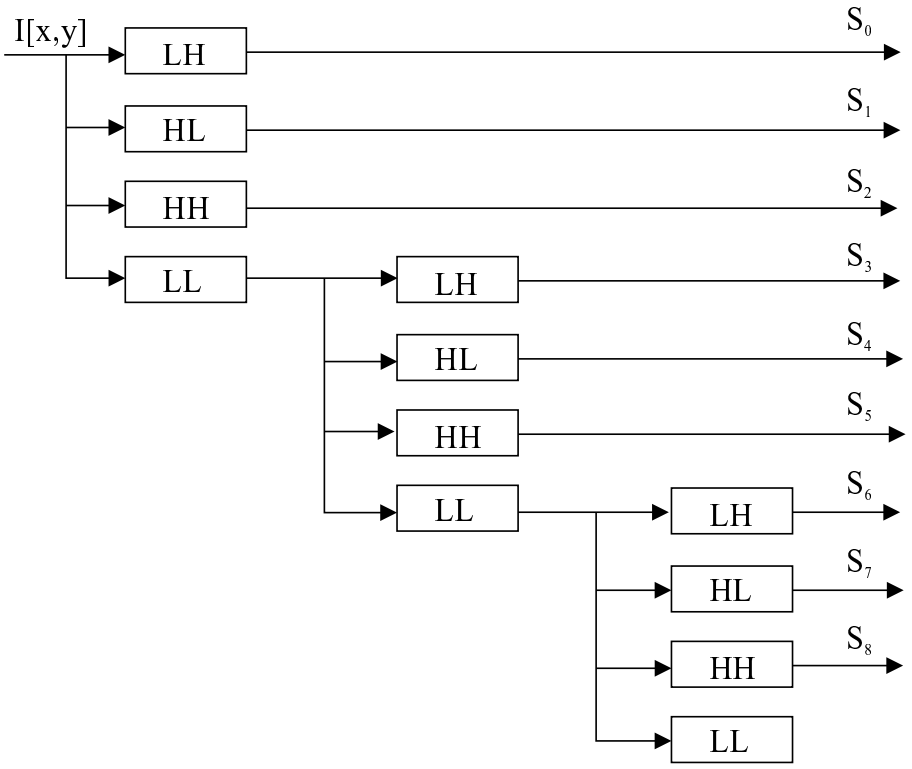


Figura 2.5: Rappresentazione Filter Bank FB della decomposizione W avelet 2D implementata in VisualSEEk

riorganizzare e raggruppare gli elementi per creare un insieme di elementi di tessitura.

Per ogni punto v_t la trasformazione T_t permette di ottenere il punto trasformato w_t secondo l'equazione:

$$\vec{w_t} = T_t \vec{v_t} \tag{2.18}$$

In molti casi è auspicabile garantire l'invarianza delle feature estratte dalle tessiture da modificazioni come la rotazione e lo scaling.

È possibile definire le trasformazioni T_t^R e T_t^S che permetto di ottenere l'invarianza delle feature da rotazione e scaling attraverso matrici binarie di dimensioni (3 x 9) del tipo:

$$T_t^R = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2.19}$$

$$T_t^S = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.20}$$

Dopo l'applicazione delle trasformazione segue il passo di quantizzazione che permette di discretizzare il dominio dei possibili valori in modo che siano più facilmente memorizzabili.

La funzione di quantizzazione Q_t mappa ciascuno dei possibili valori di w_t in uno degli N valori codificati ed indicizzati da n.

Per cui n è l'indice $n \in \{0, \ldots, N-1\}$ dell'elemento di tessitura w_t assegnato dalla funzione di quantizzazione:

$$y_n = Q_t(\vec{w_t}) \tag{2.21}$$

dove y_n rappresenta uno degli N possibili valori codificati.

Possiamo definire un elemento di tessitura come l'insieme di tutti gli elementi v_t che sono assegnati allo stesso indice n dal processo di trasformazione e quantizzazione.

Al termine di una generica fase di elaborazione delle feature sulla tessitura si ottiene un set di N elementi di tessitura più o meno rilevanti per la caratterizzazione dell'immagine.

Dagli elementi di tessitura si può determinare molto agevolmente il relativo istogramma di tessitura, con un significato del tutto analogo a quello del colore, semplicemente assegnando a ciascun elemento dell'istogramma il numero di punti diversi da 0 del corrispondente elemento di tessitura (Figura 2.6).

Dato che la funzione di quantizzazione di VisualSEEk determina solamente due possibili livelli di quantizzazione l'istogramma consiste di $2^9 = 512$ possibili elementi pari a 512 elementi di tessitura.

La funzione di quantizzazione si occupa di minimizzare l'errore introdotto dall'assegnazione di un valore qualunque ad uno dei due possibili livelli di quantizzazione in particolare si cerca di minimizzare il valore medio dell'errore al quadrato:

$$\varepsilon = E[(x - x')^2] = \int_{d_i}^{d_s} (x - x')^2 p(x) dx \tag{2.22}$$

dove d_l e d_s sono gli estremi di variazione della variabile x e $p(x)$ la sua funzione di densità di probabilità.

Dall'istogramma della tessitura è poi possibile definire, attraverso l'analogo confronto con una soglia, il relativo texture set che corrisponde ad un vettore di elementi binari dove gli elementi ad uno sono associati a valori dell'istogramma maggiori della soglia. In questa maniera i texture set vengono confrontati con le

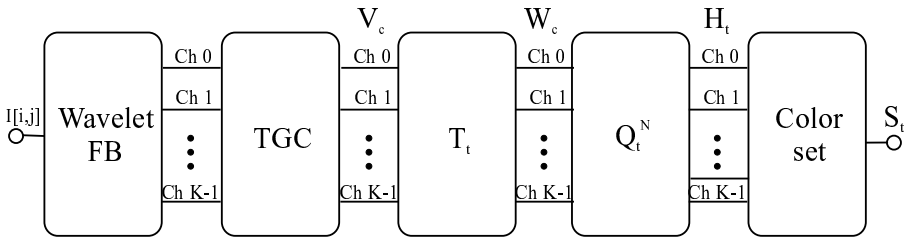


Figura 2.6: Schema a blocchi delle operazioni necessarie all'estrazione dell'istogramma delle tessiture e dei relativi texture set.

stesse funzioni di distanza definite per il confronto dei color set.

In particolare query che specificano sia regioni di colore che di tessitura possono essere trattate dalla stessa procedura.

2.3 Feature di forma.

Dopo l'estrazione, automatica, semiautomatica o manuale, delle regioni all'interno delle immagini si possono calcolare ulteriori feature che, in questo caso particolare, prendono il nome di descrittori. Questi descrittori possono coinvolgere la forma, la dimensione, la posizione e l'orientamento delle regioni.

Una proprietà auspicabile per i descrittori di forma "shape descriptor" è l'invarianza rispetto a traslazione, rotazione e scaling:

ciò significa che la proprietà, per un data regione, non deve variare o deve variare molto poco in funzione di rotazioni, traslazioni o variazioni di scala della regione considerata.

Per i descrittori relativi alle dimensioni della regione si vuole l'invarianza rispetto a traslazione e rotazione, mentre ovviamente, non è possibile avere quella rispetto allo scaling.

2.3.1 Descrittori d'area e posizione.

La posizione viene determinata univocamente dal baricentro che presenta ascissa pari alla somma delle ascisse di tutti i punti della regione divisa per il numero di punti ed ordinata pari alla somma delle ordinate di tutti i punti della regione sempre divisa per il numero di punti. L'area di una regione viene determinata contando il numero di pixel che la definiscono.

Uno dei principali descrittori di forma invece è la rettangolarità che misura quanto la regione è simile ad un rettangolo. La rettangolarità si ottiene calcolando

il rapporto tra l'area della regione e l'area del rettangolo minimo che racchiude la regione stessa.

La misura così definita non è invariante rispetto alla rotazione, per ottenere l'invarianza è necessario utilizzare un rettangolo minimo orientato come l'oggetto: bisogna cioè conoscere l'orientamento dell'oggetto.

2.3.2 Determinazione dell'orientamento.

Determiniamo l'orientamento della regione attraverso l'identificazione del suo asse maggiore. Definiamo asse maggiore della regione l'asse baricentrico rispetto al quale la regione ha il momento di inerzia minore. Una volta determinato l'asse maggiore l'orientamento viene espresso tramite l'angolo fra l'asse maggiore e il verso positivo dell'asse orizzontale.

Un primo metodo per la determinazione dell'asse maggiore consiste nel calcolare il momento di inerzia dell'oggetto rispetto alle rette passanti per il baricentro ed i punti del contorno. Tra tutte le rette possibili l'asse maggiore è quello che presenta il momento minore.

La generica retta passante per il baricentro $B(i_b, j_b)$ e per un punto del contorno $C(i_c, j_c)$ ha equazione:

$$\begin{array}{ccccc} r_{bc}: & ax + by + c = 0 & & \text{con} & \\ a = j_c - j_b & & b = i_b - i_c & & c = i_c\, j_b - i_b\, j_c \end{array} \tag{2.23}$$

Il momento di inerzia I rispetto a questa retta vale:

$$I = \sum_{P(i,j) \in C} d_P^2 \tag{2.24}$$

dove con d_P si intende la distanza del punto P dalla retta r_{bc} espressa come:

$$d_P = \frac{|ai + bj + c|}{\sqrt{a^2 + b^2}} \tag{2.25}$$

Questo è l'approccio più intuitivo e basato sulla forza bruta che richiede un notevole sforzo computazionale.

Il metodo migliore per la determinazione dell'asse maggiore consiste nel calcolare esplicitamente i soli momenti rispetto agli assi coordinati di un sistema di riferimento baricentrico e valutare poi i momenti rispetto alle direzioni associate ai punti del contorno a partire da tali grandezze, portando così ad un alleggerimento in termini di complessità computazionale.

Eseguiamo la traslazione degli assi del sistema nel baricentro:

$$\begin{array}{l} \bar{\imath} = i - i_b \\ \bar{\jmath} = j - j_b \end{array} \tag{2.26}$$

Calcoliamo l'equazione della generica retta passante per il baricentro ed un punto del contorno nel nuovo sistema di riferimento:

$$\begin{cases} \bar{\imath} = \alpha\ \lambda \\ \bar{\jmath} = \beta\ \lambda \end{cases} \Rightarrow \bar{\imath} = \frac{\alpha}{\beta}\bar{\jmath} \ \Rightarrow \beta\bar{\imath} - \alpha\bar{\jmath} = 0 \ \Rightarrow \begin{cases} a = \beta \\ b = -\alpha \\ c = 0 \end{cases} \tag{2.27}$$

Come nel caso precedente calcoliamo d_P^2 e I:

$$d_P^2 = (\beta\bar{\imath} - \alpha\bar{\jmath})^2$$

$$I = \beta^2 \sum_{P(i,j)\in C} \bar{\imath}^2 - \alpha\beta \sum_{P(i,j)\in C} \bar{\imath}\bar{\jmath} + \alpha^2 \sum_{P(i,j)\in C} \bar{\jmath}^2 \tag{2.28}$$

Dove le tre sommatorie esprimo rispettivamente il momento rispetto all'asse j, il momento di deviazione e il momento rispetto all'asse i.

I passi principali per la determinazione dell'asse maggiore sono allora:

1. Determinazione del baricentro dell'oggetto;
2. Si calcolano i momenti $I_{\bar{\imath}}$, $I_{\bar{\jmath}}$, $I_{\bar{\imath}\bar{\jmath}}$ solo una volta indipendentemente dal punto del contorno considerato:

$$I_{\bar{\imath}} = \sum_{P(i,j)\in C} (j - j_b)^2 \quad I_{\bar{\jmath}} = \sum_{P(i,j)\in C} (i - i_b)^2 \quad I_{\bar{\imath}\bar{\jmath}} = \sum_{P(i,j)\in C} (j - j_b)(i/i_b) \tag{2.29}$$

3. Per ogni punto del contorno si determina il versore che definisce la retta r_{bc}:

$$\vec{v_c} = (\alpha_c, \beta_c) = \left(\frac{i_c - i_b}{\sqrt{(i_c - i_b)^2 + (j_c - j_b)^2}}, \frac{j_c - j_b}{\sqrt{(i_c - i_b)^2 + (j_c - j_b)^2}} \right)$$
$$I = \alpha_c^2 I_{\bar{\jmath}} - 2\alpha_c\beta_c I_{\bar{\imath}\bar{\jmath}} + \beta_c^2 I_{\bar{\jmath}} \tag{2.30}$$

4. Si sceglie come asse maggiore la retta r_{bc} per la quale I è minimo.

2.3.3 Determinazione del MER.

Il passo successivo consiste nel determinare le dimensioni del rettangolo minimo orientato come la regione (Minimum Enclosing Rectangle) (MER) che si determina considerando l'asse maggiore e minore della regione e congiungendo con coppie di rette parallele ai due assi i punti del contorno disposti a distanza massima rispetto ai due assi su lati opposti.

L'asse minore si ottiene tracciando la retta passante per il baricentro, ma ortogonale all'asse maggiore:

$$\begin{array}{ccc} r_{bc'}: & a'x + b'y + c' = 0 & con \\ a' = b & b' = -a & c' = a_j b - b i_b \end{array} \tag{2.31}$$

Ottenuti i due assi si devono determinare i due punti del contorno a distanza massima dall'asse maggiore su lati opposti $(C1, C2)$ ed i due punti del contorno situati a distanza massima dall'asse minore su lati opposti $(C3, C4)$.

La determinazione del lato dove si localizza il punto rispetto all'asse si ottiene considerando la classica formula della distanza di un punto da una retta, eliminando però il modulo e considerando il segno del risultato.

Il passo successivo consente di determinare le rette passanti per punti $C1, C2, C3, C4$ e parallele agli assi che delimitano il MER e i punti di intersezione di queste ultime che individuano i vertici del MER necessari per calcolo di lunghezza (L) e larghezza (W).

Dati $C_1(i_1, j_1), C_2(i_2, j_2), C_3(i_3, j_3), C_4(i_4, j_4)$ determiniamo le rette del MER le cui equazioni hanno coefficienti:

$$\begin{array}{cccccc} r: & a_n x + b_n y + c_n = 0 & con & & & \\ l_1: & a_{l_1} = a & & b_{l_1} = b & c_{l_1} = -(a i_1 + b j_1) & \\ l_2: & a_{l_2} = a & & b_{l_2} = b & c_{l_2} = -(a i_2 + b j_2) & \\ w_1: & a_{w_1} = a' & & b_{w_1} = b' & c_{w_1} = -(a' i_3 + b' j_3) & \\ w_2: & a_{w_2} = a' & & b_{w_2} = b' & c_{w_2} = -(a' i_4 + b' j_4) & \end{array} \tag{2.32}$$

Ed i vertici:

$$\begin{array}{ll} i_{V1} = \frac{b_{l_1} c_{w_1} - c_{l_1} b_{w_1}}{a_{l_1} b_{w_1} - b_{l_1} a_{w_1}} & j_{V1} = \frac{c_{l_1} a_{w_1} - a_{l_1} c_{w_1}}{a_{l_1} b_{w_1} - b_{l_1} a_{w_1}} \\ i_{V2} = \frac{b_{l_1} c_{w_2} - c_{l_1} b_{w_2}}{a_{l_1} b_{w_2} - b_{l_1} a_{w_2}} & j_{V2} = \frac{c_{l_1} a_{w_2} - a_{l_1} c_{w_2}}{a_{l_1} b_{w_2} - b_{l_1} a_{w_2}} \\ i_{V3} = \frac{b_{l_2} c_{w_1} - c_{l_2} b_{w_1}}{a_{l_2} b_{w_1} - b_{l_2} a_{w_1}} & j_{V3} = \frac{c_{l_2} a_{w_1} - a_{l_2} c_{w_1}}{a_{l_2} b_{w_1} - b_{l_2} a_{w_1}} \\ i_{V4} = \frac{b_{l_2} c_{w_2} - c_{l_2} b_{w_2}}{a_{l_2} b_{w_2} - b_{l_2} a_{w_2}} & j_{V4} = \frac{c_{l_2} a_{w_2} - a_{l_2} c_{w_2}}{a_{l_2} b_{w_2} - b_{l_2} a_{w_2}} \end{array} \tag{2.33}$$

A partire dal MER poi si possono ricavare le dimensioni dell'oggetto L e W:

$$\begin{array}{c} L = d_{V_1,V_2} = \sqrt{(i_{v_1} - i_{v_2})^2 + (j_{v_1} - j_{v_2})^2} \\ W = d_{V_1,V_3} = \sqrt{(i_{v_1} - i_{v_3})^2 + (j_{v_1} - j_{v_3})^2} \end{array} \tag{2.34}$$

2.3.4 Descrittori di forma.

Una volta determinate le dimensioni del rettangolo orientato come la regione si ottiene la seguente definizione di rettangolarità:

$$R = \frac{A}{LW} \tag{2.35}$$

Un'ulteriore descrittore di forma è l'ellitticità che misura quanto la regione è simile ad un'ellisse:

$$E = \frac{A}{A_{LW}} \quad \text{dove} \quad A_{LW} = \frac{\pi}{4} LW \tag{2.36}$$

dove A_{LW} rappresenta l'area dell'ellisse orientata come l'oggetto avente asse maggiore pari ad L ed asse minore pari a W. L'eccentricità invece fornisce una misura di quanto la regione è "allungata" ed è definita molto semplicemente attraverso il rapporto:

$$E_C = \frac{L}{W} \tag{2.37}$$

2.3.5 Momenti invarianti.

Una rappresentazione alternativa della forma di una regione può avvenire impiegando un insieme più o meno esteso di momenti.

I momenti vengono definiti a partire da un'immagine in cui sono messe in evidenza una serie di regioni estratte attraverso le elaborazioni di un algoritmo di segmentazione.

Supponendo che tutte le regioni all'interno dell'immagine presentino un'etichetta, cioè un numero intero diverso da 0, mentre lo 0 rappresenta lo sfondo dal quale non si eseguono elaborazioni, supponiamo di definire una funzione:

$$F_k(i,j) = \begin{cases} 1 & \text{se } (i,j) \in R_k \\ 0 & \text{altrimenti} \end{cases} \tag{2.38}$$

che restituisce un valore pari ad 1 se il pixel di coordinate (i,j) appartiene alla k-esima regione e 0 in caso contrario.

Si definisce momento mn-esimo della k-esima regione la grandezza:

$$K^k_{mn} = \sum_{i=0}^{NR-1} \sum_{j=0}^{NC-1} i^m j^n F_k(i,j) \tag{2.39}$$

con $m, n \geq 0$ e NR, NC rispettivamente il numero di righe e colonne dell'immagine da elaborare.

La quantità $m+n$ è detta ordine del momento. I momenti possono essere visti come una generalizzazione delle proprietà definite precedentemente.

In particolare si possono evidenziare le seguenti analogie:

$$
\begin{aligned}
&M_{00}^k = \textstyle\sum_{i=0}^{NR-1}\sum_{j=0}^{NC-1} F_k(i,j) = A && \text{(area)}\\
&M_{10}^k = \textstyle\sum_{i=0}^{NR-1}\sum_{j=0}^{NC-1} iF_k(i,j) \Rightarrow \frac{M_{10}}{M_{00}} = i_b && \text{(coordinata i del baricentro)}\\
&M_{01}^k = \textstyle\sum_{i=0}^{NR-1}\sum_{j=0}^{NC-1} jF_k(i,j) \Rightarrow \frac{M_{01}}{M_{00}} = j_b && \text{(coordinata j del baricentro)}\\
&M_{20}^k = \textstyle\sum_{i=0}^{NR-1}\sum_{j=0}^{NC-1} i^2F_k(i,j) = I_j && \text{(Momento di inerzia asse j)}\\
&M_{02}^k = \textstyle\sum_{i=0}^{NR-1}\sum_{j=0}^{NC-1} j^2F_k(i,j) = I_i && \text{(Momento di inerzia asse i)}\\
&M_{20}^k = \textstyle\sum_{i=0}^{NR-1}\sum_{j=0}^{NC-1} ijF_k(i,j) = I_{ij} && \text{(Momento di deviazione)}
\end{aligned}
\tag{2.40}
$$

Si può dimostrare che l'insieme (infinito) dei momenti determina univocamente la forma della regione.

È possibile quindi utilizzare un sottoinsieme di questi momenti per discriminare approssimativamente la forma delle regioni.

Questi momenti non sono invarianti rispetto alla traslazione ed alla rotazione. È possibile ottenere dei momenti invarianti rispetto alla traslazione, che vengono detti momenti centrali, considerando un sistema di riferimento baricentrico.

Effettuando la trasformazione nella nuova base si ottiene:

$$K'^k_{mn} = \sum_{i=0}^{NR-1}\sum_{j=0}^{NC-1} (i-i_b)^m (j-j_b)^n F_k(i,j) \tag{2.41}$$

I momenti centrali non sono ancora invarianti rispetto alla rotazione, si pensi, ad esempio, ai momenti di inerzia che sono minimo e massimo quando l'asse maggiore coincide con uno dei due assi di riferimento dell'immagine.

Per ottenere l'invarianza rispetto alla rotazione occorrerebbe un sistema di riferimento orientato secondo gli assi maggiore e minore dell'oggetto.

Hu ha mostrato che a partire dai momenti centrali è possibile definire delle combinazioni dei momenti centrali che risultano invarianti rispetto a rotazione e scaling.

Normalizziamo i momenti centrali:

$$V_{mn} = \frac{M'_{mn}}{(M_{00})^\alpha} \quad \text{con} \quad \alpha = \frac{m+n}{2} + 1 \tag{2.42}$$

Quindi a partire dai momenti centrali si definiscono i momenti invarianti:

$$\begin{aligned}
h_1 &= V_{20} + V_{02} \\
h_2 &= (V_{20} + V_{02})^2 + 4V_{11}^2 \\
h_3 &= (V_{30} + 3V_{12})^2 + (V_{03} + 3V_{21})^2 \\
h_4 &= (V_{30} + V_{12})^2 + (V_{03} + V_{21})^2 \\
h_5 &= (V_{30} - 3V_{12})(V_{03} - V_{21})((V_{30} - V_{12})^2 - 3(V_{03} - V_{21})^2) + \\
&\quad +(3V_{12} - V_{03})(V_{03} + V_{21})(3(V_{30} - V_{12})^2 - (V_{30} - V_{21})^2)
\end{aligned} \tag{2.43}$$

Hu dimostra l'invarianza degli h_i rispetto a traslazione, rotazione e scaling nel caso continuo.

Nell'applicazione pratica su immagini digitali queste grandezze si dimostrano generalmente poco variabili per versioni modificate della stessa forma [**?**].

Questa è la caratterizzazione che QBIC estrae automaticamente da tutte le regioni evidenziate dall'utente in maniera semiautomatica [**?**].

Riassunto delle feature Estratte dai vari sistemi	
Motore	Modalità di estrazione delle feature
QBIC	Feature sul Colore: Estrae un istogramma a K=64, 256 elementi ottenuti trasformando lo spazio RGB iniziale attraverso la MTM (Trasformazione Matematica nello spazio dei colori di Munsell) successivamente raggruppati in gruppi di colori simili. Feature sulla Forma: Estrae area, circolarità, eccentricità, orientazione dell'asse maggiore ed una serie di momenti invarianti per un totale di 20 elementi. Feature sulla Tessitura: Caratterizzazione attraverso rugosità, direzionalità e luminosità.
WindSurf	Approccio Unificante: Per ogni immagine vengono gestiti gli spazi di colore RGB, HSV e OPP. Viene impiegato un approccio unificato che estrae una serie di feature che rappresentano in maniera omogenea le informazioni sul colore e sulla tessitura per ogni regione estratta dall'algoritmo di segmentazione. In particolare si applica la trasformata Wavelet ai pixel dell'immagine e per ogni regione estratta vengono memorizzate le seguenti feature: area della regione (1D), centro della regione espresso attraverso i quattro coefficienti della trasformata dell'ultimo livello di trasformazione per tutti e tre i canali di colore (12D), i coefficienti delle matrici di covarianza calcolate per i punti della regione (24D) (totale 37D).
WebSeek	Feature sul colore: Per ogni immagine viene determinato l'istogramma dei colori a 166 elementi nello spazio HSV.
VisualSEEk	Feature sul colore: Color set binario derivato dall'istogramma dei colori a 166 elementi nello spazio HSV per ogni regione determinata attraverso l'algoritmo di segmentazione. Feature sulla tessitura: Texture set binario derivato dall'istogramma della tessitura a 512 elementi ottenuto iterando tre volte la trasformata Wavelet sulle energie dei singoli pixel, applicando un passo di generazione dei canali di tessitura, trasformandoli e quantizzandoli.

Tabella 2.2: Raffronto delle feature estratte nei motori commerciali analizzati.

Capitolo 3

Similitudine di immagini

Le metriche di similitudine permettono di determinare la somiglianza delle feature, estratte attraverso i metodi precedentemente illustrati, allo scopo di eseguire query basate sul contenuto visuale delle immagini.

Le metriche di similitudine confrontano le feature appartenenti a due immagini diverse e restituiscono un valore proporzionale alla loro somiglianza:

in particolare due immagini risulteranno molto simili se ottengono valori elevati di similitudine, mentre risulteranno poco simili se ottengono valori bassi.

3.1 Metriche di similitudine tra istogrammi.

Le metriche determinano la similitudine tra immagini attraverso funzioni che esprimono la distanza tra feature, in particolare, più la distanza tra due feature è alta più le relative immagini sono diverse e viceversa.

3.1.1 Funzione di corrispondenza $\mathcal{C}$.

Il legame tra similitudine e distanza è dato dalla funzione di corrispondenza $\mathcal{C}$, definita nel campo dei numeri reali positivi a valori compresi nell'intervallo $[0, 1]$, che soddisfa le seguenti proprietà:

$$\mathcal{C}(0) = 1; \quad x_1 \leq x_2 \Rightarrow \mathcal{C}(x_1) \geq \mathcal{C}(x_2) \quad \forall x_1, x_2 \in \Re_0^+ \tag{3.1}$$

La funzione di corrispondenza assegna alla similitudine un valore inversamente proporzionale alla distanza, pertanto risulterà massima nel caso di distanza nulla.

3.1.2 Normalizzazione degli istogrammi.

Le feature sul colore e sulla tessitura sono in genere definite attraverso gli istogrammi.

Gli istogrammi rappresentano la distribuzione dei colori o degli elementi di tessitura all'interno delle immagini.

Siccome gli istogrammi sono rappresentazioni di uno spazio discreto possono essere memorizzati attraverso vettori M-dimensionali dove M rappresenta il numero di colori attraverso i quali è codificata l'immagine.

Impiegando l'istogramma come feature, le metriche di distanza si riducono alla determinazione della distanza tra due vettori in uno spazio M-dimensionale.

Un passo preliminare nella determinazione della distanza tra due istogrammi consiste nella normalizzazione di questi ultimi, operazione che permette di paragonarli indipendentemente dalle dimensioni dell'immagine da cui sono estratti.

Introduciamo le seguenti classi di normalizzazione per $r = 1, 2$ dove con $\vec{H}$ indichiamo l'istogramma e con $\vec{H}'$ l'istogramma normalizzato:

$$\vec{H}' = \frac{\vec{H}}{\left(\sum_{m=0}^{M-1} |H[m]|^r\right)^{\frac{1}{r}}} \tag{3.2}$$

3.1.3 Classificazione delle principali metriche

Possiamo definire lo spazio metrico [**?**], dove gli istogrammi sono i punti di questo spazio, se per la funzione di distanza $D(\vec{H}_i, \vec{H}_j)$ tra due istogrammi H_i e H_j sono verificate le seguenti proprietà:

$$\begin{array}{ll} D(\vec{H}_i, \vec{H}_i) = 0 & \text{(Identità)} \\ D(\vec{H}_i, \vec{H}_j) \geq 0 & \text{(Non Negativa)} \\ D(\vec{H}_i, \vec{H}_j) = D(\vec{H}_j, \vec{H}_i) & \text{(Commutativa)} \\ D(\vec{H}_i, \vec{H}_k) \leq D(\vec{H}_i, \vec{H}_j) + D(\vec{H}_j, \vec{H}_k) & \text{(Diseguaglianza Triangolare)} \end{array} \tag{3.3}$$

La varie formulazioni per il calcolo della distanza che analizzeremo in seguito possono essere raggruppate in categorie in funzione di due caratteristiche peculiari:

la complessità computazionale e la capacità nel determinare la similitudine.

In particolare per le metriche impiegate nel confronto degli istogrammi si considerano due categorie:

- la prima categoria di distanze di tipo lineare prende il nome di "Minkowski–form Distance" a cui appartengono le metriche L_1 ed L_2;

- la seconda invece è di tipo quadratico e prende il nome di 'Quadratic–form Distance".

La prima categoria di funzioni di distanza è più semplice sul piano computazionale, ma ha l'inconveniente di generare molti falsi negativi, nel senso che fornisce distanze elevate anche per immagini percettivamente molto simili.

La forma più generale delle distanze appartenenti alla prima categoria è del tipo:

$$D^r(\vec{H_q}, \vec{H_t}) = \left(\sum_{m=0}^{M-1} |H_q[m] - H_t[m]|^r \right) \tag{3.4}$$

Per $r = 1$ otteniamo la metrica introdotta da Swain e Ballard in [**?**] che prende il nome di distanza L_1 oppure di intersezione di istogrammi e presuppone che gli istogrammi da confrontare siano normalizzati.

Questa distanza confronta le coppie di elementi nella stessa posizione all'interno dell'istogramma, ecco perché piccoli cambiamenti di illuminazione possono portare immagini molto simili a notevole distanza cioè si creano molti falsi negativi. Supponendo di ordinare i vari elementi dell'istogramma di Figura 3.1, in

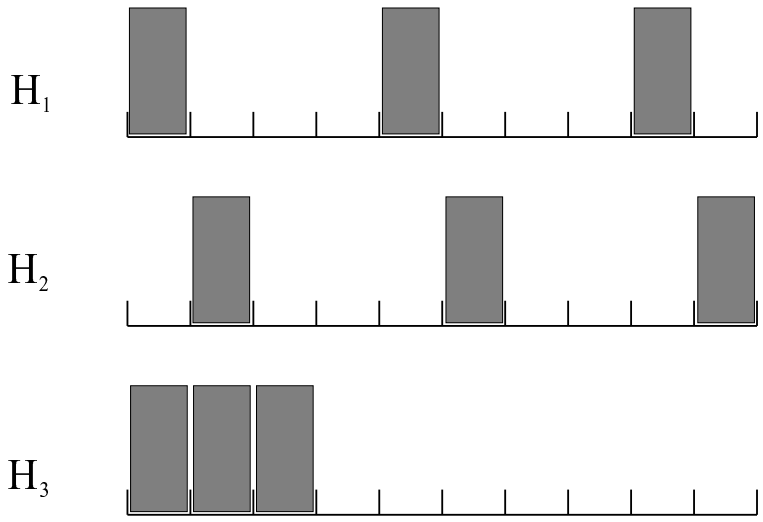


Figura 3.1: Rappresentazione di tre istogrammi H_1, H_2, H_3 a 10 elementi.

modo che elementi in posizioni adiacenti corrispondano a colori simili, allora percettivamente la distanza tra H_1 e H_2 dovrebbe essere minore di quella tra H_1 e H_3; invece impiegando L_1 ed essendo N il numero di pixel dell'immagine si ottiene la seguente catena di disequazioni:

$$L_1(H_1, H_2) = 2N > L_1(H_1, H_3) = L_1(H_2, H_3) = 1,33N \tag{3.5}$$

Per $r = 2$ invece si ottiene la classica distanza Euclidea od L_2:

$$L_2(\vec{H}_q, \vec{H}_t) = \sqrt{\sum_{m=0}^{M-1} (H_q[m] - H_t[m])^2} \tag{3.6}$$

Per migliorare le prestazioni della distanza definita da Swain e Ballard, che confronta gli elementi in posizioni omologhe, la classe di distanze quadratiche, che è stata introdotta da Niblack in [**?**], compara coppie di istogrammi confrontando ogni elemento del primo con tutti gli altri del secondo associando a ciascun confronto un peso a_{ij} definito all'interno della matrice di similitudine A.

La distanza quadratica si esprime tramite la formula:

$$D_{Quad} = (\vec{H}_q - \vec{H}_t)^T A (\vec{H}_q - \vec{H}_t) \tag{3.7}$$

Dove A è la matrice di similitudine che associa pesi diversi in base alla similitudine tra colori e che, nel caso dello spazio dei colori HSV, viene determinata nel modo seguente:

$$a_{ij} = 1 - \frac{1}{\sqrt{5}} \left[(v_i - v_j)^2 + (s_i \cos h_i - s_j \cos h_j)^2 + (s_i \sin h_i - s_j \sin h_j)^2 \right]^{\frac{1}{2}} \tag{3.8}$$

Attraverso questa formulazione si ottiene $a_{ii} = 1$ e $a_{ij} = 0$ per colori che nello spazio HSV sono separati dalla massima distanza possibile.

Siccome sono coinvolte operazioni tra matrici e vettori la determinazione della distanza quadratica risulta computazionalmente più complessa delle precedenti.

La distanza quadratica migliora la determinazione della distanza tra feature, però, se l'istogramma non presenta dei picchi ben definiti, la similitudine viene sovrastimata producendo molti falsi positivi.

Ad esempio nel caso di H_4, H_5 e H_6 di Figura 3.2 l'ordinamento delle relative distanze quadratiche non corrisponde al naturale ordinamento delle similitudini. Impiegando la distanza quadratica precedentemente definita si ottiene sempre rispetto agli istogrammi della Figura 3.2 la seguente catena di disequazioni:

$$\begin{gathered} D_{Quad}(H_4, H_5) = 0,68N > \\ > D_{Quad}(H_4, H_6) = 0,63N > \\ > D_{Quad}(H_5, H_6) = 0,55N \end{gathered} \tag{3.9}$$

La distanza quadratica è esattamente quella impiegata da QBIC [**?**] per il confronto delle feature relative al colore.

La generazione di falsi positivi è una situazione migliore, dal punto di vista dell'utente, rispetto a quella dei falsi negativi, infatti mentre i primi possono essere eliminati in fase di presentazione dei risultati attraverso l'ispezione manuale, i secondi precludono all'ispezione immagini potenzialmente interessanti.

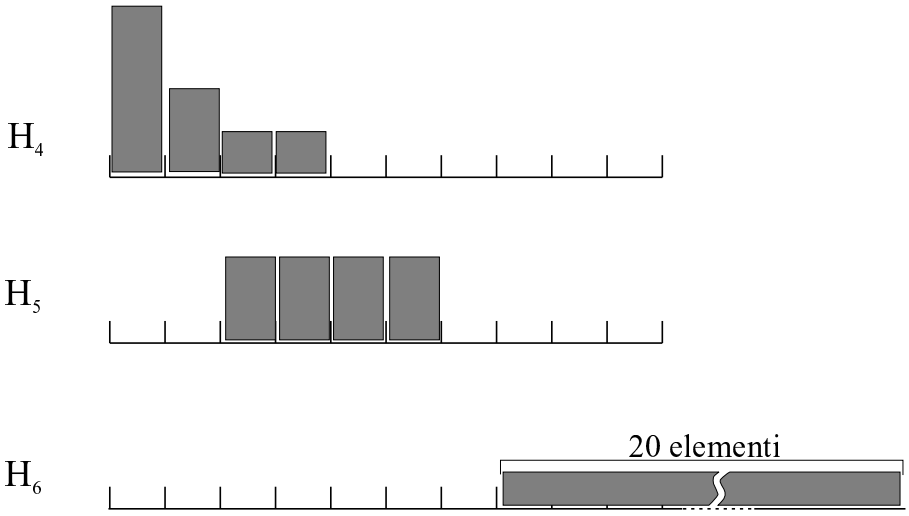


Figura 3.2: Rappresentazione di tre istogrammi H_4, H_5, H_6 a 27 elementi.

3.2 Metriche di similitudine tra Color Set.

Le funzioni di distanza, definite a partire dagli istogrammi dei colori, possono essere ulteriormente semplificate quando vengono applicate ai Color Set, che sono rappresentati da un vettore binario derivato dall'istogramma tramite l'applicazione di una soglia τ_k.

Nel caso delle metriche di tipo "Minkowski-form distance" possiamo introdurre la distanza di Hamming definita tra due binary Set s_q ed s_t come:

$$D_{Ham}(\vec{s_q}, \vec{s_t}) = \frac{|\vec{s_q} - \vec{s_t}|}{|\vec{s_q}| \cdot |\vec{s_t}|} \tag{3.10}$$

Dove $|s_c| = \sum_m s_c[m]$ con $c \in \{q, t\}$ rappresenta il numero di elementi con valore logico 1.

D_{Ham} equivale ad eseguire l'exlusive OR bit a bit tra i due Color Set.

Nel caso della distanza quadratica, l'introduzione dei Color set permette ulteriori semplificazioni, infatti definendo:

$$\mu_q = \vec{s_q^T} A \vec{s_q} \quad \mu_t = \vec{s_t^T} A \vec{s_t} \quad r_t = A\vec{s_t} \tag{3.11}$$

e sostituendo nella formula della distanza quadratica si ottiene:

$$D_{Quad}(\vec{s_q}, \vec{s_t}) = \mu_q + \mu_t - 2s_q^T r_t \tag{3.12}$$

Questa è esattamente la distanza impiegata per calcolare la similitudine delle regioni estratte da VisualSEEk attraverso l'elaborazione dei Color Set [**?**].

3.3 Metriche di similitudine tra singoli pixel.

Questa categoria di funzioni di distanza, definita in [**?**], rappresenta un approccio completamente diverso che consiste nell'ignorare completamente gli istogrammi e le distanze che operano a partire da essi, definendo la distanza direttamente sui pixel ed estraendo feature più semplici sulla distribuzione statistica dei colori.

Dalla teoria della probabilità sappiamo che una funzione di distribuzione è caratterizzata completamente attraverso i suoi momenti, ad esempio quelli centrali, così se interpretiamo la distribuzione dei colori di un'immagine come una distribuzione probabilistica allora feature rappresentative possono essere i momenti centrali.

Supponiamo di memorizzare per ogni canale di colore i primi tre momenti centrali:

il primo rappresenta il valore medio, il secondo rappresenta la varianza ed il terzo l'asimmetria (skewness) della distribuzione di ciascun colore.

Il tre momenti illustrati sono definiti dalle seguenti formule dove K è l'indice del canale di colore, i e j sono gli indici di riga e colonna, R e C il numero di righe e colonne per una data immagine I:

$$E_K^I = \frac{1}{RC}\sum_{i=1}^{R}\sum_{j=1}^{C} p_{ij}^{I,K} \tag{3.13}$$

$$\sigma_K^I = \left(\frac{1}{RC}\sum_{i=1}^{R}\sum_{j=1}^{C}(p_{ij}^{I,K} - E_K^I)^2\right)^{\frac{1}{2}} \tag{3.14}$$

$$s_K^I = \left(\frac{1}{RC}\sum_{i=1}^{R}\sum_{j=1}^{C}(p_{ij}^{I,K} - E_K^I)^3\right)^{\frac{1}{3}} \tag{3.15}$$

Supponiamo che I_1 ed I_2 siano le due distribuzioni di colore delle immagini su cui si deve calcolare la similitudine, che dispongano entrambe di Cn canali di colore e dove E_K^I, σ_K^I, s_K^I sono i momenti con $I \in \{I_1, I_2\}$ e $Cn = 3$ numero di canali delle immagini, allora la distanza è definita come:

$$D_{Mom}(I_1, I_2) = \sum_{K=1}^{Cn}(w_{i1}|E_K^{I_1} - E_K^{I_2}| + w_{i2}|\sigma_K^{I_1} - \sigma_K^{I_2}| + w_{i3}|s_K^{I_1} - s_K^{I_2}|) \tag{3.16}$$

Dove i coefficienti $w_i > 0$ rappresentano opportuni pesi a cui l'utente può attribuite valori diversi in funzione dell'importanza che vuole attribuire ai vari momenti.

Siccome questa distanza impiega un numero ristretto di momenti, essa non definisce uno spazio metrico.

Come illustrano Stricker e Orengo in [**?**] questo approccio è migliore di quelli precedentemente illustrati per il confronto degli istogrammi.

Riassunto delle metriche di distanza impiegate	
Motore	Metrica di distanza impiegata
QBIC	Colore: Distanza Quadratica; Tessitura: Distanza Euclidea pesata; Forma: Distanza Euclidea pesata.
WindSurf	Confronto tra coefficienti della trasformata Wavelet estratte dalle regioni: Distanza di Bhattacharyya
WebSeek	Colore: Distanza Quadratica.
VisualSEEk	Color Set e Texture Set: Distanza quadratica semplificata su vettori binari.

Tabella 3.1: Raffronto delle metriche di distanza implementate nei motori commerciali analizzati.

Capitolo 4

Algoritmi per l'estrazione di regioni

Il procedimento per l'estrazione di regioni permette di ottenere informazioni sulla localizzazione delle feature all'interno dell'immagine.

Quando le feature vengono confrontate a livello di intera immagine tutte le informazioni sulla loro distribuzione vengono perse degradando notevolmente le prestazioni nell'esecuzione delle query:

ad esempio possono esistere immagini con lo stesso istogramma, ma con una distribuzione dei pixel totalmente diversa.

Rappresentando le feature a livello di regione, che in genere vengono identificate attraverso il baricentro e l'estensione, si ottiene maggiore precisione nel recupero delle immagini.

4.1 Classificazione degli algoritmi di segmentazione.

Il processo tramite il quale i pixel vengono raggruppati a formare le regioni prende il nome di segmentazione.

Detta $P(x, y)$ una proprietà dell'immagine definita sul pixel, ad esempio il colore, col termine segmentazione si intende il partizionamento dell'immagine I in un insieme di regioni $\{R_1, R_2, \ldots, R_S\}$ omogenee rispetto a P che prende il nome di predicato di omogeneità. Il partizionamento di un'immagine I deve soddisfare i seguenti vincoli:

$$\begin{array}{l} \bigcup_{i=1}^{S} R_i = I \\ R_i \cap R_j = \emptyset \text{ per } i \neq j \end{array} \tag{4.1}$$

Questi richiedono che l'unione delle regioni sia l'intera immagine e che ogni coppia di regioni risulti disgiunta.

Come vedremo in seguito, per ottenere un confronto più robusto il primo vincolo può essere rilassato.

La segmentazione è un passo fondamentale per una migliore rappresentazione del contenuto informativo delle immagini; ad esempio le regioni possono essere ulteriormente analizzate per l'estrazione di nuove feature come la forma o la tessitura.

La segmentazione può essere interpretata come un processo di classificazione dove ciascun pixel viene assegnato ad una classe (regione) in funzione del valore di P.

Gli algoritmi di segmentazione devono soddisfare i seguenti criteri per l'estrazione delle regioni:

- Le regioni devono essere il più possibile omogenee rispetto al predicato P;
- Le regioni non devono avere troppe lacune;
- Regioni adiacenti devono essere caratterizzate da valori significativamente diversi di P;
- I confini fra le regioni devono essere regolari e la loro localizzazione spaziale deve essere accurata.

Gli algoritmi di segmentazione per soddisfare i criteri precedenti, oltre a tenere conto del valore del pixel, devono considerare le relazioni di adiacenza tra gruppi di pixel, in particolare possiamo analizzare due tipologie di algoritmi:

- Algoritmi Split and Merge (Dividi e Fondi);
- Algoritmi di Region Growing (Crescita di regioni).

Negli algoritmi di Split and Merge si parte dall'immagine intera e si determinano le regioni attraverso ripetute suddivisioni dell'immagine in blocchi, questi vengono ulteriormente divisi o fusi con quelli vicini fino a quando non sono soddisfatti tutti i criteri di omogeneità.

Negli algoritmi di region growing invece si parte da una serie di punti (semi) interni all'immagine e da questi semi "crescono" le regioni assegnando i rimanenti pixel al rispettivo seme in funzione del criterio di omogeneità fino a quando tutti i pixel sono stati assegnati.

Entriamo ora ad analizzare nei particolari vari algoritmi appartenenti alle due categorie sopracitate.

4.2 Algoritmi di Split and Merge.

L'algoritmo di split and merge richiede che sia definito un predicato di omogeneità sotto forma di funzione $P(R_i)$, questa riceve in ingresso l'insieme dei pixel della regione e restituisce un valore intero corrispondente all'omogeneità della regione.

L'algoritmo di split and merge, come indica il nome, può essere suddiviso in due fasi:

1. **Split:**questa fase si applica a partire dall'intera immagine, se è omogenea l'algoritmo può terminare poiché l'immagine è già segmentata, cioè composta da una sola regione.

   In caso contrario dividiamo l'immagine in due parti e ripetiamo questa suddivisione in parti ricorsivamente fino ad ottenere un set di regioni $\{R_1, R_2, \ldots, R_n\}$ che verificano il predicato di omogeneità;

2. **Merge:** in questa fase si esegue il procedimento opposto: si analizzano tutte le possibili coppie di regioni adiacenti; se il predicato risulta verificato per tutti i punti che appartengono ad entrambe le regioni allora si opera la fusione delle due in un unica regione.

   Al termine di questa fase si ottiene un set di regioni $\{R'_1, R'_2, \ldots, R'_k\}$ con $k \leq n$;

Si ripetono i due passi precedenti fino a che non ci sono più regioni da dividere o da fondere.

Al termine dell'algoritmo si ottiene un set di regioni per cui valgono le seguenti proprietà:

$$\begin{array}{lll} \forall i: & 1 < i < k & P(R_i) < \theta \\ \forall i,j: & 1 < i,j < k \wedge i \neq j & P(R_i \cup r_j) > \theta \end{array} \tag{4.2}$$

Dove θ è fissato all'interno del sistema e determina la soglia di omogeneità.

Ad esempio in [**?**] viene illustrato un sistema CBVQ che impiega per le estrazione delle regioni delle immagini memorizzate l'algoritmo di Split and Merge con l'aggiunta di una soglia sul numero minimo di pixel per regione.

Se una regione contiene un numero di pixel inferiore alla soglia viene semplicemente ignorata ed esclusa dalla partizione, in questo caso l'unione di tutte le regioni non equivale all'intera immagine.

4.3 Algoritmi di "Region Growing".

Come abbiamo già detto precedentemente gli algoritmi della categoria "Region Growing" partono selezionando un insieme di punti iniziali e ripetono in maniera iterativa l'aggregazione dei rimanenti punti a quelli inizialmente selezionati.

4.3.1 Algoritmo K–Means.

Un algoritmo che appartiene alla categoria "Region Growing" è il K-Means che definisce un insieme V di K centri di regione (cluster) a cui vengono associati gli NP punti da partizionare appartenenti all'insieme X:

$$\begin{array}{rcl} X & = & \{\vec{x_1}, \vec{x_2}, \ldots, \vec{x_{NP}}\} \\ V & = & \{\vec{\mu_1}, \vec{\mu_2}, \ldots, \vec{\mu_K}\} \end{array} \tag{4.3}$$

Il K–Means si prefigge l'obbiettivo di minimizzare la seguente funzione che sostituisce il predicato di omogeneità:

$$J = \sum_{i=1}^{k} \sum_{\vec{x_j} = CL_i} D(\vec{x_j}, \vec{\mu_i})^2 \tag{4.4}$$

dove $D(x_j, \mu_i)$ è la distanza tra il punto x_j ed il rispettivo centro della regione CL che nel caso più generale è del tipo:

$$D(\vec{x_j}, \vec{\mu_{i*}})^2 = (\vec{x_j} - \vec{\mu_i})^T (\vec{x_j} - \vec{\mu_i}) \tag{4.5}$$

con A di dimensione P x P con P dimensione del vettore $\vec{x_j}$.

Allora l'algoritmo K-means può essere suddiviso nei seguenti passi:

1. Scelta iniziale dei K centri dei cluster;

2. Assegnazione dei punti $\vec{x_j}$ al cluster CL il cui centro $\vec{\mu_{i*}}$ è più vicino a $\vec{x_j}$:

$$\begin{array}{c} \vec{x_j} \in CL_{i*} \Leftrightarrow \forall \vec{\mu_i} \in V \\ \text{si ha che} \\ D(\vec{x_j}, \vec{\mu_{i*}}) = \min_{i \in [1,K]} \{D(\vec{x_j}, \vec{\mu_i})\} \end{array} \tag{4.6}$$

3. Calcolo dei nuovi centri dei cluster:

$$\mu_i^N = \frac{\sum_{x_j \in CL_i} x_j}{card(CL_i)} \tag{4.7}$$

4. Si ripetono i punti 2 e 3 fino a quando l'algoritmo non converge cioè i cluster non cambiano più.

L'assegnazione iniziale dei k centri è di fondamentale importanza in quanto una scelta scorretta può compromettere il processo di raggruppamento dei pixel.

La situazione che si vuole evitare è che un'immagine, che contiene al suo interno almeno due regioni distinte, non riesca ad essere suddivisa dall'algoritmo K–Means a causa di un errata scelta dei K centri.

Per evitare questi problemi si sceglie un numero iniziale di "possibili centri" pari a tre volte il numero dei centri necessari. I "possibili centri" sono scelti in maniera casuale all'interno dell'insieme dei punti dell'immagine.

Il primo centro viene scelto in maniera casuale all'interno di quelli possibili, il secondo viene scelto tra i rimanenti come quello a maggiore distanza dal primo, il terzo con la massima distanza dai primi due e così via fino a quando tutti i K centri sono assegnati.

Dopo aver scelto i centri iniziali questi vengono modificati al fine di determinare la migliore partizione fra i punti.

Il valore K, numero di cluster da estrarre, non viene fissato in maniera statica, ma in funzione della singola immagine viene determinato il K ottimo all'interno dell'intervallo $[K_{MIN}, K_{MAX}]$, valutando i risultati ottenuti con tutti i possibili valori di K attraverso una funzione di validità che ci offre un giudizio sulla bontà della segmentazione.

La determinazione della funzione di validità necessita della definizione delle ulteriori funzioni di compattezza e separabilità:

$$d_{ij} = d(\vec{x_j}, \vec{\mu_i}) \tag{4.8}$$

è detta deviazione del punto x_j dal cluster a cui appartiene;

$$\sigma_i = \sum_{x_j \in CL_i} (d_{ij})^2 \tag{4.9}$$

è la variazione ovvero la sommatoria dei quadrati delle deviazioni di ogni punto per un certo Cluster;

$$\sigma = \sum_i \sigma_i = \sum_i \sum_{x_j \in CL_i} (d_{ij})^2 \tag{4.10}$$

è la variazione totale, somma delle variazioni calcolate per ogni cluster.

La compattezza della partizione è definita attraverso il rapporto tra variazione totale e numero dei punti NP:

$$\pi = \frac{\sigma}{NP} \tag{4.11}$$

La separazione invece equivale al quadrato della distanza minima:

$$sep = D^2_{Min} = \left[\min_{i \neq i'} \{ D(\vec{\mu_i} - \vec{\mu_{i'}}) \} \right]^2 \tag{4.12}$$

La funzione di validità si determina dal rapporto compattezza e separazione, minimizzare la validità è equivalente a massimizzare la funzione obbiettivo dell'eq. 4.4 che permette di ottenere una buona clusterizzazione.

$$val = \frac{\pi}{sep} \tag{4.13}$$

L'algoritmo di clustering K–Means viene impiegato per l'estrazione delle regioni all'interno del sistema WindSurf [**?**] con l'aggiunta di una soglia che permette di eliminare le regioni con un numero di punti inferiore al 2% della globalità dei punti.

4.3.2 Retro proiezione dei Color Set.

Un altro sistema, che permette di estrarre le regioni dalle immagini, impiega la Retro Proiezione dei Color Set sui pixel dell'immagine.

Il processo di estrazione delle regioni si ottiene attraverso i seguenti passi:

1. Trasformazione, quantizzazione e filtraggio dei pixel dell'immagine;
2. Retro proiezione dei color set binari sui pixel dell'immagine;
3. Assegnamento delle etichette alle regioni ed applicazione delle soglie.

In particolare il processo di retro proiezione restituisce un'immagine binaria, in cui i pixel assumono solo due possibili valori, in funzione del confronto tra le informazioni contenute nel color set e quelle dell'immagine da retro proiettare.

Come sappiamo il color set $s_c[i]$ è un vettore binario che contiene tanti elementi, quanti sono i colori dello spazio trasformato e quantizzato e che memorizza le preferenze per un sottoinsieme dei colori dell'immagine.

La retroproiezione valuta il colore k di ciascun pixel dell'immagine di ingresso $I[x, y]$ ad assegna al pixel nella stessa posizione nell'immagine di uscita $O[x, y]$ il valore dell'elemento del color set che corrisponde allo stesso colore dell'immagine d'ingresso.

$$k = I[i, j] \Rightarrow O[i, j] = s_c[i] \tag{4.14}$$

Successivamente, l'immagine binaria viene filtrata in modo da eliminare i raggruppamenti isolati di pochi pixel o per unire due regioni molto vicine che incidentalmente risultino separate.

Dopo l'operazione di filtraggio si esegue l'etichettamento dell'immagine che assegna a ciascuna regione estratta dalla retro proiezione lo stesso valore dell'etichetta.

Ogni regione per essere considerata valida deve essere compatibile con più di vincoli rappresentati da una serie di soglie; queste esprimono il valore minimo che certe proprietà possono assumere.

Se solo uno dei seguenti vincoli non è soddisfatto per una data regione, questa viene scartata ponendo a 0 la rispettiva etichetta.

Le tre soglie rappresentano:

- τ_a: corrisponde alla dimensione della regione. In particolare per ogni regione si ottiene:

$$\sum_n L[n] \geq \tau_a \tag{4.15}$$

  dove L[n] è l'istogramma locale della regione;

- τ_b: rappresenta il contributo dell'm-esimo colore del color set alla regione. In particolare per ogni regione si ottiene:

$$s_c[m] = 1 \Rightarrow L[m] \geq \tau_b \tag{4.16}$$

- τ_c:rappresenta il contributo relativo dell'm-esimo colore del color set alla regione rispetto tutti gli altri colori del color set:

$$s_c[m] = 1 \Rightarrow L[m] \geq \tau_c \sum_n L[n] \tag{4.17}$$

L'utilizzo combinato di queste tre soglie evita di sprecare spazio per la memorizzazione di regioni non significative ai fini della segmentazione.

poiché la scelta del color set ottimale per l'estrazione delle regioni necessiterebbe la retro proiezione di 2^m color set, dove m è il numero di colori in cui è quantizzato lo spazio, allora vengono inseriti altri vincoli che permettono di scartare a priori color set non adeguati.

In particolare si introducono due ulteriori soglie T_0 e T_1:

- T_0 rappresenta il fatto che se un elemento del color set che viene retro proiettato è ad 1 ci devono essere almeno T_0 pixel con quel colore nell'immagine

$$s_c[m] = 1 \Rightarrow H[m] \geq T_0 \tag{4.18}$$

  con H[m] è l'istogramma globale dei colori dell'immagine I.

- T_1 invece tiene conto del fatto che il residuo dell'istogramma globale deve contenere per ogni colore incluso nel color set almeno T_1 pixel.

  Per residuo dell'istogramma globale si intende quell'istogramma a cui vengono sottratti i pixel corrispondenti alle regioni già estratte nelle precedenti retro proiezioni.

$$s_c[m] = 1 \wedge H[m] \geq \tau_0 \Rightarrow H[m] - \sum_n L_n[m] \geq \tau_1 \tag{4.19}$$

  dove $L_n[m]$ è l'istogramma locale della n-esima regione.

La definizione del residuo implica che i vari color set siano esplorati in maniera incrementale, cioè prima si verificano tutti i color set unitari che contengono al loro interno un solo elemento ad 1, poi quelli con due elementi ad 1 e così via.

L'algoritmo di iterazione dei color set si compone dei seguenti passi:

1. Esplorazione dei color set unitari:
   (a) Dato l'istogramma dell'immagine $H[m]$, seleziona gli $m' = m$ dove $H[m] \geq \tau_0$
   (b) per ogni m' costruisci il color set c con $s_c[k] = 1$ per $k = m'$ e con $s_c[k] = 0$ negli altri casi . Eseguiamo la retroproiezione di c[k] su I[x,y] ed otteniamo O[x,y]. Per ogni regione n calcoliamo l'istogramma locale $L_n[m]$.
   (c) Calcoliamo il residuo dell'istogramma globale.
2. Color set con due elementi:
   (a) trova tutti $l' = l$ e $m' = m$ dove$H[l] \geq \tau_0, H[m] \geq \tau 0 \wedge H_R[l] \geq \tau_1, H_R[m] \geq \tau_1$
   (b) per ogni coppia m' e l' costruisci il color set S_c con $s_c[k] = 1$ per $k = m'$ e $k = l'$ e con $s_c[k] = 0$ negli altri casi.
   Eseguiamo la retro proiezione di $s_c[k]$ su $I[x, y]$ ed otteniamo $O[x, y]$. Per ogni regione n calcoliamo l'istogramma locale $L_n[m]$.
   (c) Aggiorniamo il residuo dell'istogramma globale
3. Color set a tre o più elementi: ripetiamo i passi precedenti. . .

Questo algoritmo di segmentazione è stato implementato nel sistema VisualSEEk sviluppato in [**?**].

Riassunto delle capacità di segmentazione	
Motore	Modalità di estrazione delle regioni
QBIC	Semi automatica: attraverso l'intervento dell'utente assistito da strumenti di aiuto come "Flood Fill" e "Snakes". Automatica: attraverso la scomposizione dell'immagine in un set statico di regioni rettangolari predefinite.
WindSurf	Automatica: attraverso l'applicazione dell'algoritmo K-Means che raggruppa i pixel in funzione dei valori della trasformata Wavelet nella sottobanda LL dell'ultimo livello di trasformazione utilizzando la distanza Eucliudea o di Mahalanobis.
WebSeek	nessuna politica per l'estrazione delle regioni.
VisualSEEk	Automatica: attraverso l'applicazione della retro proiezione dei color Set e dei Texture Set ed eliminazione delle regioni non rilevanti attraverso il confronto con una serie di soglie.

Tabella 4.1: Raffronto delle capacità di calcolo delle regioni e dei metodi impiegati per l'estrazione nei motori commerciali analizzati.

Capitolo 5

Indici Multidimensionali per MMDBMS

Attraverso l'introduzione degli indici si cercano di migliorare le prestazioni nel recupero dei risultati delle interrogazioni, quando le dimensioni della base di dati crescono in misura tale da non rendere più fattibile, in termini di tempo, la scansione sequenziale di tutti gli elementi memorizzati nel database.

Attraverso l'implementazione degli indici si cerca di organizzare i dati in maniera gerarchica, attraverso strutture ad albero, in modo da assegnare alla posizione all'interno dello stesso informazioni sulle proprietà su cui si vogliono ottimizzare le ricerche.

In particolare, quando le proprietà appartengono ad uno spazio multidimensionale si parla di indici multidimensionali.

5.1 k-d Tree.

La struttura ad albero detta k-d Tree viene impiegata per memorizzare ed organizzare insiemi di punti in uno spazio k-dimensionale.

Il k-d Tree appartiene alla categoria degli alberi binari, cioè con due sole diramazioni per ogni nodo non bilanciate. Lo spazio da organizzare viene diviso tramite una serie di iperpiani di dimensione $(k-1)$.

Gli iperpiani sono iso-orientati e le loro direzioni si alternano tra le k direzioni disponibili.

Ogni iperpiano deve contenere al massimo un solo punto che viene inserito nell'albero all'interno di un nodo.

Ogni nodo ha al più due figli per cui viene definita una regola per discriminare durante le varie operazioni tra queste due alternative.

I punti che andranno a popolare l'albero nel caso bidimensionale, $k = 2$,

identificano il centro di regioni dell'immagine, con proprietà di un certo interesse, e verranno identificati dalle rispettive coordinate cartesiane.

La struttura dei nodi di Figura 5.1 dell'albero è la seguente:

```
Kdnode{
   Info:   information;
   XVal:   real;
   YVal:   real;
   LLink:  *Kdnode;
   RLink:  *Kdnode;
}
```

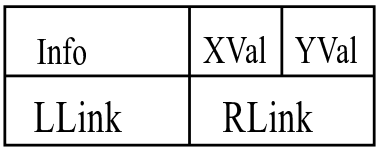


Figura 5.1: Struttura di un nodo di 2d Tree.

Il campo Info contiene tutte le proprietà del punto che l'utente intende memorizzare. I campi XVal e YVal invece rappresentano rispetto ad una data origine, unica per tutti i punti memorizzati, le coordinate cartesiane del punto memorizzato. I campi LLink e RLink sono puntatori a strutture di tipo Kdnode radici rispettivamente dei sotto alberi di sinistra e di destra.

Supponendo che T sia il puntatore alla radice dell'albero e che N sia un nodo all'interno dell'albero definiamo il livello del nodo N detto $level(N)$ come:

$$Level(N) = \begin{cases} 0 & \text{se N è la radice T dell'albero} \\ level(P) + 1 & \text{se N è un figlio di P} \end{cases} \tag{5.1}$$

Un 2-d tree, che memorizza punti bidimensionali, per essere considerato tale deve soddisfare le seguenti condizioni:

1. Se N è un nodo nell'albero di livello pari, allora ogni nodo M nel sottoalbero di sinistra puntato da N.LLink soddisfa la proprietà che $M.XVal < N.XVal$ ed ogni nodo P nel sottoalbero di destra puntato da N.RLink soddisfa la proprietà che $P.XVal \geq N.XVal$.

2. Se N è un nodo nell'albero di livello dispari, allora ogni nodo M nel sottoalbero di sinistra puntato da N.LLink soddisfa la proprietà che $M.YVal < N.YVal$ ed ogni nodo P nel sottoalbero di destra puntato da N.RLink soddisfa la proprietà che $P.YVal \geq N.YVal$.

5.1.1 Inserimento e ricerca di nodi in un 2-d tree.

La forma dell'albero dipende oltre che dai valori inseriti anche dall'ordine in cui vengono inseriti.

Inserire un nuovo elemento N all'interno di un albero con radice T avviene seguendo le due regole precedentemente definite.

Supponiamo che P sia il puntatore al nodo dell'albero esaminato nell'interazione corrente. L'algoritmo di inserimento esplora in maniera ricorsiva i nodi dell'albero già inseriti partendo dalla radice e confrontando il punto da inserire con quello memorizzato nel nodo.

Al primo passo P assume il valore della radice dell'albero, se N e P hanno le stesse coordinate il nodo esiste già e l'inserimento è completato, in caso contrario spostiamo il puntatore P, eseguendo un confronto sui valori delle ascisse di N e P, al figlio di sinistra se $N.XVal < P.XVal$ od al nodo di destra se $N.XVal \geq T.XVal$.

Se il punto memorizzato nel nodo puntato da P e quello da inserire coincidono abbiamo terminato perché il nodo esiste già, in caso contrario questa volta scegliamo il sotto albero da esaminare in base al confronto dei valori delle ordinate in particolare spostiamo P al figlio di sinistra se $N.YVal < P.YVal$ od al nodo di destra se $N.YVal \geq P.YVal$.

Generalizzando il procedimento, il puntatore al nuovo nodo da esaminare viene assegnato al figlio di sinistra o di destra in base al confronto con i valori delle ascisse o delle ordinate in funzione del livello raggiunto all'interno dell'albero.

Se il livello è pari si confrontano le ascisse, viceversa se è dispari, si confrontano le ordinate.

Il nuovo nodo viene inserito nell'albero quando P punta a NULL, ciò significa che si è raggiunta la fine di un ramo dell'albero, allora creiamo un nuovo nodo inserendo i dati da memorizzare ed assegniamo al puntatore che prima era NULL il valore delle nuova locazione inserita e a questo punto l'algoritmo di inserimento può terminare.

Regione	CentroId(XVal,YVal)
RegId 1	(24,36)
RegId 2	(50,30)
RegId 3	(47,48)
RegId 4	(20,68)
RegId 5	(6,40)

Tabella 5.1: coordinate del baricentro delle regioni da inserire nel 2-d Tree.

Nella Tabella 5.1.1 e nella Figura 5.2 sono riportate una serie di regioni, estrat-

te da precedenti elaborazioni su un'ipotetica immagine, con evidenziati i rispettivi centri, a cui è stata sovrapposta una griglia, con origine nell'angolo in basso a sinistra, con celle di forma quadrata di un centimetro di lato. Supponiamo di

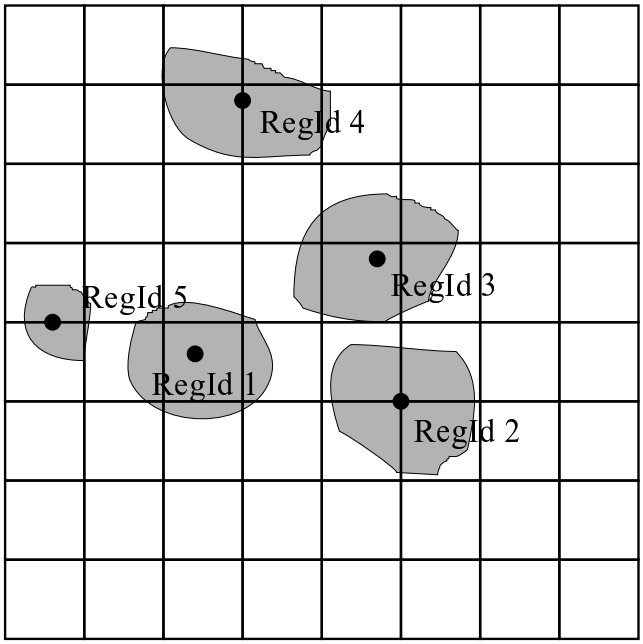


Figura 5.2: Disposizione delle ipotetiche regioni da inserire nell'indice.

selezionare una serie di regioni e di inserirle in successione nell'ordine prestabilito all'interno dell'albero 2-d Tree assegnando al campo Info l'identificativo della regione ed ai campi XVal e YVal le coordinate del centro.

1. L'inserimento di RegId1 crea il primo ed unico nodo dell'albero con $Info = RegId1$, $XVal = 24$ e $Yval = 36$.

   L'inserimento del nodo che memorizza RegId1 definisce in maniera implicita una regione contente l'intera mappa. In generale ogni nodo N inserito definisce una nuova regione Reg(N) che contiene al suo interno il punto di coordinate $(N.XVal, N.YVal)$.

   L'inserimento di un nuovo punto divide in due, orizzontalmente se il livello del nodo è dispari verticalmente se il nodo è pari, la regione relativa al nodo inserito.

2. Quando inseriamo il nodo con $N.Info = RegId2$ con $N.XVal = 50$ e $N.YVal = 30$ poiché ci troviamo su un livello dispari dobbiamo con-

frontare i valori delle ascisse dell'elemento contenuto nella radice con quello da inserire:

scegliamo il puntatore al sottoalbero di sinistra se $N.XVal < T.XVal$ oppure il puntatore al sottoalbero di destra nel caso che $N.XVal \geq T.XVal$.

Nel nostro esempio scegliamo il sottoalbero di destra poiché Regid2 risiede nella metà destra della regione individuata da RegId1, infatti $N.XVal = 50$ è maggiore di $T.XVal = 24$.

Mentre nel precedente inserimento la regione era stata divisa verticalmente, l'inserimento di RegId2 divide orizzontalmente la regione posta a destra della linea verticale.

3. Il terzo nodo da inserire contiene i dati relativi a RegId3 con posizione $N.XVal = 47$ e $N.YVal = 48$ al primo passo confrontiamo i valori delle ascisse di RegId1 e RegId3, poiché $N.XVal \geq T.XVal$ scegliamo il sottoalbero di destra.

   Al secondo passo confrontiamo il valore di ordinata del figlio selezionato $P.Info = RegId2$ con quello del nuovo elemento da inserire N e otteniamo che $N.YVal \geq P.YVal$ per cui inseriamo il nuovo nodo nel puntatore al sottalbero di destra del nodo P che è libero.

   La nuova regione delimitata dall'ascissa di RegId1 e dalla ordinata di RegId2 viene divisa verticalmente lungo l'ascissa di RegId3.

4. Il nodo successivo ha coordinate $N.XVal = 20$ e $N.YVal = 68$ e il campo Info=RegId4.

   Al primo passo confrontiamo i valori delle ascisse di $T.Info = RegId1$ e $N.Info = RegId4$ e poiché $N.XVal \geq T.XVal$ scegliamo il sottoalbero di destra.

   Al secondo passo confrontiamo il valore di ordinata di $P.Info = RegId2$ e verifichiamo che $N.YVal \geq P.YVal$ e scegliamo nuovamente il sottoalbero di destra che contiene il nodo $Q.Info = RegiId3$.

   A questo punto confrontiamo nuovamente le ascisse del punto da inserire con quella dell'ultimo punto raggiunto poiché $N.XVal < Q.XVal$ inseriamo nel puntatore di sinistra che è vuoto il nuovo elemento.

   La nuova regione delimitata dalle ascisse di RegId1 e RegId3 e dall'ordinata di RegId2 e viene divisa orizzontalmente lungo l'ordinata di RegId4

5. Infine quando inseriamo l'ultimo nodo con coordinate $N.XVal = 6$ e $N.YVal = 40$ e $N.Info = RegId5$.

Verifichiamo immediatamente che $N.XVal < T.XVal$, che il puntatore di sinistra è libero per cui il nuovo nodo viene inserito come figlio della radice e creando una nuova regione delimitata dall'ascissa di RegId1 che viene divisa orizzontalmente dall'ordinata di RegId5.

Il risultato di questa serie di inserimenti e visibile nelle Figure 5.3 e 5.4 Come

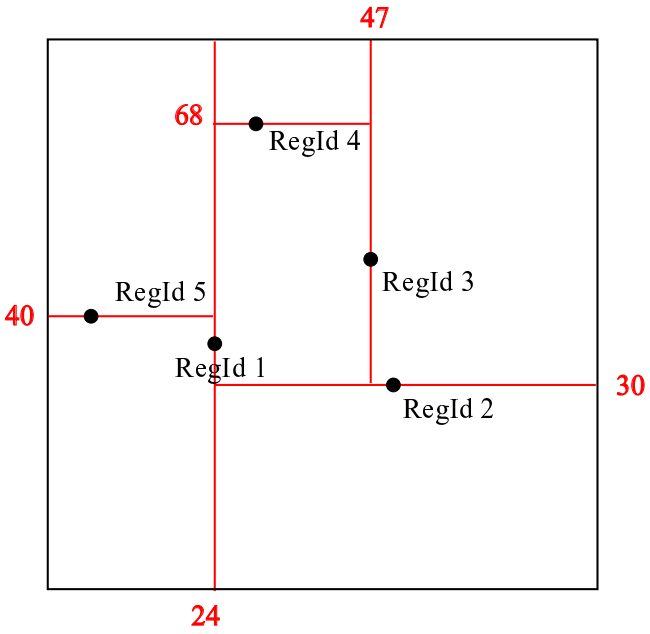


Figura 5.3: Schema di suddivisione del piano dell'immagine determitato dall'inserimento delle 5 regioni nel 2-d Tree.

possiamo notare nel peggiore dei casi il costo dell'inserimento dell'n-esimo nodo può richiedere al massimo $O(n)$ confronti.

5.1.2 Cancellazione di nodi in un 2-d Tree.

La parte più complessa nella gestione dei K-d Tree è sicuramente la cancellazione di un nodo.

Consideriamo un 2-d Tree con T come radice, supponiamo di voler cancellare un elemento nella posizione (x, y).

Il primo passo nella procedura di cancellazione consiste nel verificare la presenza del nodo N da cancellare all'interno dell'albero. Dopo aver identificato il nodo con la proprietà che $N.XVal = x$ e $N.YVal = y$ se è una foglia, entrambi i

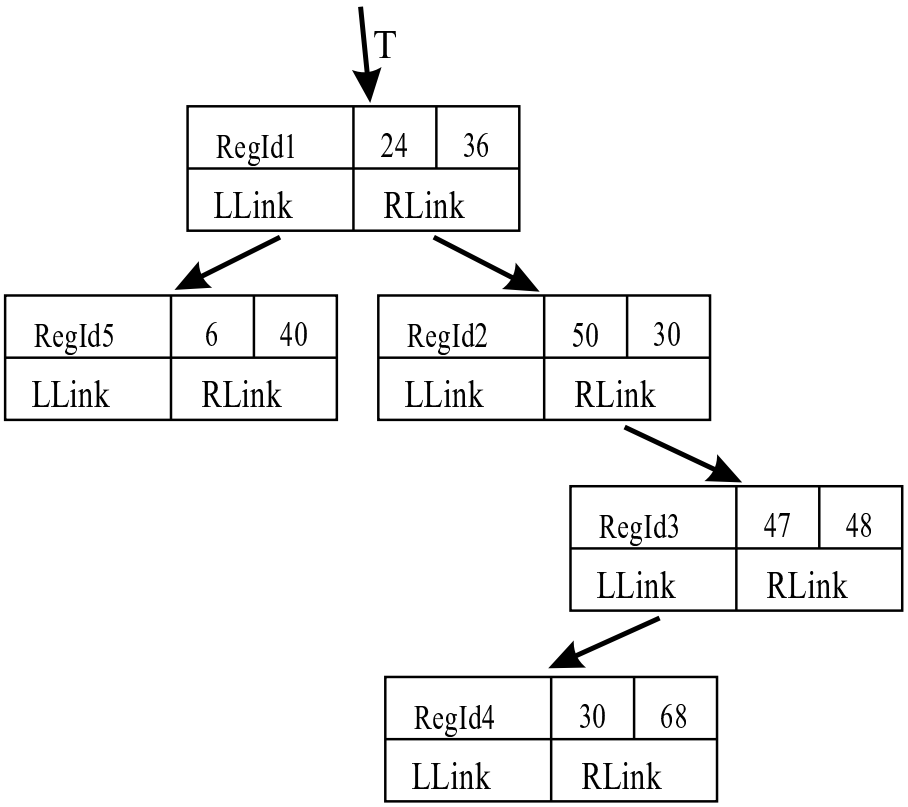


Figura 5.4: Grafico del 2-d Tree generato dall'inserimento delle 5 regioni.

puntatori a NULL, l'eliminazione del nodo si effettua molto semplicemente eliminando il nodo in questione, ponendo a NULL il puntatore nel nodo padre che ne memorizza il riferimento e liberando la relativa memoria.

Se il nodo da eliminare è un nodo interno la cancellazione è più complessa: in questo caso almeno uno dei due sottoalberi puntati dai puntatori N.LLink e N.RLink che identifichiamo con T_l e T_r rispettivamente, è non vuoto.

Il procedimento di cancellazione implica la ricerca di un nodo R appartenete ad T_l o a T_r che possa sostituire N in modo da poterlo cancellare (ricorsivamente) senza perdite di dati dal sottoalbero in cui risiede.

Per cui l'algoritmo di cancellazione applicato ad un nodo interno può essere scomposto in tre passi:

1^o **Passo:** trovare un sostituto R in $T_i \in (T_l, T_r)$ per il nodo N;

2^o **Passo:** sostituire i campi Info, XVal e YVal di R con quelli di N;

3^o **Passo:** cancellare ricorsivamente R da Ti.

Il passo critico in questo algoritmo è l'identificazione del nodo R che deve soddisfare la condizione di mantenere inalterate le relazioni posizionali tra i diversi punti.

Le condizioni che deve soddisfare R per essere un valido sostituto sono:

1. per ogni nodo M in T_l deve essere verificato che $M.XVal < R.XVal$ se $Level(N)$ è pari e che $M.YVal < R.YVal$ se $Level(N)$ è dispari;

2. per ogni nodo M in T_r deve essere verificato che $M.XVal \geq R.XVal$ se $Level(N)$ è pari e che $M.YVal \geq R.YVal$ se $Level(N)$ è dispari;

Ciò implica che se T_r non è vuoto ed il livello di N è pari, allora ciascun nodo in T_r che ha il valore più piccolo possibile di XVal all'interno di T_r è un adeguato sostituto per N.

Allo stesso modo se T_r non è vuoto ma il livello di N e dispari, allora ciascun nodo in T_r che ha il valore più piccolo possibile di YVal all'interno di T_r è un adeguato sostituto per N.

Supponiamo, ad esempio, di voler cancellare il nodo T che contiene le informazioni relative a RegId1 dall'albero di Figura 5.4.

Valutiamo il sottoalbero di destra che contiene i nodi RegId2, RegId3 e RegId4 alla ricerca di un sostituto R del nodo da cancellare:

poiché il livello della radice è pari il nodo sostitutivo è quello che presenta il valore minore di ascissa che nel nostro caso è RegId4. Sostituiamo quindi i campi Info, XVal e YVal del nodo radice con quelli di RegId4 e poi cancelliamo quest'ultimo.

In generale, la ricerca di un sostituto nel sottoalbero di sinistra è possibile solo sotto certe condizioni.

Se il livello di N è pari un sostituto appropriato R è ogni nodo in T_l tale che R.XVal sia il maggiore di tutti gli altri in T_l. Analogamente se il livello di N è dispari un sostituto appropriato R è ogni nodo in T_l tale R.YVal sia il maggiore di tutti gli altri in T_l.

Il problema nella ricerca di un sostituto nel sottoalbero di sinistra sussiste nel fatto che può esistere più di un nodo con il valore massimo di XVal o YVal in T_l e ciò fa si che il secondo passo della procedura di cancellazione invalidi la seconda proprietà nella definizione dei 2-d Tree.

In generale, si preferisce ricercare il sostituto di N nel sottoalbero di destra, se questo non è vuoto.

Se il sottoalbero di destra è vuoto allora scegliamo R in T_l con il più piccolo valore di XVal se il livello di N è pari od il più piccolo valore di YVal se il livello di N è dispari e modifichiamo il secondo passo nella maniera seguente:

2^o passo (modificato): sostituire i campi Info, XVal e YVal di R con quelli di N e porre $N.RLink = N.LLink$ e $N.LLink = NULL$.

5.1.3 Query su intervalli di valori in 2-d Tree.

Una query su un intervallo in un albero 2-d è una ricerca che permette di recuperare tutti i punti situati all'interno di una circonferenza centrata nel punto (X_c, Y_c) e di raggio r. La risposta a questa query recupera tutti i punti inseriti nell'albero che hanno distanza dal centro minore del raggio.

La struttura ad albero permette di recuperare questi punti in maniera intelligente escludendo a priori tutti i sottoalberi nell'albero complessivo che definiscono regioni che non hanno intersezione con il cerchio selezionato.

Nell'elaborazione di una query su un intervallo è necessario ricordare che ogni nodo N inserito all'interno del 2-d Tree definisce una regione.

In particolare, per i punti dell'esempio precedente, si ottiene:

1. Il nodo con etichetta RegId1 rappresenta la regione di tutti i punti del dominio applicativo;

2. Il nodo con etichetta RegId2 rappresenta la regione di tutti i punti (x, y) tali che $x \geq 24$;

3. Il nodo con etichetta RegId3 rappresenta la regione di tutti i punti (x, y) tali che $x \geq 24$ e $y \geq 30$;

4. Il nodo con etichetta RegId4 rappresenta la regione di tutti i punti (x, y) tali che $24 < x < 47$ e $y \geq 30$;

5. Il nodo con etichetta RegId5 rappresenta la regione di tutti i punti (x, y) tali che $x < 24$;

In generale ad ogni nodo N vengono associate quattro ulteriori costanti che assieme definiscono la regione associata.

XMIN e XMAX rappresentano rispettivamente gli estremi inferiore e superiore per le ascisse della regione. YMIN e YMAX rappresentano rispettivamente gli estremi inferiore e superiore per le ordinate della regione.

Queste costanti vanno a modificare la definizione del nodo nel seguente modo:

```
Kdnode_mod{
    Info: information;
    XVal, YVal: real;
    LLink, RLink: *Kdnode_mod;
    XMIN, XMAX, YMIN, YMAX: real;
}
```

Quando inseriamo nodi di questo tipo definiamo le nuove costanti nella seguente maniera:

1. La radice dell'albero ha rispettivamente XMIN e YMIN uguali all'estremo inferiore, XMAX e YMAX uguali all'estremo superiore, del dominio applicativo;

2. Se il generico nodo N ha P come padre con $level(P)$ pari, allora:

$$\begin{array}{lll} N.XMIN = P.XMIN & \text{se} & N = P.LLink \\ N.XMIN = P.XVal & \text{se} & N = P.RLink \\ N.XMAX = P.XVal & \text{se} & N = P.LLink \\ N.XMAX = P.XMAX & \text{se} & N = P.RLink \\ N.YMIN = P.YMIN & & \\ N.YMAX = P.YMAX & & \end{array} \tag{5.2}$$

3. Se il generico nodo N ha P come padre con $level(P)$ dispari, allora:

$$\begin{array}{lll} N.YMIN = P.YMIN & \text{se} & N = P.LLink \\ N.YMIN = P.YVal & \text{se} & N = P.RLink \\ N.YMAX = P.YVal & \text{se} & N = P.LLink \\ N.YMAX = P.YMAX & \text{se} & N = P.RLink \\ N.XMIN = P.XMIN & & \\ N.XMAX = P.XMAX & & \end{array} \tag{5.3}$$

Supponiamo, ad esempio, di eseguire la seguente query:

recuperare tutti i punti contenuti nel cerchio di centro $(50, 40)$ e raggio $r = 18$, è facile verificare che la risposta a questa query ritorna i punti corrispondenti a RegId2, RegId3. L'algoritmo di ricerca analizza la query nella seguente maniera:

1. La regione rappresentata da RegId1 è l'intera mappa e la circonferenza è completamente contenuta in essa, ma il relativo punto non vi appartiene, per cui proseguiamo analizzando i suoi figli.

2. La regione definita dal figlio di sinistra non interseca la circonferenza per cui lo escludiamo insieme a tutti i suoi eventuali figli.

   La regione definita dal figlio di destra, invece interseca il cerchio di ricerca e in particolare il punto RegId2 che è contenuto al suo interno lo inseriamo nell'insieme dei risultati, proseguiamo analizzando i figli di RegId2;

3. RegId3 è il figlio destro di RegId2 e la sua regione interseca il cerchio di ricerca, il punto è contenuto al suo interno per cui lo riportiamo come risultato.

4. L'ultimo punto è quello relativo a RegId4 la cui regione interseca il punto, ma non lo contiene per cui lo escludiamo dal risultato.

Il costo di una query su in intervallo di valori in un generico k-d tree presenta una complessità dell'ordine di $O(kn^{(1-\frac{1}{k})})$ che si specializza per $k = 2$ in $O(2\sqrt{n})$.

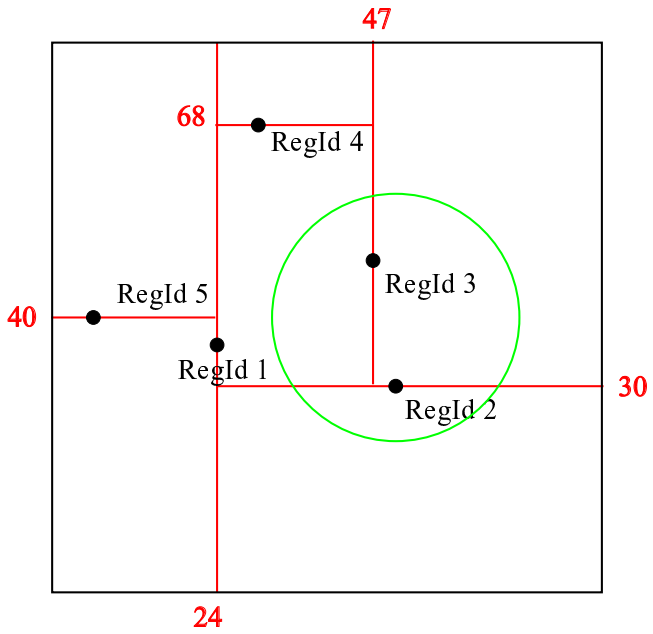


Figura 5.5: Schema di query su un intervallo di valori.

5.1.4 k-d tree per $k \geq 2$

Generalizzando l'algoritmo di inserimento per $k > 2$ i campi XVal e YVal vengono sostituiti da un vettore Val[] di k elementi dello stesso tipo.

Un albero k-d tree per essere considerato valido deve soddisfare per ogni nodo N le seguenti tre proprietà:

1. Definiamo $i = level(N) \text{mod} k$;

2. Per ogni nodo M nel sottalbero di sinistra del nodo N deve essere verificato che $M.Val[i] < N.Val[i]$;

3. Per ogni nodo M nel sottalbero di destra del nodo N deve essere verificato che $M.Val[i] >= N.Val[i]$.

Tutti gli algoritmi visti nel caso precedente per $k = 2$ si generalizzano in maniera analoga al caso $k > 2$.

In particolare per $k = 1$ si ottengono i classici alberi binari.

Gli svantaggi principali dell'impiego di un k-d tree sono la dipendenza dall'ordine di inserimento e la dispersione dei punti all'interno della struttura [**?**].

Una soluzione consiste nell'utilizzare una variante del k-d tree detta adaptive k-d tree che che utilizza serie di iperpiani che non necessariamente contengono

punti da memorizzare, ma che dividono la globalità dei punti in due sottoinsiemi con uguale cardinalità. Il procedimento di suddivisione viene iterato fino ad ottenere sottoinsiemi con un solo elemento.

In questo caso non è più necessario alternare le k direzioni degli iperpiani, infatti ogni nodo interno dell'albero contiene le informazioni sulla direzione e la posizione del relativo iperpiano.

I punti veri e propri vengono memorizzati al livello di foglie. Purtroppo questa struttura non è molto dinamica ed è di difficile aggiornamento.

Un'ulteriore variante BSP tree elimina rispetto alla precedente l'isorientamento degli iperpiani, fissa una soglia sotto la quale arrestare il processo di suddivisione dei sottospazi e memorizza più di un punto in ogni nodo foglia [**?**, **?**].

5.2 Point QuadTree.

I point QuadTree come i 2-d tree vengono impiegati per memorizzare punti bidimensionali, al contrario però nei QuadTree ogni punto divide la regione che lo contiene in quattro parti.

In un 2-d tree ogni nodo N inserito divide la regione che lo contiene in due parti tramite una linea passante per il punto $(N.XVal, N.YVal)$ in direzione verticale se il $level(N)$ è pari o orizzontale se il $level(N)$ è dispari.

Nel caso dei QuadTree Figura 5.6 le quatto parti in cui viene divisa la regione, dalle due linee passanti per il punto $(N.XVal, N.YVal)$, vengono denominate NO (Nord Ovest), SO (Sud Ovest), NE (Nord Est) e SE (Sud Est), ad ognuno di questi quadranti corrisponde un puntatore ad un nodo figlio.

Nei point QuadTree al contrario dei 2-d tree ogni nodo N può avere fino a quattro figli.

La definizione del nodo è la seguente:

```
QuadTreeNode {
    Info: Information;
    XVal: real;
    YVal: real;
    NO, SO, NE, SE: *QuadTreeNode;
}
```

5.2.1 Inserimento e ricerca di nodi in point QuadTree.

Illustriamo come vengono inseriti, in un point QuadTree con radice puntata da T, i punti della Tabella 5.1.1:

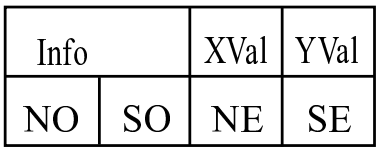


Figura 5.6: Struttura di un nodo di Point QuadTree.

1. Inizialmente l'albero è vuoto e l'inserimento del punto RegId1 avviene nel nodo radice: il dominio applicativo viene diviso in quatto parti da due rette ortogonali passanti per il punto $(24, 36)$.

2. L'inserimento del secondo punto RegId2, che rispetto a Regid1 è situato nel quadrante di SUD EST, implica la creazione di un nuovo nodo contenente RegId2 il cui riferimento viene memorizzato nel puntatore SE del nodo RegId1.

3. Il nodo contenente le informazioni su RegId3 si trova nel quadrante di NORD EST rispetto a RegId1 e poiché il puntatore a questa regione è libero creiamo un nuovo nodo e memorizziamo il suo riferimento nel puntatore NE del nodo contenente RegId1.

4. Quando inseriamo il punto corrispondente a RegId4 notiamo che si trova, come il precedente, nel quadrante si NORD EST rispetto a RegId1 e quindi si trova in un quadrante già occupato da un altro punto. Rispetto a quest'ultimo RegId4 si trova nel quadrante di NORD OVEST per cui creiamo un nuovo figlio del nodo RegId3, memorizzando il suo riferimento nel puntatore NO e creiamo una nuova suddivisione in quatto parti dell'area a NORD OVEST di RegId3.

5. L'ultimo punto da inserire, rappresentato RegId5, è localizzato rispetto a RegId1 nel quadrante di NORD OVEST che ha il rispettivo puntatore libero. Allora memorizziamo il riferimento a RegId5 nel puntatore NO di RegId1 ed eseguiamo la suddivisione in quattro nuove parti della regione di NORD OVEST.

I risultati di questo inserimento sono visibili nelle Figure 5.7 e 5.8 Ogni nodo N inserito nel Point QuadTree rappresenta, come nel caso dei 2-d tree, una regione che viene definita tramite le quattro costanti $XMin, XMax, YMin, YMax$ che esprimono i seguenti vincoli sui punti (x, y) appartenenti alla regione: $x \geq XMin, x < XMax, y \geq YMin, y < YMax$ e che andremo a inserire nella definizione del nodo.

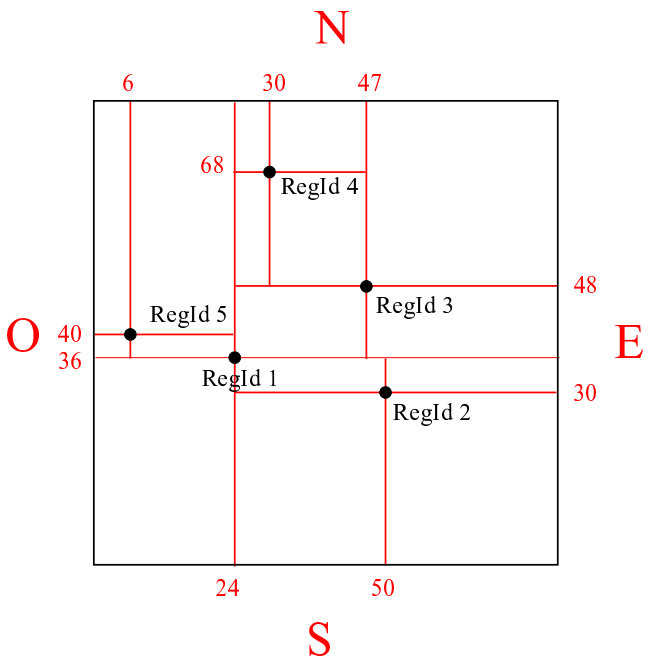


Figura 5.7: Schema di suddivisione del piano dell'immagine determitato dall'inserimento delle 5 regioni nel Point QuadTree.

I valori delle quatto costanti introdotte vengono assegnati al momento dell'inserimento del nodo N secondo i seguenti criteri:

1. Se il nodo N è la radice dell'albero puntato da T allora $XMin$ e $YMin$ assumono il valore minimo del domino applicativo, $XMax$ e $YMax$ assumono il valore superiore del dominio applicativo.

2. Se P e il padre di N, allora la Tabella 5.2.1 descrive come vengono definiti, in funzione del nodo padre, i valori delle quattro costanti per il rispettivo figlio.

Come si può notare sia nel caso dei 2-d tree che nei point QuadTree con n nodi il tempo di inserimento/ricerca nel caso peggiore è dell'ordine di $O(n-1)$ anche se, il più delle volte, i QuadTree sono più compatti dei relativi 2-d Tree.

5.2.2 Cancellazione nei Point QuadTree.

Quando cancelliamo un nodo N non foglia da un point QuadTree con radice T dobbiamo per prima cosa cercare, come nel caso dei 2-d tree, un sostituto del nodo da cancellare.

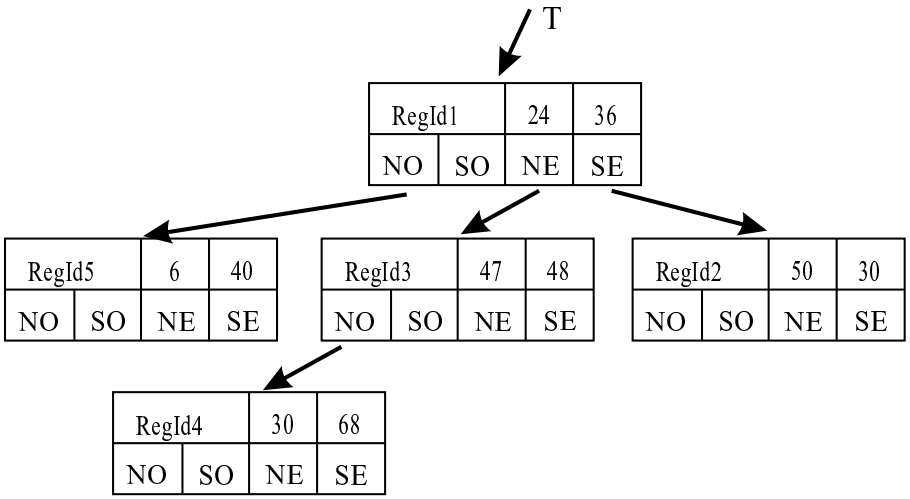


Figura 5.8: Grafico del QuadTree generato dall'inserimento delle 5 regioni.

Caso	N.XMin	N.XMax	N.YMin	N.YMax
N=P.NO	P.XMin	P.XMin+w/2	P.YMin+h/2	P.YMax
N=P.SO	P.XMin	P.XMin+w/2	P.YMin	P.YMin+h/2
N=P.NE	P.XMin+w/2	P.XMax	P.YMin+h/2	P.YMax
N=P.SE	P.XMin+w/2	P.XMax	P.YMin	P.YMin+h/2
Con $w = (P.XMax - XMin)$ e $h = (P.YMax - YMin)$				

Tabella 5.2: Assegnamento dalle costanti $XMin, XMax, YMin, YMax$ al nodo figlio in funzione dei valori del nodo padre in un Point QuadTree.

Nel caso dei nodi foglia la cancellazione è banale: basta porre a NULL il puntatore nel nodo padre che lo memorizza e liberare la memoria assegnata al nodo.

La cancellazione di un nodo interno è un procedimento complesso che non è sempre possibile senza effettuare una parziale organizzazione dei nodi già inseriti.

La determinazione di un nodo sostituto R avviene all'interno di uno dei quattro possibili sottoalberi del nodo da cancellare N in modo che siano verificate contemporaneamente le seguenti proprietà:

1. Ogni altro nodo R_1 nel sottoalbero N.NO è a NORD OVEST di R;
2. Ogni altro nodo R_2 nel sottoalbero N.NE è a NORD EST di R;
3. Ogni altro nodo R_3 nel sottoalbero N.SO è a SUD OVEST di R;
4. Ogni altro nodo R_4 nel sottoalbero N.SE è a SUD EST di R.

Come abbiamo accennato precedentemente non è sempre possibile, per tutti i possibili nodi interni N da cancellare, trovare un nodo sostituto R che soddisfi tutte e quatto le proprietà.

Quindi, nel peggiore dei casi, la cancellazione di un nodo può comportare il reinserimento di tutti quei nodi che impediscono il verificarsi delle quattro condizioni precedenti.

5.2.3 Query su intervalli di valori in Point QuadTree.

Le query sugli intervalli di valori vengono trattate esattamente nella stessa maniera delle rispettive sui 2-d tree. L'intervallo di ricerca viene rappresentato da una circonferenza con centro (Xc, Yc) e raggio r.

Una query su un intervallo permette di recuperare tutti i punti contenuti all'interno della circonferenza. Poiché ogni nodo nel point QuadTree rappresenta una regione, l'algoritmo di ricerca scarta i sottoalberi rispetto ai quali la relativa regione non ha intersezioni con il cerchio di ricerca.

Il processo di query inizia esaminando il nodo radice T, se la regione associata alla radice, cioè l'intero dominio applicativo, ha un intersezione con l'intervallo di ricerca si valuta se anche il nodo è interno all'intervallo e in caso affermativo si include il punto nei risultati della ricerca.

Il passo successivo determinato dall'esistenza di intersezione, indipendentemente dall'appartenenza del punto al risultato della query, è l'esame dei quatto sottoalberi puntati da NO, NE, SO, SE. Se non esiste intersezione con la regione associata al nodo tutto il sottoalbero che si sviluppa dal nodo viene scartato perché sicuramente non contiene punti interessanti, in caso contrario si valuta l'appartenenza del punto alla regione.

L'algoritmo di ricerca termina quando non ci sono più nodi da esaminare. [**?**, **?**]

5.3 MX QuadTree.

In tutte le strutture precedentemente analizzate la "forma" dell'albero dipende dall'ordine in cui i vari oggetti vengono inseriti.

In particolare l'ordine influisce sull'altezza finale dell'albero che è anche la grandezza che determina la complessità, in termini di operazioni elementari, delle operazioni di inserimento e ricerca di un generico valore.

Sia nei 2-d tree che nei point QuadTree ogni nodo rappresenta una regione che nei primi viene divisa in due nei secondi in quattro sottoregioni dall'inserimento di un nuovi elementi. Queste suddivisioni sono irregolari dato che dipendono dalla posizione $(N.XVal, N.YVal)$ dello specifico punto all'interno della regione.

Al contrario, lo scopo principale del MX QuadTree è quello di assicurare che la "forma" dell'albero risulti indipendente dal numero e dall'ordine dei nodi inseriti, con il più il vantaggio di semplificare gli algoritmi di ricerca e di cancellazione.

Gli MX QuadTree implicano la definizione di una griglia quadrata di 2^K elementi di lato dove K è una costante, fissa per l'applicazione, che riflette la granularità con cui vogliamo studiare gli elementi che andranno a popolare la nostra applicazione.

I nodi del MX QuadTree hanno la stessa struttura dei nodi del point QuadTree ampliati con la definizione delle quattro costanti $XMin, XMax, YMin, YMax$ che i figli derivano in maniera analoga dai valori del loro genitore secondo la Tabella 5.3. In questo caso ogni volta che viene inserito un nuovo punto la

Caso	N.XMin	N.XMax	N.YMin	N.YMax
N=P.NO	P.XMin	P.XMin+w/2	P.YMin+w/2	P.YMax+w
N=P.SO	P.XMin	P.XMin+w/2	P.YMin	P.YMin+w/2
N=P.NE	P.XMin+w/2	P.XMax+w	P.YMin+w/2	P.YMax+w
N=P.SE	P.XMin+w/2	P.XMax+w	P.YMin	P.YMin+w/2
Con $w = (N.XMax - N.XMin)$				

Tabella 5.3: Assegnamento dalle costanti $XMin, XMax, YMin, YMax$ al nodo figlio in funzione dei valori del nodo padre in un MX quadTree.

regione che lo contiene viene divisa esattamente nel mezzo.

5.3.1 Inserimento e ricerca di nodi un MX QuadTree.

Illustriamo come vengono inseriti, in un MX QuadTree con radice puntata da T, i punti dell'esempio precedente 5.3.1:

Regione	cetroid(XVal,YVal)	Xeff	Yeff
RegId 1	(24,36)	2	4
RegId 2	(50,30)	5	3
RegId 3	(47,48)	5	5
RegId 4	(30,68)	4	7
RegId 5	(06,40)	0	4

Tabella 5.4: Coordinate del baricentro delle regioni calcolate per $K = 3$.

Per memorizzare questi valori fissiamo $K = 3$. Il valore di K ci permette di determinare i valori delle ascisse e delle ordinate effettive secondo le seguenti

formule:

$$(Xeff, Yeff) = (\frac{N.XVal}{2^K}, \frac{N.YVal}{2^K}) \tag{5.4}$$

Procediamo con l'inserimento dei punti della Tabella 5.3.1 che individuano le

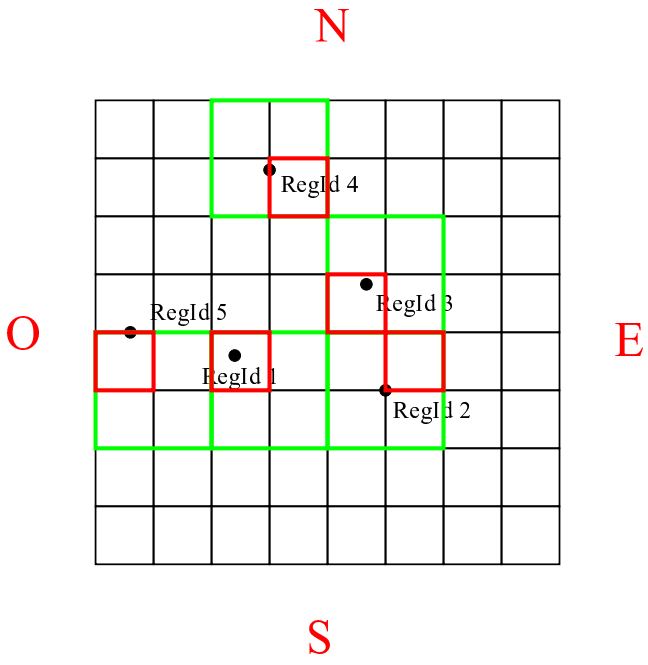


Figura 5.9: Schema di suddivisione del piano dell'immagine determitato dall'inserimento delle 5 regioni nell MX QuadTree.

celle di Figura 5.9 nella griglia 8 x 8.

1. L'inserimento del punto relativo a RegId1 con coordinate $(2, 4)$ causa la creazione dei seguenti nodi: il nodo radice che rappresenta l'intera regione ad il nodo rappresentante il quadrante di SUD OVEST di dimensione 4x4 che contiene il punto. All'interno di questa sottoregione il punto si trova nel quadrante di NORD EST di dimensione 2x2 e nel quadrante di NORD OVEST di dimensione 1x1.

   Creiamo il nodo radice ed un percorso di due nodi ai livelli 1 e 2 e memorizziamo il punto nel nodo foglia al livello 3.

2. L'inserimento del punto relativo a RegId2 con coordinate $(5, 3)$ causa la creazione/attraversamento dei seguenti nodi: il nodo radice rappresenta l'intera regione ed il nodo rappresentante il quadrante di SUD EST di dimensione 4x4 che contiene il punto. All'interno di questa sottoregione il punto

si trova nel quadrante di NORD OVEST di dimensione 2x2 e nel quadrante di NORD EST di dimensione 1x1.

Partendo dal nodo radice creiamo un percorso di due nodi ai livelli 1 e 2 e memorizziamo il punto nel nodo foglia al livello 3.

3. L'inserimento del punto relativo a RegId3 con coordinate $(5, 5)$ causa la creazione/attraversamento dei seguenti nodi: Il nodo radice che rappresenta l'intera regione ed il nodo rappresentante il quadrante di NORD EST di dimensione 4x4 che contiene il punto. All'interno di questa sottoregione il punto si trova nel quadrante di SUD OVEST di dimensione 2x2 e nel quadrante di SUD OVEST di dimensione 1x1.

   Partendo dal nodo radice creiamo un percorso di due nodi ai livelli 1 e 2 e memorizziamo il punto nel nodo foglia al livello 3.

4. L'inserimento del punto relativo a RegId4 con coordinate $(4, 7)$ causa la creazione/attraversamento dei seguenti nodi: Il nodo radice che rappresenta l'intera regione ed il nodo rappresentante il quadrante di NORD OVEST di dimensione 4x4 che contiene il punto. All'interno di questa sottoregione il punto si trova nel quadrante di NORD EST di dimensione 2x2 e nel quadrante di SUD EST di dimensione 1x1.

   Partendo dal nodo radice creiamo un percorso di due nodi ai livelli 1 e 2 e memorizziamo il punto nel nodo foglia al livello 3.

5. L'inserimento del punto relativo a RegId5 con coordinate $(0, 4)$ causa la creazione/attraversamento dei seguenti nodi: Il nodo radice che rappresenta l'intera regione ad il nodo rappresentante il quadrante di SUD OVEST di dimensione 4x4 che contiene il punto.

   All'interno di questa sottoregione il punto si trova nel quadrante di NORD OVEST di dimensione 2x2 e nel quadrante di NORD OVEST di dimensione 1x1.

   Partendo dal nodo radice creiamo un percorso di due nodi ai livelli 1 e 2 e memorizziamo il punto nel nodo foglia al livello 3.

Come si può notare in Figura 5.10 questa struttura ha dimensioni molto maggiori rispetto alle precedenti, ma presenta diversi vantaggi sotto l'aspetto della gestione dei dati. Siccome i nodi contenenti le informazioni risiedono sempre al livello delle foglie il percorso per raggiungerle richiede sempre lo stesso numero di attraversamenti pari a K. L'utilizzo di una struttura indipendente dall'ordine di inserimento dei nodi implica "forme" più regolari riducendo le variazioni dei tempi

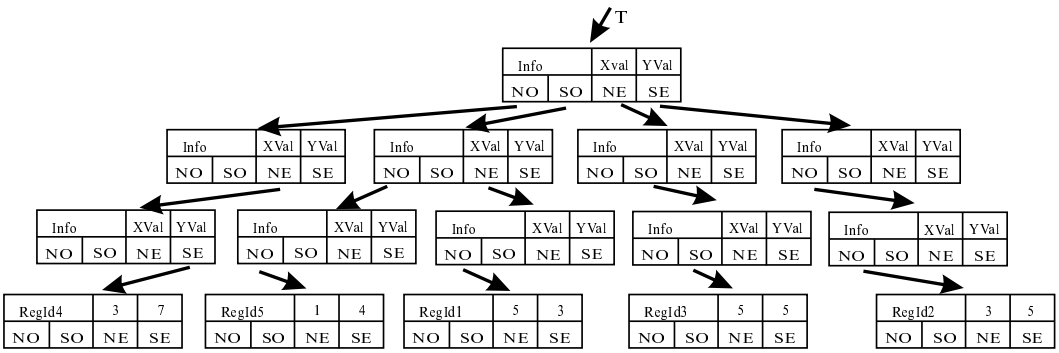


Figura 5.10: Grafico del MX QuadTree generato dall'inserimento dei centri delle 5 regioni.

di recupero. Inoltre l'impiego degli MX QuadTree permette di risolvere i problemi di cancellazione illustrati precedentemente nel caso dei point QuadTree dove si richiedeva il reinserimento di alcuni nodi.

La complessità computazionale nell'inserimento o nella ricerca di un nodi è dell'ordine di $O(K)$.

5.3.2 Cancellazione negli MX QuadTree.

La cancellazione di un nodo P in un MX QuadTree è un'operazione semplice perché tutti i nodi sono memorizzati nelle foglie.

Se N è un nodo interno all'MX QuadTree con radice T allora la regione, che implicitamente rappresenta questo nodo, deve contenere almeno un punto effettivamente memorizzato in una foglia nel sottoalbero del nodo N.

Se vogliamo cancellare il punto di coordinate (x, y) nell'albero puntato da T dobbiamo mantenere la proprietà precedente. Quando andiamo a cancellare un nodo prima di tutto verifichiamo la sua presenza all'interno dell'albero, poi cancelliamo il riferimento nel nodo padre del nodo P e liberiamo la relativa memoria.

Il passo successivo consiste nel verificare che nel nodo padre di P ci sia almeno un puntatore non vuoto, in caso contrario anche il nodo padre non è più necessario e può essere eliminato.

La verifica sui nodi interni procede a ritroso collassando tutti i nodi nel cammino dal nodo P al nodo radice T. Ad esempio, la cancellazione del nodo relativo a RegId1 implica che anche il nodo padre venga eliminato perché rimane vuoto.

5.3.3 Query su intervalli di valori in MX QuadTree.

Le query sugli intervalli di valori negli MX QuadTree sono eseguite nella stessa maniera dei Point QuadTree con due piccole differenze.

La prima è che il contenuto di XMin, XMax, YMin, YMax è diverso. La seconda riguarda il fatto che essendo i punti memorizzati nelle foglie il controllo sull'appartenenza del punto alla circonferenza di ricerca deve essere effettuato solo in corrispondenza delle foglie.

Le query sugli intervalli negli MX QuadTree risultano molto efficienti in particolare la complessità computazionale è dell'ordine di $O(N + 2^K)$ dove N è il numero dei punti nel risultato della query [**?**].

5.4 R-Tree.

Gli R-tree sono una struttura dati che, tra le varie applicazioni, può essere impiegata per memorizzare regioni rettangolari di un'immagine all'interno delle quali sono definite un insieme di proprietà.

Questi rettangoli vengono derivati determinando per ogni regione di forma arbitraria, ottenuta tramite precedenti elaborazioni dell'immagine, l'MBB (Minimum Bounding Box) il rettangolo minimo che la contiene.

Gli R-tree sono particolarmente adatti a memorizzare grosse quantità di dati su disco infatti, essendo gli accessi a questa risorsa più lenti degli accessi in memoria, gli R-tree permettono di ottimizzare il numero delle letture da questa risorsa. Ogni R-Tree ha associato un ordine che è un intero K.

Ogni nodo può memorizzare al massimo K regioni rettangolari, ma ne deve contenere almeno K/2 con eccezione della radice. Ciò significa che ogni nodo interno, con l'eccezione della radice, deve essere almeno mezzo pieno.

Solitamente K viene scelto in modo che un nodo risieda su una sola pagina in modo da poter essere recuperato con un solo accesso al disco. Memorizzare in ogni nodo almeno K/2 regioni rettangolari implica che l'altezza di un R-tree, anche in presenza di molte regioni, si mantenga contenuta diminuendo ulteriormente il numero di accessi al disco necessari alle operazioni di ricerca.

L'altezza di un R-tree contente N nodi è al massimo $|\log - K/2N| - 1$ poiché ci sono almeno K/2 ramificazioni per ogni nodo.

La struttura di un R-tree è del tipo:

```
RtreeNode{
    Info1, Info2, ... , InfoK: Information;
    Ret1, Ret2, ... , RetK: rettangolo;
    P1, P2, ... , PK: *RtreeNode;
}
```

Ogni nodo contiene K elementi che memorizzano una regione rettangolare il campo InfoK contiene l'identificativo della regione, con limiti definiti dal rettangolo RetK e che contiene al suo interno i nodi del sottoalbero puntato da PK.

Possiamo considerare l'R-tree, percorrendolo dai livelli più alti ai più bassi, come una gerarchia di regioni rettangolari incluse una all'interno dell'altra.

Gli elementi memorizzati in nodo interno N rappresentano regioni rettangolari fittizie o di raggruppamento N.RetK che contengono al loro interno tutte le regioni del sottoalbero con radice N.PK. All'interno dei nodi foglia invece, risiedono elementi che memorizzano gli MBB di una regione reale ed un riferimento ad ulteriori proprietà. In Figura 5.11 sono rappresentate una serie di regioni rettangolari es-

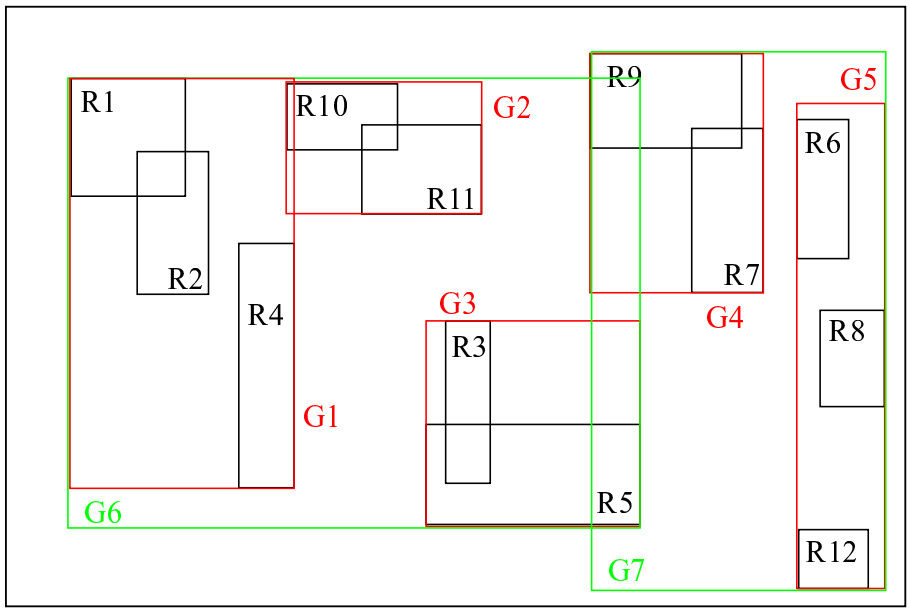


Figura 5.11: Schema delle regioni e dei relativi raggruppamenti da inserire nell'R-Tree.

tratte da precedenti elaborazioni dell'immagine. In particolare, sono evidenziati in colore nero gli MBB delle regioni in rosso e verde i rettangoli di raggruppamento di primo e secondo livello.

L'R-tree ottenuto inserendo in successione le precedenti regioni rettangolari è rappresentato in Figura 5.12. Ogni rettangolo può essere memorizzato in modi diversi, ad esempio tramite una coppia di punti oppure tramite un punto, una lunghezza e una larghezza.

In particolare, un R-tree può memorizzare anche insiemi di punti sotto forma di rettangoli con lunghezza e larghezza nulla.

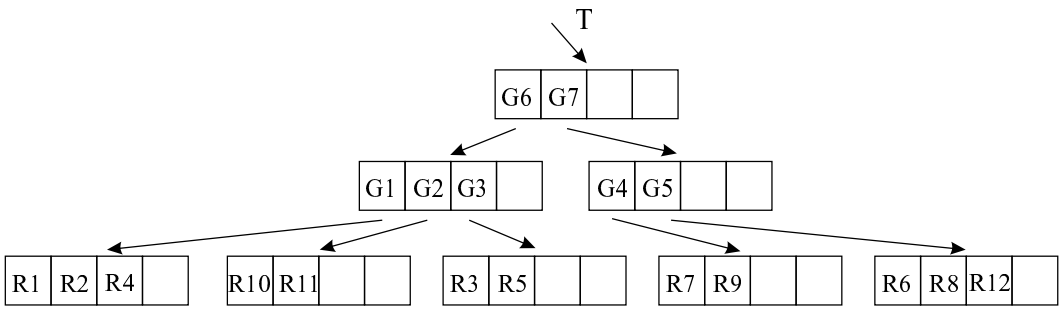


Figura 5.12: Grafico del R-Tree generato dall'inserimento delle regioni dello schema precedente.

5.4.1 Ricerche ed aggiornamenti di R-tree.

L'algoritmo di ricerca percorre l'albero dalla radice alle foglie in maniera simile ad un B-tree.

Siccome i rettangoli di raggruppamento possono essere parzialmente sovrapposti, le ricerche in un R-tree, invece di seguire un unico percorso dalla radice alla foglia da recuperare, devono percorrere in successioni serie di percorsi e ciò implica l'accesso ad un numero maggiore di nodi rispetto, ad esempio, al caso della ricerca in un QuadTree.

Supponiamo di avere un R-tree con T come radice, l'algoritmo di ricerca permette di recuperare tutti le regioni N.RetK che sono sovrapposte, interamente o parzialmente, alla regione S utilizzata per la ricerca. L'algoritmo di ricerca si divide in due passi:

1. **Ricerca nel sottoalbero.** Se T non è una foglia controlla ogni figlio E del nodo per determinare quando E.ret è contenuto all'interno di S.

   In caso affermativo riesegui l'algoritmo di ricerca sul sottoalbero avente radice E.P.

2. **Ricerca nelle foglie.** Se T è una foglia controlla ogni figlio E per determinare se E.ret è contenuto in S. Se è così includo E nella risposta

L'algoritmo di inserimento aggiunge i nuovi elementi al livello delle foglie; i nodi pieni vengono divisi in due pieni solo a metà e questo processo può propagasi verso l'alto fino alla radice.

L'algoritmo di inserimento si divide in quatto passi:

1. **Trova spazio per il nuovo elemento.** Invoca ScegliFoglia per selezionare un nodo foglia L dove inserire E.

2. **Aggiungi un elemento al nodo foglia.** Se in L c'è spazio per un nuovo nodo inseriscilo, in caso contrario invoca DividiNodo per ottenere L ed LL contenenti E più gli elementi precedenti.

3. **Propaga i cambiamenti verso l'alto.** Invoca AdattaAlbero su L e passa anche LL se c'è stata una divisione.

4. **Crescita verso l'alto.** Se la propagazione delle divisioni dei nodi verso l'alto coinvolge anche la radice crea una nuova radice con due figli che derivano dalla divisione di quella vecchia.

L'algoritmo ScegliFoglia percorre l'albero partendo dalla radice e sceglie le ramificazioni che richiedono il minor allargamento del relativo rettangolo fino a raggiungere un nodo foglia in cui inserire il nuovo nodo:

1. **Inizializzazione.** Assegna ad N il valore della radice.

2. **Se N è una foglia**, restituisci N.

3. **Scegli il sottoalbero.** Se N non è una foglia assegna ad F il valore del puntatore al nodo che memorizza il rettangolo F.ret che richiede, per includere E.ret, l'espansione minore. Nel caso di possibilità multiple scegli quella con area minore.

4. **Discesa fino alla foglia.** Assegna ad N il valore del figlio puntato da F.P e riparti dal punto 2.

L'algoritmo AdattaAlbero sale da una foglia L fino alla radice adattando se necessario, i rettangoli di raggruppamento e propagando le divisioni dei nodi se necessario:

1. **Inizializza.** Setta N=L se L era stato diviso setta NN=LL per il secondo nodo.

2. **Controlla se fatto.** Se N è la radice termina.

3. **Adatta i rettangoli di raggruppamento nel nodo padre.** Definiamo P il nodo padre di N e En l'elemento nel nodo padre che contiene il riferimento al nodo figlio. Adatta En.Reg in modo da contenere tutti i rettangoli nel nodo N.

4. **Propaga le divisioni verso l'alto.** Se N e il compagno diviso NN esistono crea una nuovo elemento Enn la cui regione Enn.Reg contiene tutti i rettangoli in NN e con Enn.P che punta ad NN. Aggiungi se c'è posto Enn in P, in caso contrario invoca di nuovo DividiNodo per produrre P e PP contenenti Enn e tutti gli altri elementi di P.

Quando gli algoritmi precedenti decidono di inserire un nuovo nodo e il riferimento a quest'ultimo non trova posto nel rispettivo padre poiché tutti e K i puntatori sono impegnati è necessario eseguire la divisione del nodo tramite l'algoritmo DividiNodo.

Questo algoritmo ridistribuisce i K+1 nodi tra il nodo pieno ed un nuovo nodo, la ridistribuzione deve avvenire in maniera tale che sia molto improbabile il fatto che entrambi i nodi vengano esaminati nelle successive ricerche.

Siccome la decisione sull'effettuare o no l'esplorazione di un nodo dipende dal fatto che il suo rettangolo sia sovrapposto a quello di ricerca allora la ridistribuzione dei nodi deve essere tale da minimizzare l'area totale dei due rettangoli ottenuti dalla divisione dei nodi nei due gruppi.

Presentiamo ora diverse versioni dell'algoritmo di divisione del nodo con diverse complessità computazionali:

le versioni più pesanti ottimizzano meglio la divisione del nodo rispetto alle più leggere. La prima versione, detta esaustiva, è la via più diretta per trovare la divisione del nodo di area minima e consiste nel generare tutti i possibili raggruppamenti per scegliere quello di area minore.

In questo caso il numero di possibilità è approssimativamente 2^{K-1} e poiché un valore ragionevole per K è 50, allora il numero di possibili divisioni è molto elevato.

Vengono quindi implementate versioni diverse dell'algoritmo di divisione di costo rispettivamente quadratico e lineare in K. Questo algoritmo tenta di mantenere l'area totale piccola, anche se non è detto che determini l'area più piccola possibile.

L'algoritmo a costo quadratico seleziona due dei (K+1) elementi che andranno inseriti uno nel primo e uno nel secondo nodo, scegliendo la coppia che se fosse inserita nello stesso gruppo consumerebbe più spazio, cioè quelli per cui l'area del rettangolo che contiene entrambe gli elementi meno la somma delle aree dei due elementi viene massimizzata.

Gli elementi rimanenti vengono assegnati ai due gruppi uno alla volta. Ad ogni passo l'espansione dell'area necessaria ad inserire l'elemento correntemente esaminato viene valutata per entrambi i raggruppamenti e l'elemento viene inserito nel gruppo che richiede l'espansione minore.

L'algoritmo DividiNodo con complessità quadratica è costituito dai seguenti passi:

1. **Selezione il primo elemento di ogni gruppo.** Applica l'algoritmo SelezionaSeme per scegliere i due elementi che diventano i primi dei due gruppi. Assegna i due elementi a ciascun gruppo.

2. **Controlla se fatto.** Se tutti gli elementi sono stati assegnati al rispettivo

nodo termina, altrimenti se i rimanenti servono a raggiungere la capacità minima del nodo (K/2) allora assegnali tutti a lui e termina.

3. **Selezione il successivo elemento.** Applica l'algoritmo SelezionaSuccessivo per selezionare il successivo elemento da assegnare da aggiungerlo al nodo il cui rettangolo di raggruppamento deve essere espanso di meno per contenerlo.

   Se tutti e due i nodi devono essere allargati della stessa quantità seleziona quello con area totale minore oppure quello con meno elementi. Ritorna al punto 2.

L'algoritmo SelezionaSeme seleziona due elementi che diventano i primi elementi dei due nodi frutto della divisione del nodo pieno:

1. **Calcola l'inefficienza provocata dall'inserimento di due elementi nello stesso gruppo.** Per ogni coppia di elementi E1 ed E2 crea il rettangolo R che contiene E1.reg e E2.reg. Calcola $D = area(R) - area(E1.reg) - area(E2.reg)$.

2. **Scelta della coppia migliore.** Scegli la coppia con D maggiore.

L'algoritmo SelezionaSuccessivo divide i rimanti elementi nei due nodi.

1. **Determina il costo provocato dall'inserimento di un elemento in un nodo.** Per ogni elemento E non ancora inserito calcola D1 che rappresenta l'incremento d'area del rettangolo di raggruppamento necessario per inserire l'elemento E nel primo gruppo e D2 che rappresenta l'incremento per l'inserimento nel secondo gruppo.

2. **Seleziona l'elemento migliore.** Seleziona l'elemento con la massima differenza tra D1 e D2.

L'algoritmo con complessità lineare in K è identico al precedente, ma usa una versione differente per SelezionaSeme, mentre SelezioneSuccessivo sceglie uno qualunque tra gli elementi rimanti.

L'algoritmo SelezionaSeme di complessità lineare rispetto alla capacità K del nodo seleziona i primi due elementi che vengono inseriti uno nel primo e l'altro nel secondo nodo.

1. **Trova i rettangoli estremi lungo le due dimensioni.** Lungo le due dimensioni trova gli elementi che hanno il più alto lato inferiore e il più basso lato superiore. Registra la distanza tra i due.

2. **Aggiusta la forma del raggruppamento rettangolare.** Normalizza la distanza dividendola per l'intera larghezza dell'insieme delle regioni lungo la corrispondente dimensione.

3. **Seleziona la coppia estrema.** Seleziona la coppia con il valore più alto di distanza normalizzata lungo le due dimensioni.

5.4.2 Cancellazione di elementi in un R-tree.

La cancellazione in un R-tree sarebbe semplice poiché l'elemento da cancellare risiede in un nodo foglia, ma l'esistenza del vincolo sul contenuto minimo di un nodo, che deve contenere almeno K/2 elementi, implica che debba essere effettuato un ulteriore controllo sul fatto che il nodo non si svuoti al di sotto del minimo consentito.

L'algoritmo Cancella rimuove l'elemento E dall'R-tree:

1. **Trova il nodo contenente il record da cancellare.** Invoca la funzione TrovaFoglia per localizzare il nodo foglia L contenente l'elemento E. Termina se l'elemento non esiste.

2. **Cancella il record.** Rimuovi E da L

3. **Propaga le modifiche.** Invoca CondensaNodi passando il nodo foglia L.

4. **Albero minimo.** Se il nodo radice ha un solo figlio, dopo che l'albero è stato elaborato, il figlio diventa la nuova radice.

Algoritmo TrovaFoglia è analolgo all'algoritmo di ricerca precedentemente illustrato.

L'algoritmo CondensaNodi riceve il riferimento L del nodo foglia contenente l'elemento E appena cancellato, se il nodo in questione ha meno di K/2 elementi anche quest'ultimo deve essere eliminato e gli elementi rimanenti ridistribuiti.

La cancellazione di nodi semivuoti si propaga verso l'alto, se necessario. Adatta tutti i rettangoli di raggruppamento sul cammino dal nodo eliminato alla radice rendendoli i più piccoli possibili.

1. **Inizializzazione.** Assegna N=L. Definisci Q come un set di nodi dove inserire tutti quei nodi da eliminare o che devono essere vuotati.

2. **Trova l'elemento padre.** Se N è la radice salta al punto 6. In caso contrario assegna a P il valore del nodo padre di N e a En l'elemento che punta ad N in P.

3. **Elimina i nodi semivuoti.** Se N ha meno di K/2 nodi cancella En da P e aggiungi N al set Q.

4. **Sistema il rettangolo di raggruppamento.** Se N non viene eliminato aggiusta En.reg per includere, nel minore spazio possibile, gli elementi rimanenti in N.

5. **Sali di un livello nell'albero.** Assegna N=P e ritorna al punto 2.

6. **Reinserisci gli elementi orfani.** Reinserisci tutti gli elementi presenti nel set Q. Gli elementi superstiti dai nodi foglia eliminati sono reinseriti tramite l'algoritmo di inserimento già illustrato.

5.4.3 Aggiornamenti ad altre operazioni.

L'operazione di aggiornamento di un nodo, che implica la variazione delle dimensioni del rettangolo che definisce la regione, necessita la cancellazione, la modifica e il reinserimento del nodo.

Altri tipi di operazioni di ricerca come quelle che permettono di recuperare tutti gli elementi completamente inclusi nel rettangolo di ricerca oppure quelli in cui è contenuto il rettangolo di ricerca sono facilmente implementabili con piccole variazioni degli algoritmi precedenti [**?**].

5.4.4 Evoluzioni degli R-tree.

Per ridurre i problemi legati alla possibile sovrapposizione di regioni viene introdotta una variante degli R-tree che prende il nome di $R^+ - tree$. Gli $R^+ - tree$ adottano il clipping durante l'inserimento delle regioni cioè viene eliminata la sovrapposizione dei rettangoli di raggruppamento che risiedono nello stesso livello dell'albero.

Gli oggetti che intersecano più di un intervallo rettangolare vengono memorizzati in più di una pagina.

Adottando questa politica, il recupero delle regioni che intersecano l'intervallo di ricerca viene velocizzato poiché il numero di percorsi da esplorare viene ridotto. L'inserimento di nuovi elementi all'interno di un $R^+ - tree$ richiede di percorre più volte il percorso dalla radice ai nodi foglia in funzione del numero di intersezioni che questa ha con i vari raggruppamenti presenti all'interno dell'indice.

Durante l'attraversamento dell'albero la regione viene divisa in n frammenti che vengono inseriti nei vari nodi foglia in funzione della loro posizione.

Per ogni frammento inserito, il relativo nodo deve essere espanso quel tanto per contenerlo e per ogni espansione bisogna evitare tutte le possibili sovrapposizione. In alcuni casi non è possibile espandere alcun rettangolo di raggruppamento, per contenere il nuovo nodo, senza causare sovrapposizioni. Si crea quindi una situazione di stallo dove alcuni raggruppamenti devono essere sciolti e i relativi elementi reinseriti.

Negli $R^+ - tree$ in caso di divisione di un nodo le divisioni successive si propagano non solo verso l'alto come negli R-tree, ma anche verso il basso e possono portare ad ulteriori frammentazioni dei raggruppamenti.

Nelle cancellazioni di nodi per prima cosa, bisogna localizzare tutti i frammenti nei vari nodi eliminarli e verificare per ogni nodo le condizioni sulla capacità minima in modo da condensare quelli che non la soddisfano, ridistribuendo gli elementi rimanenti.

Gli $R^* - tree$ sono una ulteriore variante degli R-tree che rimuovono una serie di difetti individuati da un attento studio del comportamento degli R-tree sotto diverse distribuzioni di dati. In particolare, questi studi confermano che la fase di inserimento è quella più critica sotto il profilo delle prestazioni tra tutte le possibili operazioni.

Gli $R^* - tree$ adattono una politica detta di reinserimento forzato: se un elemento deve essere inserito in un nodo che è già pieno il nodo non viene diviso, ma si rimuovono p elementi dal nodo e si reinseriscono all'interno dell'albero.

Al parametro p in genere viene assegnato un valore pari al 30 % della capacità K del nodo.

Gli $R^* - tree$ tengono in conto i seguenti obbiettivi di gestione degli inserimenti:

- La sovrapposizione tra i diversi rettangoli di raggruppamento deve essere minimizzata. Più bassa è la sovrapposizione minore è la probabilità che si debbano seguire percorsi multipli.

- La lunghezza dei margini deve essere minimizzata. Dove il margine è la somma delle lunghezze dei bordi del rettangolo. Assumendo un area fissa l'oggetto con il margine minore è il quadrato. Minimizzare il margine invece dell'area implica che le regioni assumano una forma vicina al quadrato.

- L'area coperta dai nodi interni deve essere minimizzata

- L'utilizzo dell'area di immagazzinamento deve essere massimizzata.

In conclusione, gli $R^* - tree$ differiscono dagli R-tree per gli algoritmi di inserimento, mentre quelli di cancellazione e ricerca rimangono sostanzialmente immutati [**?**, **?**].

5.5 Indici multidimensionali e memorizzazione delle feature.

Nelle pagine precedenti abbiamo illustrato una serie di strutture dati in grado di agevolare la ricerca ed il recupero di dati multidimensionali. In particolare strutture come R-tree, R*-tree, Point QuadTreee, ecc ... sono impiegate all'interno di molti sistemi di CBVQ per velocizzare e razionalizzare il recupero dei risultati delle interrogazioni.

Se vogliamo implementare un indice basato, ad esempio, su un R-tree, per ogni tipologia di feature estratta dal sistema sulle immagini memorizzate nel database, debbono essere soddisfatti i seguenti vincoli:

1. La "distanza" tra due elementi memorizzati deve corrispondere alla distanza Euclidea dei punti appartenenti allo spazio delle feature che li rappresentano;

2. La dimensionalità dello spazio delle feature deve mantenersi ragionevolmente bassa.

Molti indici multi dimensionali degenerano in maniera esponenziale al crescere della dimensione dello spazio delle feature.

In particolare per i Point QuadTree la complessità è proporzionale all'iperpiano che delimita la regione di query e l'iperpiano cresce esponenzialmente con l'aumentare della dimensionalità. Verifiche pratiche dimostrano che i sistemi basati sugli R-tree gestiscono senza problemi spazi fino a 20 dimensioni.

In particolare la variante $R^* - tree$ impiegata in QBIC ottimizza efficacemente i tempi di recupero. Vediamo quindi le caratteristiche e le tipologie di feature estratte dalle immagini da QBIC e le operazioni adottate per far si che le feature soddisfino la coppia di vincoli precedentemente definiti. QBIC estrae tre tipi di feature:

quelle sul colore, quelle sulla forma e quelle sulle tessiture. Le feature sul colore impiegano un istogramma a K-elementi.

Concettualmente K può essere molto grande se si vogliono rappresentare tutte le possibili combinazioni di tre canali di colori.

In pratica però tutti i sistemi per ridurre il valore di K raggruppano insieme i colori simili e scelgono un singolo colore come rappresentativo dell'intero raggruppamento.

In questo modo otteniamo un istogramma con K=256 o K=64 colori rappresentativi di altrettanti gruppi. Ogni elemento dell'istogramma quindi rappresenta la percentuale di pixel che sono più simili ad un dato colore.

La similitudine tra due istogrammi viene poi valutata attraverso la distanza quadratica:

$$D^2_{Quad}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^T A(\vec{x} - \vec{y}) \tag{5.5}$$

dove la matrice A con elementi a_{ij} definisce la similitudine tra i colori i e j.

Come si può notare facilmente, la distanza Euclidea è un caso particolare della distanza quadratica con A che diventa la matrice identità.

La differenza tra le due distanze consiste nel fatto che la distanza quadratica confronta ogni colore con tutti gli altri, mentre in quella Euclidea si considera solo quello in posizione omologa, quindi la seconda è molto più semplice da calcolare della prima.

Quindi le feature sul colore non verificano entrambi i vincoli precedentemente definiti in particolare la dimensionalità dello spazio del colore è troppo alta (K=64,256) e la distanza tra feature non è Euclidea.

Le feature sulla forma sono composte da un insieme di venti coefficienti che rappresentano attributi caratteristici della forma come l'area, la circolarità, l'eccentricità, l'orientazione dell'asse maggiore ed un insieme di momenti invarianti.

La distanza tra due vettori di coefficienti viene calcolata attraverso la distanza Euclidea pesata, dove a ciascun elemento della matrice diagonale A è associato un coefficiente particolare.

In questo caso solamente il vincolo sulla dimensione dello spazio delle feature non è rispettato.

Le feature sulle tessiture sono una terna di attributi che rappresentano rugosità, contrasto e direzionalità, la funzione di distanza è la distanza Euclidea pesata e in questo caso entrambi i vincoli sono verificati.

Illustriamo una serie di soluzioni per entrambi i problemi creati dall'introduzione della coppia di vincoli sopracitati.

L'idea principale che sta dietro ad entrambe le soluzioni consiste nell'impiegare un approccio basato sul concetto di "signature": si tratta di creare un filtro che può introdurre falsi positivi, ma che non introduce falsi negativi, cioè come risultato della query si ottengono più immagini di quelle strettamente necessarie, ma non si perde nessuna delle immagini realmente importanti.

La soluzione a questi problemi permette di mappare le feature nello spazio X in uno spazio X' ottenuto attraverso la trasformazione $X' = f(X)$ dove è possibile definire una distanza $D'(\cdot)$ che sottostima la distanza $D(\cdot)$ definita sullo spazio X; cioè:

$$D'(\vec{X'}, \vec{Y'}) \leq D(\vec{X}, \vec{Y}) \tag{5.6}$$

Una trasformazione che verifica questa proprietà, quando si utilizza la distanza Euclidea, è, ad esempio, la proiezione di punti tridimensionali sul piano xy troncando la coordinata z.

La ragione per cui si vuole che l'equazione precedente è sia verificata è l'intenzione di garantire che una query su un intervallo non perda nessun dei possibili risultati.

Ad esempio, la query "recupera tutti i vettori di feature $\vec{X}$ con distanza minore di ε da un vettore di query Q", restituisce tutti gli elementi tali che $D(\vec{X}, \vec{Q}) \leq \varepsilon$, invece operando nello spazio a minore dimensionalità si recuperano tutti gli $\vec{X}'$ tali che $D'(\vec{X}', \vec{Y}') \leq \varepsilon$ e poiché è verificata la disequazione $D'(\vec{X}', \vec{Y}') \leq D(\vec{X}, \vec{Q}) \leq \varepsilon$ si conclude che $X \subseteq X'$.

Illustriamo due diverse forme della trasformazione f.

Illustriamo la prima applicandola alle feature sul colore dove si determina un estremo inferiore per la distanza quadratica (Quadratic Distance Bounding), mentre per la seconda utilizziamo funzioni che preservano la distanza che sfruttano la trasformata Karhunen Loeve (KL).

Quadratic Distance Bounding determina una nuova metrica di distanza, molto più semplice di quella quadratica, derivata dalla determinazione del valore medio di ciascun canale appartenete allo spazio di colore.

KL mappa vettori ad n elementi in vettori, sempre ad n elementi con il vantaggio però che dopo la trasformazione abbiamo la maggior parte delle "informazioni" concentrate nei primi per cui possiamo semplicemente azzerare un certo numero di elementi a partire dalla fine del vettore ottenendo uno spazio a dimensione ridotta.

La chiave per il recupero efficiente di immagini attraverso gli istogrammi passa infatti attraverso la definizione di una nuova distanza di tipo Euclideo più semplice da calcolare in grado di approssimare la distanza quadratica.

Per ottenere ciò determiniamo l'estremo inferiore di D_{Quad} attraverso un multiplo della distanza media cioè:

$$D_{Quad} \geq kD_{Med} \tag{5.7}$$

Come è ovvio la distanza media è molto più semplice da calcolare ed in particolare agevola l'impiego di indici multidimensionali.

Ciò permette di utilizzare D_{Med} come filtro attraverso il quale ottenere un sottoinsieme rilevante di immagini sul quale applicare la D_{Quad} per selezionare solo gli elementi veramente importanti.

Infatti se vogliamo recuperare gli elementi con $D_{Quad} \leq \varepsilon$ tutto quello che dobbiamo fare è selezionare i vettori per i quali è verificata la disequazione $kD_{Med} \leq \varepsilon$, facendo ciò siamo sicuri di recuperare gli stessi elementi della precedente più qualche falso positivo da eliminare attraverso l'applicazione della D_{Quad}.

La prestazione globale ne risulta notevolmente migliorata poiché D_{Quad} elabora un numero molto ridotto di elementi ed inoltre è possibile implementare un indice di dimensioni contenute.

La distanza media viene definita su uno spazio di dimensioni molto ridotte formato solamente da tre elementi ottenuti calcolando la media degli elementi appartenenti ai tre canali di colore, attraverso i quali è definito lo spazio di colore delle immagini.

Supponiamo di trovarci, ad esempio, nello spazio RGB dove ogni punto dell'immagine è definito da una terna di numeri tipicamente compresi tra 0 e 255.

Il colore medio dell'immagine è determinato dalla terna:

$$\begin{aligned} \vec{x} &= (R_{Med}, G_{Med}, B_{Med}) \\ R_{Med} &= \tfrac{1}{N}\textstyle\sum_{p=1}^{N} R(p) \\ G_{Med} &= \tfrac{1}{N}\textstyle\sum_{p=1}^{N} G(p) \\ B_{Med} &= \tfrac{1}{N}\textstyle\sum_{p=1}^{N} B(p) \end{aligned} \tag{5.8}$$

Dove N è il numero di pixel dell'immagine e R(p), G(p) e B(p) sono rispettivamente le intensità dei colori rosso, verde e blu del p-esimo pixel.

Dati i vettori 3D dei colori medi di due immagini, definiamo D_{Med} attraverso la distanza Euclidea nello spazio 3D dei vettori del colore medio:

$$D_{Med}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^T(\vec{x} - \vec{y}) \tag{5.9}$$

Dimostriamo ora per quali valori di k la disequazione tra D_{Quad} e D_{Med} risulta verificata.

Premettiamo le seguenti definizioni:

W matrice di K x K elementi ,che contiene informazioni su ognuno dei K colori che rappresentano l'istogramma. In particolare il generico elemento w_{ij} si ottiene dalla formula:

$$w_{ij} = (R_i, G_i, B_i)^T(R_j, G_j, B_j) \tag{5.10}$$

Analogamente definiamo W' (K-1) x (K-1) attraverso la matrice W nella seguente maniera:

$$w'_{ij} = w_{ij} - w_{iK} - w_{Kj} + w_{KK} \tag{5.11}$$

Stessa cosa per la matrice di similitudine A ottenendo A'.

TEOREMA 1 *Dati D_{Quad} e D_{Med}, W' e A' matrice positiva e semidefinita possiamo dire che:*

$$D^2_{Quad} \geq \lambda_1 D^2_{Med} \tag{5.12}$$

dove λ_1 è il più piccolo degli autovalori dell'espressione:

$$A'z = \lambda W'z \tag{5.13}$$

Questo risultato ci assicura, come precedentemente illustrato, di poter migliorare le prestazioni nell'esecuzione delle query eseguendo un filtraggio preventivo, applicando la funzione di distanza D_{Med} sui colori medi, computazionalmente poco onerosa. Successivamente, sul sottoinsieme ridotto di immagini in uscita dalla funzione di filtro, si applica la distanza D_{Quad}, sull'istogramma completo, in modo da ottenere come risultato solamente le immagini più "simili".

La seconda modalità di riduzione della dimensione delle feature non introduce una nuova definizione di distanza e quindi è applicabile solamente nel caso in cui la distanza sia già di tipo Euclideo.

Utilizzando la notazione per le operazioni tra matrici la distanza può essere rappresentata dalla formula:

$$(\vec{x} - \vec{q})^T(\vec{x} - \vec{q}) \tag{5.14}$$

Dove $\vec{x}$ e $\vec{q}$ sono una coppia di vettori di dimensione N con N=20 che rappresentano le feature sulla forma delle regioni calcolate rispettivamente sull'immagine da confrontare e sull'immagine di query.

Se entrambi i vettori sono trasformati attraverso una trasformazione ortonormale A, la distanza tra i due vettori viene preservata.

Ciò può essere facilmente dimostrato dato che vale la proprietà che $A^TA = I$ quindi la distanza può essere calcolata secondo la seguente formula:

$$\begin{array}{c}(A\vec{x} - A\vec{q})^T(A\vec{x} - A\vec{q}) = (A(\vec{x} - A\vec{q}))^T A(\vec{x} - A\vec{q}) = \\ = (\vec{x} - \vec{q})^T A^T A(\vec{x} - \vec{q}) = \sum_i (x_i - q_i)^2\end{array} \tag{5.15}$$

Illustriamo un metodo che sottostima la distanza tra due feature utilizzando uno spazio a minore dimensionalità.

La riduzione dello spazio è ottenuta semplicemente selezionando un sottoinsieme dell'insieme delle feature trasformate.

Se $a_i x$ rappresenta l'i-esima feature trasformata ed a_i i-esima riga della matrice di trasformazione A, si ottiene la riduzione dimensionale assegnando ad a_i il vettore nullo per tutte le righe con indice i compreso nell'intervallo $(m+1, n)$, dove n ed m sono rispettivamente il numero di feature estratte prima e dopo la riduzione di dimensionalità.

La somma dell'equazione precedente può essere scomposta in due parti:

$$\sum_{i=1}^{n}(x_i - q_i)^2 - \left(\sum_{i=1}^{m}(\vec{a}_i(\vec{x} - \vec{q}))^2 + \sum_{i=m+1}^{n}(\vec{a}_i(\vec{x} - \vec{q}))^2\right) = 0 \tag{5.16}$$

Eseguendo un semplice arrangiamento di questa equazione possiamo vedere che l'errore introdotto, calcolando la distanza su un sottoinsieme delle feature trasformate, è dato da:

$$\delta = \sum_{i=1}^{n}(x_i - q_i)^2 - \sum_{i=1}^{m}(\vec{a}_i(\vec{x} - \vec{q}))^2 = \sum_{i=m+1}^{n}(\vec{a}_i(\vec{x} - \vec{q}))^2 > 0 \qquad (5.17)$$

Da questa equazione si può concludere che la distanza calcolata sulla troncatura delle feature trasformate sottostima sempre la distanza sulle feature originali, poiché $\delta > 0$, quindi non si perde nessun potenziale risultato valido.

Le trasformazioni possibili appartengono a due grandi famiglie:

- Trasformazioni "data dipendenti", come la trasformazione di Karhunen Loeve, che necessita di un campione di dati per eseguire un'analisi statistica per la determinazione della matrice di trasformazione A.
- Trasformazioni "data indipendenti", come le trasformazioni Coseno Discreto (DCT), Harr o Fourier, dove la matrice A è determinata a priori.

Le trasformazioni "data dipendenti" possono essere personalizzate per uno specifico insieme di immagini così da raggiungere prestazioni migliori.

Lo svantaggio è dovuto al fatto che l'insieme delle immagini deve essere statico (ad esempio CD-ROM) in modo da evitare complesse riorganizzazioni necessarie per riadattare la matrice di trasformazione A al nuovo insieme di immagini.

Un ulteriore problema è dovuto alla scelta del campione utilizzato per la determinazione della matrice di trasformazione A che deve essere rappresentativo dell'intero database.

Al contrario delle precedenti, le trasformazioni "data indipendenti" non presentano problemi di riorganizzazione al variare della popolazione del database, ma ottengono anche prestazioni peggiori. [**?**, **?**]

Riassunto delle capacità di indicizzazione	
Motore	Struttura ad indice implementata
QBIC	$R^* - Tree$ su tutte e tre le tipologie di feature estratte (colore, tessitura e forma) insieme a tecniche di riduzione dimensionale.
WindSurf	Nessun tipo di indice implementato: le feature estratte vengono memorizzate sequenzialmente.
WebSeek	nessuna politica di indicizzazione
VisualSEEk	viene implementato un R-Tree che memorizza il rettangolo che contiene la regione (MBR) e un point QuadTree che memorizza le coordinate del centro della regione.

Tabella 5.5: Raffronto delle tipologie di indici multidimensionali implementati nei motori commerciali analizzati.

Capitolo 6

Stato dell’arte dei MMDBMS

Le moderne tecnologie permettono di acquisire, manipolare, trasmettere e archiviare con facilità grandi quantità di immagini e filmati in Database Multimediali, tuttavia i sistemi in grado di ritrovare agevolmente queste informazioni sono ancora limitati.

Gli strumenti della generazione precedente si basavano, per l’esecuzione delle query, sulle descrizioni testuali che accompagnavano ogni immagine e che venivano annotate manualmente da chi le inseriva nel database.

Queste descrizioni sono però sempre soggettive, quasi mai sono esaustive del contenuto dell’immagine e comunque, l’annotazione manuale diventa impensabile quando il numero di immagini da gestire aumenta in maniera esponenziale.

Se potessimo realizzare un programma in grado di estrarre, in maniera automatica, frasi semanticamente rilevanti da un’immagine, il problema della ricerca e del recupero di queste ultime potrebbe essere svolto dai classici motori di ricerca testuali.

La possibilità di realizzare questi algoritmi, per un insieme di immagini senza nessuna restrizione sui soggetti contenuti, però va oltre le possibilità fornite dalle attuali conoscenze nel campo del riconoscimento delle immagini.

Il processo tramite il quale la mente umana elabora le caratteristiche fisiche di un’immagine, identifica gli oggetti presenti in essa e associa descrizioni semantiche non si adatta alle capacità elaborative dei sistemi odierni. Ciò che invece si adatta molto bene alle caratteristiche dei computer odierni è la capacità di eseguire misure quantificabili, altamente ripetitive e di archiviarle in una memoria a lungo termine.

6.1 QBIC.

Query **B**y **I**mage **C**ontent (**QBIC**) [**?**] sono una serie di nuovi strumenti software, sviluppati presso i laboratori della IBM, che permettono l'espressione e l'esecuzione di query, basate sul contenuto non semantico, di una serie di immagini e filmati memorizzati in un database.

Le proprietà cardine di QBIC che lo caratterizzano maggiormente rispetto ai sistemi di precedenti generazioni sono:

- L'utilizzo del contenuto non semantico delle immagini e dei filmati, come i colori, le tessiture e le forme, per la risoluzione delle query;
- La possibilità di esprimere, con l'ausilio di strumenti di tipo grafico, una vasta gamma di query tramite schizzi a mano libera o attraverso la selezione di parti di immagini

QBIC fornisce una serie di funzioni per l'estrazione di proprietà quantificabili (feature) e le relative misure di similitudine (distance measure) tramite le quali esegue le query definite dall'utente.

Le feature attualmente implementate sono:

- **Colore medio:** viene estratto il valore medio per le tre componenti (R, G, B) del colore di ciascun pixel dell'immagine e la similitudine viene calcolata su questi tre valori;
- **Istogramma dei colori:** viene calcolata la distribuzione dei colori in uno spazio a 256 valori predeterminati, la similitudine viene calcolata su questa distribuzione;
- **Disposizione:** l'immagine viene suddivisa in un set predeterminato di regioni e per ognuna di queste viene estratto il colore dominante quindi la similitudine viene calcolata per ogni regione, così la misura è sensibile alla disposizione dei colori;
- **Tessitura:** vengono estratte informazioni sulle tessiture presenti nell'immagine, la similitudine è determinata sulla base di diversi attributi come direzionalità, rugosità e contrasto;
- **Testo:** è possibile associare manualmente a ciascuna immagine una descrizione testuale tramite la quale selezionare per le successive elaborazioni solo una determinata categoria di immagini;
- **Definita dall'utente:** Qualunque altra feature e metrica di confronto generata dall'utente

QBIC estrae le feature che vengono memorizzate in tabelle contenute in un database.

Può interfacciarsi con i database relazionali più diffusi come Oracle o Db2 oppure con le utility dbm di UNIX che forniscono funzioni simili a quelle di un comune database.

In particolare in quest'ultimo caso basta creare una directory col nome uguale al nome del database che si vuole aprire e porre in questa directory set di feature con estensione uguale al nome del catalogo in cui vogliamo inserirle. Il processo di interrogazione del database può essere scomposto in due fasi:

la prima composta da operazioni preliminari che consistono nell'elaborazione di tutte le immagini presenti nel database per l'estrazione di tutte le feature definite, nella creazione di una serie di indici per ogni feature che si vuole considerare e nella generazione dei thumbernail; versioni rimpicciolite e di dimensioni predeterminate delle immagini originarie da impiegare per la visualizzazione dei risultati della query.

Questa serie di operazioni prende il nome di "database population" e deve essere eseguita ogni volta che si crea un nuovo catalogo di immagini.

Gli indici creati in questa fase e memorizzati all'interno del database, vengono impiegati nella seconda fase detta "database query" per recuperare le immagini che rispetto ad una data metrica di similitudine presentano le feature più simili a quelle estratte dai campioni forniti in vari modi dall'utente.

QBIC implementa una vasta gamma di possibilità di query in particolare una prima classificazione distingue tra:

- **Query semplici:** coinvolgono un solo tipo di feature, ad esempio cercare tutte le immagini nel database che hanno un istogramma dei colori simile a quello dell'immagine fornita dall'utente;

- **Query complesse:** coinvolgono più tipi di feature con pesi variabili che ne determinano la relativa importanza e che vengono suddivise in due ulteriori categorie:

- **Multi-feature:** in cui vengono recuperate le immagini che più soddisfano contemporaneamente tutte le feature specificate, ad esempio cercare tutte le immagini che hanno una distribuzione dei colori e una tessitura simile a quella dell'immagine fornita dall'utente;

- **Multi-pass:** in cui vengono specificati più feature di ricerca che sono applicate in sequenza in modo che al risultato ottenuto in base al confronto con la prima feature venga applicato un riordinamento in funzione delle altre. Ad esempio cercare tutte le immagini che hanno una distribuzione dei

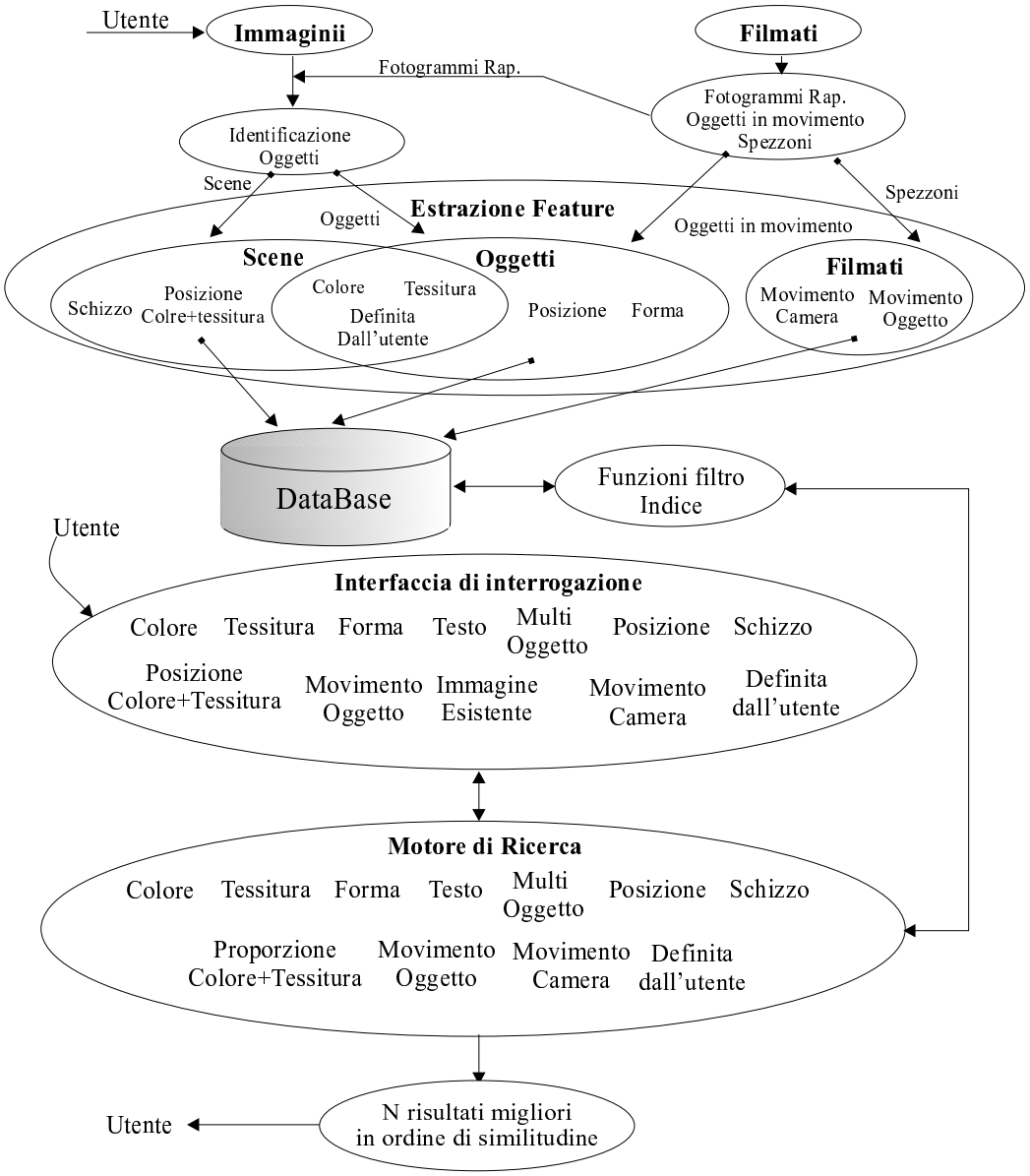


Figura 6.1: Architettura del motore di ricerca QBIC.

colori simile e riordinare il risultato in funzione della presenza di una data tessitura.

Oltre ad implementare diverse modalità di query **QBIC** fornisce pure diversi metodi per specificare l’immagine o parti di essa da utilizzare come termini di paragone:

- **“Query by example”:** l’utente specifica il nome di un’immagine interna al catalogo che si sta esaminando, di cui sono già disponibili le feature, come immagine di paragone con cui confrontare tutte le altre presenti nel database estraendo quelle più simili;

- **“Query by image”:** l’utente specifica il nome di un’immagine esterna, da cui non sono state ancora estratte le feature, come immagine di paragone con cui confrontare tutte le altre presenti nel database.

  Prima di eseguire la ricerca si estraggono le feature coinvolte nella query dall’immagine esterna non ancora elaborata e poi si esegue il confronto con le feature presenti nel database estraendo quelle più simili;

- **“Query by picker”:** si dà la possibilità diretta all’utente di disegnare una serie di regioni rettangolari con un dato colore dominante e di eseguire una ricerca sulla presenza, nelle immagini catalogate, di quella distribuzione spaziale.

  Non è necessario definire tutte le regioni presenti, ma solo quelle che consideriamo più rilevanti, le parti non colorate sono semplicemente ignorate. La parziale limitazione, data dal poter specificare solo rettangoli, permette di codificare la query sotto forma di stringa di testo contenente la dimensione dell’area di disegno e una serie di rettangoli di un dato colore con certe dimensioni contenuti all’interno dell’area di disegno.

  Questa possibilità risulta molto utile in ambiente client-server dove il client trasmette questo semplice messaggio al server che così ricostruisce il disegno della query originale estrae le feature e le compara con quelle memorizzate all’interno del suo database.

6.1.1 Fase 1: “Database Population”.

Gli elementi che costituiscono il dominio dell’applicazione e che sono trattati nella fase di popolazione consistono essenzialmente in immagini statiche e filmati.

Per le immagini statiche in particolare si parla di scene, immagini complete, e di oggetti che possono essere contenuti all’interno della scena. Per ogni oggetto

identificato vengono estratte le relative feature (feature extraction) che vengono immagazzinate assieme a quelle estratte dalle scene.

L'identificazione degli oggetti all'interno della scena (object identification) avviene in maniera semi automatica cioè con l'intervento dell'utente che si avvale di strumenti software che ne facilitano l'estrazione.

Due di questi strumenti per la separazione degli oggetti dallo sfondo sono:

- **"Flood fill":** permette di separare gli oggetti dalla scena quando questi hanno una dinamica dei colori sufficientemente diversa da quella dello sfondo in modo da poter determinare in maniera semi automatica una soglia adeguata.

  Questa soglia viene determinata in maniera interattiva selezionando, tramite il puntatore del mouse, una serie di punti e specificando se questi appartengono all'oggetto o allo sfondo.

- **"Snakes":** permette all'utente di separare gli oggetti dallo sfondo disegnando manualmente una spezzata vicino al contorno dell'oggetto che automaticamente viene adattata via software al contorno eseguendo la massimizzazione dell'intensità del gradiente dell'immagine lungo la spezzata.

Per i filmati invece l'operazione di inserimento nel database delle feature è più complessa poiché queste vengono estratte solo per le immagini statiche.

I filmati quindi vengono divisi in spezzoni (shots) e per ogni spezzone viene estratto un fotogramma rappresentativo (R–frame) dell'azione che si svolge nello spezzone.

Una serie di fotogrammi continui possono essere raggruppati nello stesso spezzone se: riguardano la stessa scena, coinvolgono una singola operazione della telecamera, contengono oggetti od eventi distinti e persistenti.

Dopo aver determinato i vari spezzoni per ognuno di essi si associa un R-frame che è un'immagine statica e viene elaborata al pari delle altre immagini vere e proprie. La determinazione del fotogramma rappresentativo può essere il primo, l'ultimo o quello centrale dello spezzone, oppure nel caso di lunghe panoramiche dove non è possibile determinare un R–frame rappresentativo dell'intero spezzone si può definire una fusione di più fotogrammi in un unica immagine dove sono sovrapposti tutti gli oggetti coinvolti.

6.1.2 Fase 2: "Database Query".

Durante la query i criteri di ricerca immessi dall'utente con l'aiuto di un'interfaccia grafica (query interface) sono elaborati per estrarne le feature che vengono

comparate, tramite funzioni di similitudine (match engine), con quelle contenute nel database. Successivamente gli N risultati migliori, sotto forma di thumbernail, vengono visualizzati sullo schermo.

Il metodo più semplice per definire i criteri di ricerca può essere quello di fornire un'immagine di prova e recuperare quelle più simili, oppure può avvenire in maniera semplificata attraverso strumenti di tipo grafico:

tramite dei cursori si specificano le incidenze dei tre colori principali dell'immagine per la feature del colore medio, oppure, per quella sull'istogramma, si disegnano una serie di barre di colori diversi e con dimensioni proporzionali alle percentuali di colore che compongono il relativo istogramma.

Per la tessitura è possibile selezionare tramite il mouse tra una serie di possibili campioni. Query più complesse permettono all'utente di disegnare una serie di regioni di un dato colore in modo da legare le zone di colore alla loro posizione, oppure permettono di tracciare linee a mano libera per definire le forme ed i contorni degli oggetti da ricercare.

Gli utenti interagiscono con la (query interface) che genera un input per la (match engine), questa interagisce anche col modulo di (filtering/indexing) che permette di velocizzare le ricerche nel database.

L'utilizzo di sistemi di filtraggio e di indicizzazione permette di ridurre il decadimento delle prestazioni quando le dimensioni del database aumentano.

L'uso di funzioni veloci di filtraggio permette di effettuare una prima scrematura dei dati in modo da passare alla match engine, potenzialmente più lenta, solamente una parte dei dati senza perdite di accuratezza nel recupero dei risultati.

6.1.3 Implementazione.

QBIC gestisce tre categorie principali di feature: sul colore, sulle tessitura e sulle forme che sono state trattate in dettaglio nel capitolo 2.

Le feature sul colore impiegate in QBIC sono calcolate sul valore medio nello spazio (R, G, B) e in quello (H, S, V) e sull'istogramma dei colori a 256 elementi impiegando una metrica quadratica per il calcolo della distanza (Paragrafo 2.1).

Per la tessitura si impiega una versione modificata dell'algoritmo di Tamura [**?**] che prende in considerazione tre parametri: rugosità, contrasto e direzionalità che rappresentano rispettivamente scala, vivacità e presenza di una direzione principale (Paragrafo 2.2).

Per le forme si considerano i seguenti parametri: circolarità, eccentricità, orientamento dell'asse maggiore e un set di momenti algebrici invarianti (Paragrafo 2.3).

QBIC sfrutta anche la trasformata KL (Paragrafo 5.5) per ridurre le dimensioni delle feature ed indicizzarle tramite un R-tree (Paragrafo 5.4).

6.2 WindSurf.

Analizziamo ora il sistema **WindSurf** (**W**avelet-based **IND**exing of **I**mages **U**sing **R**egion **F**ragmentation) per il recupero di immagini basato sul contenuto [**?**].

Windsurf si occupa dei problemi relativi all'estrazione e all'indicizzazione delle feature caratteristiche dell'immagine e si occupa del recupero delle immagini appartenenti ad un certo Database che rispondono alle richieste poste dall'utente.

Windsurf impiega un nuovo modello di similitudine nel quale ogni immagine è prima decomposta in regioni sulla base di informazioni su colore e tessitura rappresentate nel dominio delle frequenze, mentre la similitudine complessiva tra coppie di immagini è raggiunta attraverso la somme di quelle delle regioni che le compongono.

Ogni regione viene rappresentata attraverso un insieme di attributi che sono in grado di caratterizzarle in maniera univoca; tali attributi prendono il nome di feature di similarità.

Le feature vengono confrontate attraverso opportune funzioni di distanza che misurano la similitudine tra le regioni a cui esse appartengono.

Il sistema di organizzazione di gruppi di immagini avviene attraverso la definizione delle collezioni. Ogni collezione, individuata per mezzo di un nome, rappresenta un insieme di percorsi (path) che possono indicare directory su disco, su CD-ROM, indirizzi URL o comunque immagini memorizzate in posizioni diverse del file-system.

La collezione è gestita attraverso una struttura tabellare:

ogni voce della tabella contiene informazioni relative alla collezione stessa ed è in grado di caratterizzarla in modo univoco; tali informazioni sono il nome della collezione, lo spazio di colore utilizzato, il tipo e il livello di trasformata Wavelet e la metrica utilizzata per il clustering.

Ad ogni voce della tabella delle collezioni è associata una tabella delle immagini in essa contenute; ogni elemento di tale tabella indica sia il cammino da seguire per poter recuperare l'immagine che la dimensione della sottobanda a bassa frequenza utilizzata durante il processo di clustering. Il sistema si propone di svolgere due operazioni principali.

1. Estrazione delle regioni e delle relative feature: questa operazione consente di indicizzare tutte le immagini che appartengono ad una data collezione.

   Affinché l'elaborazione possa avvenire, l'utente deve specificare il nome con cui intende identificare la collezione oppure la collezione che vuole aprire, nel caso in cui questa sia già stata elaborata.

   Dopo l'immissione del nome con cui identificare la collezione, è possibile specificare una serie di parametri rilevanti per l'estrazione di regioni e per il

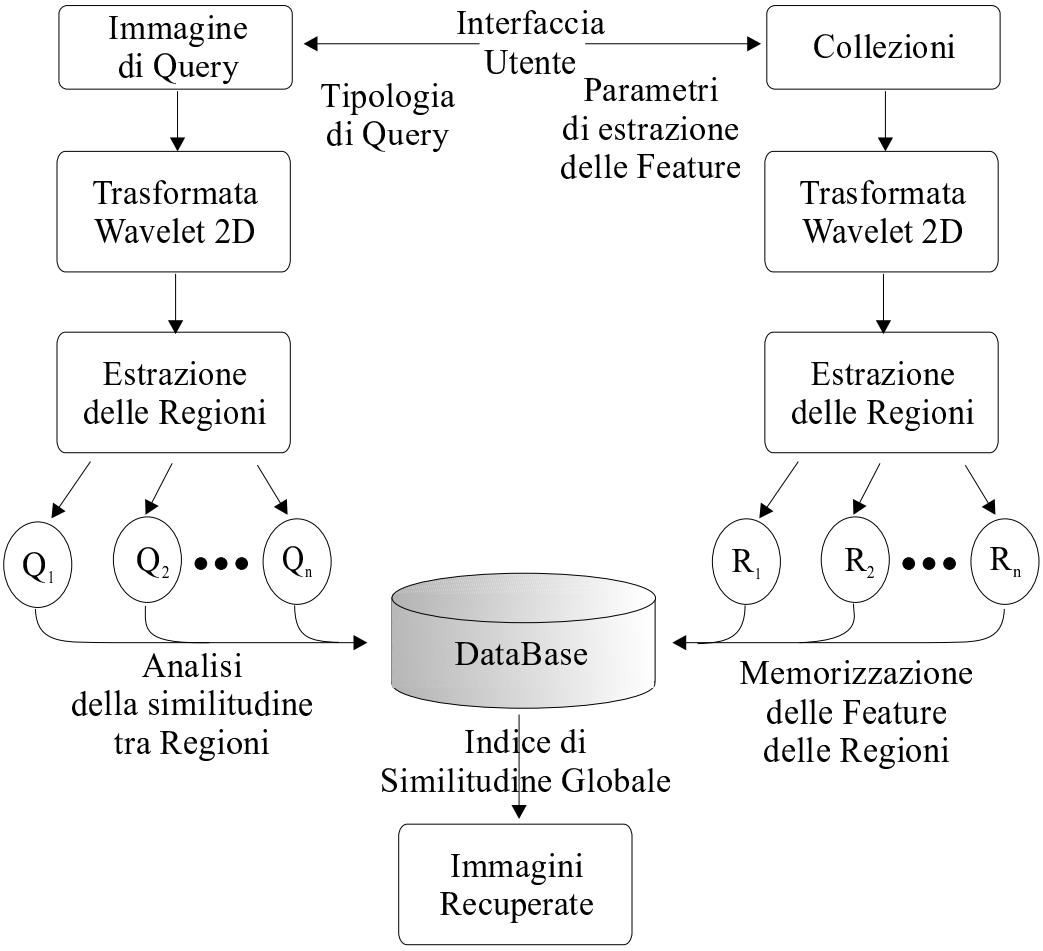


Figura 6.2: Architettura del motore di ricerca Windsurf.

calcolo delle “feature” corrispondenti. In particolare è possibile specificare il tipo di trasformata, il numero di passi di trasformazione, lo spazio di colore in cui sono rappresentati i pixel dell’immagine ed infine la metrica di distanza da utilizzare nell’algoritmo di clustering per l’estrazione delle regioni.

2. Recupero di immagini “simili” attraverso query di tipo visuale: con questa opzione l’utente deve per prima cosa aprire una collezione su cui siano già state compiute le operazione del punto 1., poi deve fornire l’immagine di query (query by example) in modo che vengano recuperate le K immagini più simili (K-nearest query). Oltre a fornire l’immagine di query, l’utente deve specificare il valore di K e se vuole applicare l’estrazione delle regioni

anche all'immagine di query.

In entrambe le fasi si fa uso di un'interfaccia grafica attraverso la quale l'utente specifica quale delle due operazioni deve essere eseguita e quali sono i parametri necessari all'esecuzione dell'operazione.

I risultati della query vengono presentate all'utente attraverso un'ulteriore interfaccia grafica dove le immagini sono visualizzate in versione miniaturizzata.

6.2.1 Estrazione delle feature.

L'architettura globale di **Windsurf**, che si occupa dell'estrazione delle feature delle regioni, è composta da tre moduli. Ogni modulo esegue una serie di elaborazioni che trasformano le immagini di ingresso in una rappresentazione finale sotto forma di feature le quali ne codificano le caratteristiche peculiari.

I tre moduli sono:

DWT: l'immagine d'ingresso è analizzata nel dominio tempo/frequenza.

Estrazione delle regioni: dall'immagine trasformata vengono estratte una serie di regioni omogenee rispetto ad un dato predicato.

Estrazione delle feature: dalle regioni così ottenute vengono calcolate e memorizzate una serie di feature caratteristiche.

Al primo passo l'immagine è elaborata attraverso la trasformata Wavelet, descritta nel Paragrafo 2.2, e il risultato di questa elaborazione restituisce i coefficienti della trasformata per ciascun canale di colore, oppure le energie e le cross–energy dei coefficienti della trasformata. In particolar modo le energie e le cross–energy forniscono informazioni sia sui singoli canali, che sulla correlazione tra coppie di canali distinti.

La scelta di una particolare grandezza come risultato dell'applicazione della trasformata influenza la realizzazione del passo successivo di estrazione delle regioni, in particolare l'algoritmo di clustering K–Means (Paragrafo 4.3.1) il quale si prefigge di minimizzare la distanza tra tutti i punti elaborati ed una serie di punti che giocano il ruolo di centri delle regioni.

La scelta dei valori su cui applicare la clusterizzazione influenza anche la scelta della funzione per la determinazione della distanza tra i punti delle regioni.

Prima di effettuare una scelta sul tipo di coefficiente da utilizzare, è necessario decidere quali sottobande, risultato della WT, devono essere elaborate nella clusterizzazione.

In particolare per un dato livello possiamo utilizzare: solo le alte frequenza $S \in \{LH, HL, HH\}$, solo le basse frequenze $S \in \{LL\}$, oppure sia le alte che

le basse ed anche una combinazione più complessa di sottobande di frequenza appartenenti a livelli di trasformazione diversi.

A fronte di analisi sperimentali si è verificato che in Windsurf i risultati migliori sul recupero di immagini si ottengono considerando per la clusterizzazione solamente le basse frequenze dell'ultimo livello di trasformazione.

$$x_j = e_j^{l;S} \tag{6.1}$$

Se si utilizzano le energie dei coefficienti della trasformata si lavora in uno spazio a 6 dimensioni e la distanza che si utilizza e quella Euclidea.

Come sappiamo dal capitolo sulle metriche, la distanza Euclidea è una distanza di tipo lineare dove la matrice A è la matrice identità:

$$D(x_j, \mu_i) = (x_j - \mu_i)^T \cdot (x_j - \mu_i) \tag{6.2}$$

Se si suppone, ad esempio, di essere in un caso bidimensionale questa formulazione ci consente di dire che i punti aventi distanza $D(x_j, \mu_i) \leq R$ dal centro μ_i sono contenuti nel disco di raggio R centrato in μ_i ed appartengono ad un unico cluster.

Se utilizziamo per il clustering le energie, i centri μ_i sono punti nello spazio 6D. I dati di input sono normalizzati prima di essere utilizzati dall'algoritmo di clustering.

La normalizzazione viene effettuata sulla base del primo e secondo momento:

$$\eta_1^0 = \frac{\sum_{j=1}^{NP} x_j}{NP} \quad \eta_2 = \frac{\sum_{j=1}^{NP} (x_j - \eta_1^0)^2}{NP} \tag{6.3}$$

che rappresentano la media e la varianza dei punti.

Pertanto i dati di input sono normalizzati utilizzando la seguente formula:

$$x_j^{Norm} = \frac{x_j - \eta_1^0}{\sqrt{\eta_2}} \tag{6.4}$$

Nel caso di immagini uniformi, poiché non esiste variabilità dei dati, il secondo momento è nullo ed i coefficienti normalizzati vengono posti pari a zero.

Il secondo approccio utilizzato considera i soli coefficienti della trasformata Wavelet, ovvero ogni punto j in ingresso all'algoritmo K–Means è definito nello spazio 3D come:

$$w_j = e_j^{l;S} \tag{6.5}$$

Le proprietà di correlazione fra punti in questo caso sono ottenute utilizzando nel K–Means la metrica di Mahalanobis che è sempre di tipo quadratico ed utilizza

come matrice A, l'inversa della matrice di covarianza, definita sui coefficienti nella quale le componenti omologhe rappresentano le energie e le componenti non omologhe le correlazioni.

Definita la media dei coefficienti del canale c, con $c \in \{0, 1, 2\}$, pari a:

$$m_c^{l;S} = \frac{1}{NP} \sum_{j=1}^{NP} w_{c,j}^{l:S} \tag{6.6}$$

con NP numero di punti appartenenti alla sottobanda S; indichiamo nel seguito con la notazione:

$$C^{l;S} = \left\{ cov_{c,d}^{l;S} | S \in \{LL, LH, HL, HH\} \wedge l \leq L \right\} \tag{6.7}$$

la matrice di covarianza calcolata sulla sottobanda S del livello l di trasformata Wavelet i cui coefficienti sono definiti come:

$$cov_{c,d}^{l;S} = \frac{1}{NP} \sum_{j=1}^{NP} w_{c,j}^{l:S} w_{d,j}^{l:S} - m_c^{l;S} m_d^{l;S} \quad c, d \in \{0, 1, 2\} \tag{6.8}$$

Si ha che per $c = d$ l'elemento $cov_{c,c}^{l;S}$ rappresenta l'energia del c-esimo canale, per $c \neq d$ $cov_{c,d}^{l;S}$ indica la correlazione fra il canale c e d: i canali possono infatti essere positivamente o negativamente correlati. Tale matrice è simmetrica poiché la correlazione tra c e il canale d è uguale alla correlazione tra d e c.

Sulla base della precedente definizione riformuliamo la definizione della distanza quadratica sostituendo ad A la matrice di covarianza ottenendo la distanza di Mahalanobis:

$$D(x_j, \mu_i)^2 = (x_j - \mu_i)^T \left(C^{l;S}\right)^{-1} (x_j - \mu_i) \tag{6.9}$$

dove con $\left(C^{l;S}\right)^{-1}$ si indica l'inversa della matrice di covarianza delle informazioni utilizzate per il clustering.

Dalla stessa definizione della matrice di covarianza si ha che i coefficienti sono in grado di fornire informazioni sulla forma dei cluster; pertanto questo tipo di approccio consente di individuare cluster con forma iperelissoidale. L'inversa della matrice fornisce la normalizzazione dei dati.

Si può notare come i due approcci risultino omogenei, infatti entrambi utilizzano informazioni di correlazione:

in un caso, interne alle feature stesse, nell'altro indirettamente per mezzo della matrice di covarianza; inoltre in entrambi i casi si ha la normalizzazione esplicita od implicita dei dati.

Durante la clusterizzazione bisogna porre in evidenza il fatto che non sempre è possibile mettere in evidenza regioni e quindi si otterrebbero risultati non significativi.

In questo caso si parla di immagini non clusterizzabili, cioè formate da un unico pattern, ossia percettivamente omogenee. Il parametro sulla base del quale si valuta se procedere o meno con la clusterizzazione è dato dalla traccia della matrice di covarianza calcolata sui punti della sottobanda LL utilizzata per il clustering.

La traccia è calcolata come la somma degli elementi della diagonale principale che è anche pari alla somma degli autovalori λ_c della matrice:

$$T_{C^{l;LL}} = \sum_{c=0}^{2} cov_{c,c}^{l;LL} = \sum_{c=0}^{2} \lambda_c \tag{6.10}$$

Dalla definizione stessa della matrice di covarianza si deduce come i suoi coefficienti siano in grado di fornire informazioni sulla variabilità dei dati.

Infatti la matrice di covarianza può essere rappresentata come un iperelissoide i cui assi principali sono dati dagli autovettori di $C^{l;LL}$, mentre gli autovalori ne determinano la lunghezza.

Una scarsa variabilità fra i coefficienti fa pensare alla presenza di un pattern omogeneo, un'alta variabilità indica la presenza di oggetti disomogenei gli uni con gli altri. Quindi se la traccia ha un valore basso l'immagine può essere considerata come un pattern unico e non viene suddivisa in regioni, se ha un valore alto si procede con l'algoritmo di clustering.

Il valore di soglia[1] con il quale è confrontata la traccia è stato ottenuto stimando l'omogeneità di un insieme di immagini campione.

Ogni regione ottenuta attraverso il K–Means deve essere identificata utilizzando un insieme di attributi che siano in grado di caratterizzarla in maniera univoca; tali attributi sono detti feature di similitudine e saranno utilizzati durante il processo di ricerca per il recupero di immagini percettivamente simili.

Windsurf utilizza come feature di similarità un insieme di informazioni che definiscono uno spazio 37D così costituito:

DIMENSIONE: la dimensione rappresenta il numero di punti che compongono la regione.

CENTRI: ogni regione è individuata per mezzo del centro di gravità rappresentato dal centro del cluster; poiché l'informazione a nostra disposizione è suddivisa in bande di frequenza, ogni regione viene individuata utilizzando le informazioni di ogni sottobanda del livello utilizzato per il clustering.

[1] In Windsurf pari a 10^3 nello spazio HSV

Perciò indichiamo i centri della regione Ri con la notazione:

$$V_{Ri} = (\mu_I^{LL}, \mu_I^{LH}, \mu_I^{HL}, \mu_I^{HH}) \tag{6.11}$$

dove μ_i^S è un punto nello spazio 3D e ogni componente di μ_i^S è il valore del centro rispetto ad ogni canale di colore. Pertanto V_{Ri} rappresenta il vettore dei centri delle quattro sottobande coinvolte. (Complessivamente 12D 4 bande di frequenza·3 canali di colore)

FEATURE: le feature "vere e proprie" sono i coefficienti delle matrici di covarianza calcolate sui punti che costituiscono la regione; analogamente a quanto visto per i centri, le matrici di covarianza sono calcolate su ogni sottobanda di frequenza.

Siccome la matrice di covarianza può essere vista come una matrice a blocchi, considerando solo i blocchi sulla diagonale principale che sono simmetrici, l'insieme dei coefficienti relativi alle quattro sottobande di frequenza costituisce un vettore 24D (4 bande di frequenza·6 elementi della matrice simmetrica di covarianza di dimensione 3 x 3).

Indichiamo quindi con C_{Ri} l'insieme delle quattro matrici di covarianza relative alla regione Ri e C_{Ri}^S per riferirsi alla matrice di una specifica sottobanda S.

6.2.2 Esecuzione delle query.

La seconda attività svolta dal sistema è l'esecuzione delle query che permettono di recuperare le K immagini più simili ad una fornita come esempio.

La similitudine tra immagini viene determinata in base alla similitudine delle rispettive regioni, definiamo $Q = (q_1, q_2, \ldots, q_n)$ l'immagine di query con n regioni e $T = (t_1, t_2, \ldots, t_m)$ una generica immagine appartenente ad un certo catalogo con evidenziate m regioni.

La similitudine tra due generiche regioni appartenenti a due immagini diverse q_i e t_j è definita attraverso la seguente funzione:

$$sim(q_i, t_j) = \alpha_{ij}\beta_{ij}\mathcal{C}(Dist_F(q_i, t_j)) \tag{6.12}$$

Il coefficiente α_{ij} viene detto coefficiente di similarità in dimensione relativa e prende in esame la dimensione relativa delle regioni rispetto a quella delle immagini a cui appartengono:

$$\alpha_{ij} = \frac{dim(q_i) + dim(q_j)}{dim(Q) + dim(T)} \tag{6.13}$$

dove $dim(\cdot)$ indica il numero di punti,cioè l'area, al livello di trasformazione considerato per l'immagine o la regione specificata come argomento. Tale coefficiente favorisce il match fra le regioni più grandi individuate all'interno delle immagini.

Il coefficiente β_{ij} esprime la similarità rispetto la dimensione assoluta, cioè confronta le regioni in funzione del numero dei punti che le compongono:

$$\beta_{ij} = \frac{|dim(q_i) - dim(q_j)|}{dim(q_i) + dim(q_j)} \tag{6.14}$$

Il valore di β_{ij} sarà perciò tanto maggiore quanto più le regioni sono formate da un numero simile di punti.

L'ultima parte della formula per la determinazione della similitudine tra regioni definisce la distanza tra le feature della regione. Windsurf utilizza la distanza di Bhattacharyya che viene usata per confrontare cluster di forma elissoidale:

$$Dist_F^2 = \frac{1}{2}\ln\left(\frac{\left|\frac{C_{q_i}+C_{t_j}}{2}\right|}{|C_{q_i}|^{\frac{1}{2}}|C_{t_j}|^{\frac{1}{2}}}\right) + \frac{1}{8}\left[(V_{q_i}-V_{t_j})^T\left(\frac{C_{q_i}+C_{t_j}}{2}\right)^{-1}(V_{q_i}-V_{t_j})\right] \tag{6.15}$$

dove la notazione $|\cdot|$ indica il determinante della matrice in esame.

Questa distanza confronta direttamente i vettori dei centri V e le matrici di covarianza C delle regioni di query e di target. La funzione $Dist_F$ migliora le prestazioni della distanza quadratica di Mahalanobis infatti, a meno di un fattore di scala, il secondo termine di dist è la distanza di Mahalanobis calcolata utilizzando la matrice di covarianza media delle due regioni.

Questa equazione è la somma di due componenti, la prima si basa unicamente sulle matrici di covarianza, la seconda include anche la differenza fra i centri. Questi due termini contengono, rispettivamente, informazioni sulla forma delle regioni e sulla posizione dei centri.

Questa formulazione migliora le prestazioni della metrica di Mahalanobis poiché il primo termine consente di non annullare il valore della distanza per quelle regioni che hanno i centri coincidenti.

Siccome ogni regione contiene informazioni relative alle diverse sottobande di frequenza, la metrica $Dist_F$ è in realtà meglio espressa come somma pesata calcolata sulle diverse sottobande di frequenza.

$$Dist_F^2 = \sum_{S=1}^{4} peso_S \cdot dist_S^2(q_i, t_j) \quad \text{con} \quad S \in \{LL, LH, HL, HH\} \tag{6.16}$$

La funzione $\mathcal{C}$ prende il nome di funzione di corrispondenza che traduce le distanze in similarità (Paragrafo 3.1) ed in questo caso è una funzione di tipo esponenziale.

Il passo successivo consiste nella determinazione della similitudine globale di due immagini a partire dai valore delle similitudini delle rispettive regioni. Windsurf determina la similitudine globale rispettando i seguenti criteri:

1. Una regione q_i non può essere simile contemporaneamente a due regioni distinte t_j, t_k;
2. Due regioni distinte q_j, q_k non possono essere simili ad una stessa regione t_i.

Per soddisfare questi requisiti si definiscono una coppia di funzioni.

In particolare la funzione g soddisfa soltanto il primo requisito:

$g : Q \to T$ con $Q = (q_1, q_2, \ldots, q_n)$ e $T = (t_1, t_2, \ldots, t_m)$ tale che $\forall q_i \in Q$

$$\begin{array}{c} g(q_i) = t_j \in T \\ \Leftrightarrow \\ \mathcal{C}(Dist_F(q_i, t_j)) = \max_{k \in \{1,\ldots,m\}} \{\mathcal{C}((Dist_F(q_i, t_j))|\forall t_k \in T\} \end{array} \tag{6.17}$$

In altre parole ogni regione q_i dell'immagine di query Q non può corrispondere con due regioni distinte di una stessa immagine T, ma è necessario scegliere quella più "simile" (cond. 1).

Sulla base di questa definizione e delle condizioni precedenti, S è definita come segue:

$$S(q_i) = \begin{cases} t_j & \Leftrightarrow \quad \begin{array}{c} g(q_i) = t_j \wedge \\ \exists q_{i'} \neq q_i : g(q_{i'}) = t_j \wedge \\ \mathcal{C}(Dist_F(q_{i'}, t_j)) > \mathcal{C}(Dist_F(q_i, t_j)) \end{array} \\ \text{non definita} & \quad \text{altrimenti} \end{cases} \tag{6.18}$$

La prima condizione di tale formulazione esprime l'iniettività della funzione S (cond. 2), viceversa la seconda condizione indica la parzialità della funzione di similarità stessa.

Definiamo pertanto la similarità complessiva fra coppie di immagini, $Q = (q_1, q_2, \ldots, q_n)$ e $T = (t_1, t_2, \ldots, t_m)$ come:

$$SIM(Q, T) = \sum_{i=1}^{n} sim(q_i, S(q_i)) = \sum_{i=1}^{n} sim(q_i, t_{j(i)}) \tag{6.19}$$

dove $j(i)$ è l'indice della regione T che presenta la miglior corrispondenza con $q_i \in Q$, ovvero $t_{j(i)} = S(q_i)$.

Si ha inoltre che $sim(q_i, S(q_i)) = 0$ se $S(q_i)$ non è definita. L'esecuzione di una query è un'operazione molto complessa perché richiede il confronto di un elevato numero di regioni:

se NR è il numero totale di regioni e n il numero di regioni in cui è scomposta l'immagine di query il calcolo della simitarità complessiva richiede $O(nNR)$ operazioni per la restituzione delle K immagini più simili.

6.3 WebSeek.

Webseek è un motore di ricerca, sviluppato presso la Columbia University, che fornisce strumenti per la ricerca di immagini e filmati sulla rete.

La chiave della versatilità di questo nuovo sistema risiede principalmente nella sua doppia natura, che fonde al suo interno sia l'elaborazione di informazioni testuali che l'analisi dei contenuti per catalogare e recuperare sia immagini che filmati.

Questo sistema interagisce con la rete collezionando informazioni multimediali tramite agenti automatici, ne estrae le feature nei domini testuali e visuali, le cataloga in un database e crea una serie di indici per velocizzare ed ottimizzare le ricerche secondo criteri definiti in maniera interattiva dall'utente.

6.3.1 Popolazione del Database.

La popolazione del database è condotta tramite una serie di agenti autonomi detti "spider". Questi agenti viaggiano all'interno della rete navigando attraverso i collegamenti ipertestuali passando da un documento HTML all'altro recuperando e catalogando all'interno del proprio database tutti i filmati e le immagini incontrate.

Il particolare il processo sopra delineato viene realizzato da tre agenti autonomi diversi, ognuno con compiti particolari:

Spider 1: assembla liste di pagine Web possibili candidate a contenere immagini, filmati o collegamenti ad esse (Figura 6.3);

Spider 2: estrae gli URL delle immagini e dei filmati contenute nelle pagine Web recuperate dello spider 1 (Figura 6.3);

Spider 3: recupera ed analizza, tramite gli URL estratti dallo spider 2, immagini e filmati(Figura 6.4).

Il processo di recupero prende il via da una serie di URL di partenza non ancora esplorati che sono passati all'ingresso dello spider 1. Questo esegue una prima navigazione sulla rete scaricando le pagine attraverso il protocollo HTTP e passando le pagine recuperate in formato HTML allo spider 2.

Lo spider 2 per prima cosa estrae gli URL di nuove pagine, incontrati durante l'esplorazione, e li inserisce nella lista degli indirizzi ancora da esplorare, poi esaminando gli URL li converte, se necessario, in indirizzi assoluti e assegna ciascun URL ad una delle seguenti categorie HTML, immagini o filmati, in base all'estensione codificata secondo lo standard MIME e li passa allo spider 3. Lo spider 3 si occupa di scaricare le immagini e filmati dalla rete utilizzando gli indirizzi forniti dallo spider 2 e di memorizzarle localmente nel database. Dopo che

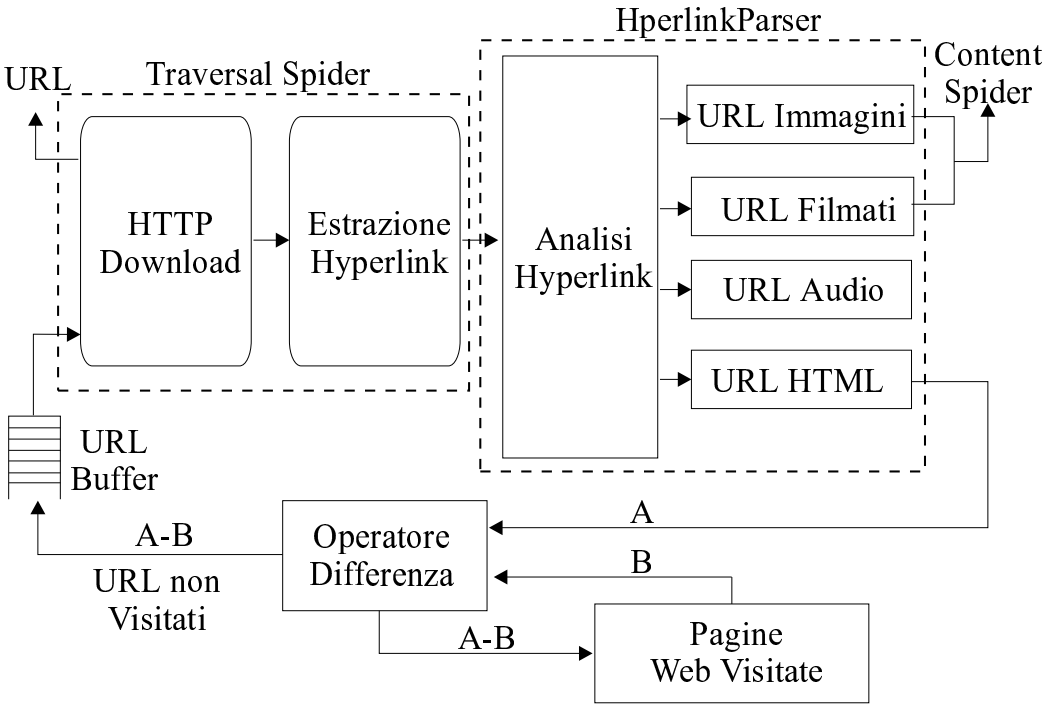


Figura 6.3: Architettura degli Spider 1 e 2 del motore di ricerca WebSeek.

il contenuto visuale è stato scaricato, diverse importanti operazioni sono eseguite su di esso:

- Determinazione del contenuto semantico delle immagine e dei filmati tramite l'analisi di parametri testuali;
- Estrazione e memorizzazione delle feature visuali che verranno impiegate per le ricerche basate sul contenuto non semantico;
- Estrazione e memorizzazione di altri attributi come le dimensioni per le immagini o il numero di fotogrammi per i filmati;
- Generazioni di versioni altamente miniaturizzate degli elementi originari per una più rapida visualizzazione dei risultati delle query.

Per le immagini la miniaturizzazione avviene spazialmente sotto campionando ed applicando formati di memorizzazione compressa dei dati come jpg o gif.

Per i video invece la miniaturizzazione avviene sia spazialmente con le modalità precedenti che temporalmente catturando un fotogramma ogni secondo ed eliminando eventuali duplicati tra fotogrammi adiacenti, rianimando il tutto, partendo dai fotogrammi superstiti, sotto forma di GIF animata.

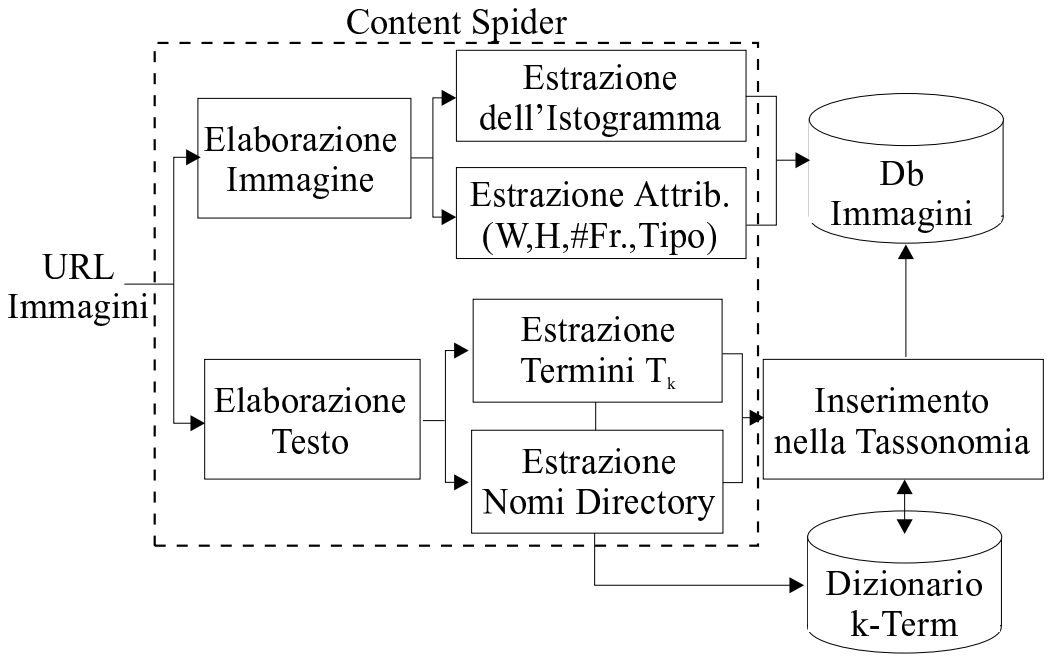


Figura 6.4: Architettura dello Spider 3 del motore di ricerca WebSeek.

Quando un video è recuperato come risultato di una query questo appare come una sequenza di campioni animati.

6.3.2 Estrazioni delle feature testuali.

Webseek organizza le immagini ed i filmati associandoli, in maniera semi automatica o manuale, ad una o più categorie facenti parte di una gerarchia di soggetti.

Il procedimento semi automatico per il quale immagini e filmati vengono inseriti in una serie di categorie sfrutta come punto di partenza lo studio degli indirizzi URL che puntano agli elementi considerati. Spesso il campo URL e ad altri tags HTML contengono già al loro interno un'efficace descrizione del contenuto semantico dell'immagine o del filmato a cui puntano, per cui il processo tramite il quale Webseek determina la categoria di inserimento non può non prescindere da queste informazioni.

Gli elementi visuali sono riferiti all'interno di un documento HTML in due modi: direttamente all'interno della pagina visualizzata correntemente `<ing src=URL alt=[alt text]>`, oppure indirettamente attraverso un collegamento ipertestuale `<href=URL> [hyperlink text]</a>`.

I termini significativi sono estratti tramite particolari funzioni dagli URL, dal testo del link o dal tag alt. Una funzione spezza la stringa, estraendo una serie di termini T_k , ogni volta che si incontra un carattere non alfabetico.

Tutti i termini T_k estratti vengono inseriti in una tabella associando a ciascuno di essi un contatore f_k che tiene traccia del numero di occorrenze. La lista viene ordinata in maniera crescente rispetto al contatore in modo da incontrare i termini più significativi all'inizio dell'analisi che da questo punto in poi procede in maniera manuale.

L'utente seleziona i termini manualmente creando una tabella di termini chiave T_k^* con le seguenti caratteristiche:

- non devono essere termini generici, ad esempio immagine, icona, o grafica sono termini non rappresentativi;
- non devono essere termini che causino casi di omonimia in modo che l'assegnazione delle immagini alle classi di soggetti non crei ambiguità ad esempio il termine "rock" può indicare una certa roccia oppure un cantante.

Ogni termine valido T_k^* viene inserito in una tabella detta dizionario dei termini chiave che associa ad ogni termine chiave un classe di soggetti C_x all'interno della gerarchia dei soggetti.

Una classe di soggetti C_x è un raggruppamento di immagini e filmati che hanno lo stesso contenuto semantico. Una tassonomia di soggetti è l'organizzazione in una gerarchia is–a di tutte le classi di soggetti.

L'insieme di soggetti che popolano il mondo viene diviso in una serie di categorie di primo livello ad esempio animali, arte, astronomia, intrattenimento, ecc... Ogni volta che un nuovo termine chiave viene inserito nel relativo dizionario, una nuova sottoclasse viene creata in maniera manuale nella tassonomia e la corrispondenza tra soggetto e termine chiave è registrata nel dizionario. Per ogni nuova immagine o filmato recuperato una serie di informazioni sono inserite in un una serie di tabelle (chiavi primarie in maiuscolo):

```
Immagini (IMID, url, nome, formato, altezza,
          larghezza, N#fotogrammi, data, testo);
Tipi     (IMID,TIPO);
Soggetti (IMID,SOGGETTO); Testo(IMID,TERM);
Feature (IMID,istogramma);
```

Dove TIPO, SOGGETTO e istogramma sono tipi di dato complessi definiti come:

```
TIPO in     [foto a colori, grafica a colori, video, immagine
             bianco e nero, immagine a toni di grigio];
SOGGETTO    classe di soggetti inclusa dalla tassonomia;
ISTOGRAMMA  istogramma nello spazio HSV
            discretizzato a 166 valori.
```

6.3.3 Estrazione delle feature visuali.

Webseek [**?**, **?**] fornisce strumenti per le ricerche basate sul contenuto per immagini e filmati che permettono di estrarre gli istogrammi della distribuzione dei colori.

L'istogramma risulta essere la feature più generale possibile rendendo così le ricerche indipendenti dal dominio applicativo. L'istogramma implementato in Webseek è una versione discretizzata a 166 elementi dello spazio HSV (Paragrafo 2.1).

Gli istogrammi vengono calcolati e memorizzati all'interno del database per essere recuperati e confrontati in fase di query. La funzione di distanza, che permette di calcolare in maniera automatica la similitudine tra due immagini, è la classica distanza quadratica normalizzata (Paragrafo 3.1).

Il calcolo dell'istogramma viene anche impiegato per la determinazione del tipo di immagine sfruttando l'analisi discriminante di Fisher.

Recentemente oltre al calcolo dell'istogramma è stato inserito un collegamento ad un altro motore ricerca basato sul contenuto, VisualSEEK sviluppato sempre presso la Columbia University, in grado di estrarre feature sulla distribuzione spaziale di zone di colore uniforme.

6.3.4 Esecuzione delle Query.

Per ricercare immagini e filmati con caratteristiche particolari l'utente può esprimere query con cui recuperare tutte le immagini con determinate caratteristiche.

Se la query coinvolge solo campi testuali può essere espressa tramite il classico linguaggio SQL, mentre se vengono coinvolte feature visuali è necessario specificare immagini di riferimento o modificare un istogramma, partendo da quello di un'immagine simile aumentando o diminuendo le occorrenze dei colori, ed applicando metriche di similitudine per recuperare immagini con una distribuzioni dei colori simile.

Dopo aver ricevuto una serie di risultati come risposta ad una certa query, l'utente può operare delle manipolazioni aggiungendo e rimuovendo degli elementi interagendo con i risultati di altre query.

Questi meccanismi permettono di esprimere query più raffinate in modo da estrarre solo le immagini rilevanti per una data ricerca. In particolare è possibile eseguire operazioni sui risultati di due query totalmente indipendenti per ottenere query complesse.

Si considerino le seguenti query:

```
A=Query(Soggetto="natura")
C=Query(term="tramonto")
```

è possibile applicare ai risultati di A e C le seguenti operazioni:

Unione B = Query (Soggetto = ”natura” or term = ”tramonto”);

Intersezione B = Query (Soggetto = ”natura” and term = ”tramonto”);

Sottrazione B = Query (Soggetto = ”natura” and not term = ”tramonto”);

Sostituzione B = A = Query (Soggetto = ”natura”);

Mantenimento B = C = Query(term = ”tramonto”);

L'utente può ulteriormente elaborare il risultato di B applicando un riordinamento in funzione della similitudine con una feature visuale, oppure decidere di esaminare l'intero database in funzione della similitudine con una data feature visuale [**?**].

I risultati delle query vengono visualizzati 15 per volta, ordinando secondo la maggiore similitudine, le miniature generate in fase di popolazione del database oppure in maniera semplificata visualizzando solamente i nomi dei file.

Un ulteriore strumento per perfezionare i risultati delle ricerche permette all'utente di esprimere un giudizio sulla rilevanza o non rilevanza su ogni immagine riportata nel risultato della query, oppure può chiedere di recuperare tutte le immagini più simili ad una di quelle nel risultato dell'ultima query effettuata.

Questo strumento prende il nome di “Relevance Feedback” e viene implementato attraverso l'interfaccia utente specificando per ogni immagine rilevanza, indifferenza e non rilevanza ai fini dei risultati della query.

Il nuovo istogramma è definito dal precedente sommando gli istogrammi definiti come rilevanti e sottraendo quelli definiti come non rilevanti e trascurando quelli definiti indifferenti.

6.4 VisualSEEK.

VisualSEEK è l'implementazione di un motore di ricerca per immagini e filmati basato sul contenuto che fornisce all'utente una suite di strumenti grafici molto evoluti per formulare query su vasti database multimediali.

L'utente dialoga con il server che contiene il database tramite un interfaccia visuale sul client specificando, tramite gli strumenti forniti, le caratteristiche salienti delle immagini da ricercare. Il server ritorna le immagini che più somigliano alla caratteristiche specificate dall'utente.

Inizialmente VisualSEEK implementa le feature necessarie per recuperare le immagini in base al colore ed alla distribuzione spaziale di regioni di colore.

L'interfaccia fornisce pure gli strumenti per associare altre proprietà delle regioni di colore come la tessitura e il movimento, ma le ricerche su questi tipi di feature non sono ancora state implementate.

Mentre gli strumenti precedenti erano implementati in HTML e avevano un'interfaccia con funzionalità limitate, che permetteva solamente la ricerca su un colore specificato dall'utente, VisualSEEK invece è realizzato in Java e fornisce una interfaccia evoluta per esprimere query complesse su più caratteristiche peculiari delle immagini, inoltre le prestazioni e l'accuratezza dei risultati sono notevolmente migliori dei precedenti sistemi.

L'obbiettivo principale di VisualSEEK è quello di creare uno strumento per la ricerca di immagini e filmati che sia facile da utilizzare e che fornisca potenza e flessibilità nell'espressione delle query.

Il sistema sfrutta per il recupero delle immagini i dati estratti tramite algoritmi di elaborazione automatica dell'immagine e non impiega parole chiave o descrizioni testuali immessi manualmente dell'utente.

6.4.1 Implementazione Software.

VisualSeek [**?**] implementa diversi obbiettivi su cui si è focalizzata la ricerca:

- estrazione automatica di regioni localizzate con le relative feature;
- rappresentazione efficiente delle feature;
- conservazione delle proprietà spaziali;
- indicizzazione e recupero veloce di immagini e filmati.

Le feature estratte da VisualSeek per ogni immagine inclusa nel database locale comprendono il colore, la tessitura e la distribuzione spaziale. Visualseek fornisce un'alternativa agli istogrammi di colore definendo i "color set" per la rappresentazione del colore di regioni all'interno di immagini (Paragrafo 2.1).

L'istogramma dei colori è ottenuto discretizzando lo spazio dei colori dell'immagine e contando quanti pixel per ciascun colore sono presenti nell'immagine. La definizione dei "color set" implica l'applicazione di una serie di operazioni preliminari per modificare ed ottimizzare la dinamica dell'immagine originaria.

Il primo passo consiste in un passaggio dallo spazio RGB, dove ciascun pixel è codificato tramite la terna di colori rosso, verde e blu, ad un nuovo spazio definito dalla terna HSV dove ciascun pixel è codificato in base a tre valori che codificano rispettivamente la tinta (Hue), la luminosità (value) e la saturazione (Saturation).

Questo spazio viene impiegato per la maggiore uniformità rispetto allo spazio RGB. La trasformazione tra i due spazi è non lineare, ma facilmente invertibile.

Il passo successivo consiste nella quantizzazione che discretizza i valori dello spazio tridimensionale HSV ed associa ad ogni punto nello spazio dell'immagine un elemento all'interno di un vettore di quantizzazione. Quindi, incrementando il valore della corrispondente locazione all'interno di un vettore di M elementi, l'istogramma viene determinato automaticamente.

Dopo la quantizzazione viene definito uno spazio binario M dimensionale dove a ciascun asse corrisponde un colore, un generico vettore in questo spazio consta di M elementi che possono assumere solo due valori 0 o 1 e definisce un "color set" che è una selezione di un certo numero di colori dello spazio discretizzato.

- Ogni colore nel "color set" deve essere visivamente distinguibile dagli altri
- il "color set" deve includere tutti i colori distinguibili.

La quantizzazione adottata soddisfa le condizioni sopracitate dividendo lo spazio delle tinte in 18 possibili intervalli e quelli della saturazione e della luminosità in tre intervalli ciascuno, questa quantizzazione permette di ottenere 18x3x3=162 colori a cui devono essere aggiunte anche 4 tonalità di grigio per cui M=166.

La retro proiezione (back projection) dei "color set" opera a valle delle operazioni di trasformazione e discretizzazione dello spazio dei colori dell'immagine e di un'operazione di filtraggio, tramite un filtro mediano. Quindi è il primo passo per l'estrazione di regioni colore all'interno dell'immagine (Figura 6.5). Operazioni analoghe si compiono per la caratterizzazione delle tessiture estraen-

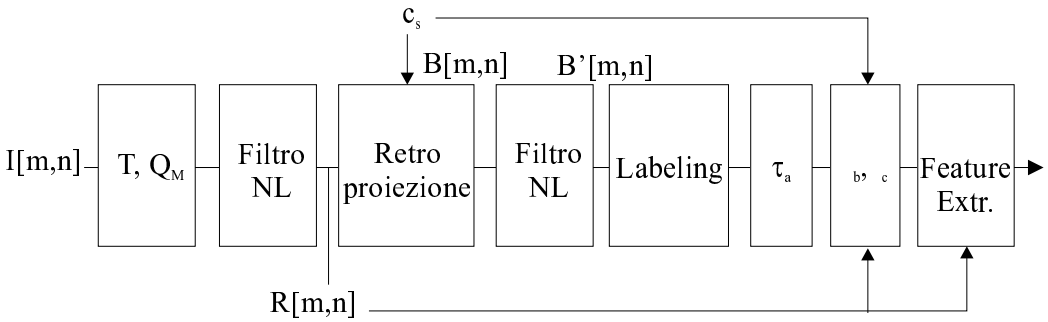


Figura 6.5: Passi necessari all'estrazione delle feature sul colore e sulla Tessitura in VisualSEEk.

do, con un procedimento simile al precedente, i "texture set" come descritto nel Paragrafo 2.2.

L'estrazione delle regioni avviene iterando il meccanismo di retro proiezione con tutti i possibili "color set" e "Texture set", ed è stata illustrata nel Paragrafo 4.3 riguardante le tipologie di algoritmi per l'estrazione di regioni.

Per ogni regione valida viene calcolato il rettangolo minimo che le contiene (MBR), il centro della regione, l'area in pixel e il "color set" sotto forma di vettori a 166 elementi.

Tutti questi dati vengono memorizzati in una tabella delle regioni che come chiavi contiene un identificatore dell'immagine e un identificatore della regione, poiché più regioni possono essere estratte per ciascuna immagine. Il passo successivo consiste nel definire le strategie di query che comprendono la creazione di indici e la definizione di funzioni di distanza.

Per le query sulle regioni di colore la prima cosa da definire è la similitudine tra due colori nello spazio HSV, che implica la definizione di una matrice A simmetrica contenente i coefficienti di similitudine per ogni coppia di colori (Paragrafo 3.1). La definizione di una misura per la distanza tra due generici "color set" è alla base del recupero di regioni aventi colore simile.

Data la relativa similitudine tra istogrammi e "color set" viene impiegata una versione modificata della distanza quadratica tra istogrammi di colori (Paragrafo 3.2)

Le operazioni di query possono essere notevolmente accelerate calcolando preventivamente μ_t e tutta la serie di $R_t[m]$ per $m \in \{0, 1, \ldots, M-1\}$ per ogni immagine inserita nel database e per ogni regione estratta attraverso la retro proiezione.

Viene definita una nuova tabella contenente questi valori avente come chiavi l'identificativo dell'immagine e della regione, indici secondari sono creati sui parametri calcolati μ_t e $R_t[m]$ per facilitare le ricerche su intervalli, in particolare i classici alberi binari.

L'immagine più simile a quella definita nella query è quella che all'interno al catalogo presenta il minimo valore della differenza:

$$\mu_t - 2 \sum_{\forall c_q[m]=1} r_t[m] \tag{6.20}$$

Per recuperare una serie di immagini basta fissare una soglia τ_x di massima distanza e recuperare tutte le immagini che rendono verificata la disequazione $dist \leq \tau_x$.

Come si può notare dalle formule precedenti il calcolo della distanza tra due immagini può essere vista come una serie di query sugli intervalli (range query) all'interno degli indici creati e quindi viene eseguita in maniera molto efficiente.

Per eseguire query sulla disposizione delle regioni di colore vengono considerati i seguenti parametri:

- Distanza euclidea tra i centri delle due regioni. Spesso durante una query non interessa l'esatta posizione spaziale, ma un certo range di posizioni che si ottiene disegnando come MBR della query un rettangolo più grosso di

quello minimo e fissando una distanza nulla se entrambe le regioni, pur avendo centri non sovrapposti, si trovano comunque all'interno dello stesso rettangolo;

- Differenza tra le aree delle due regioni in valore assoluto;

- Estensione spaziale degli MBR. Si utilizza la stessa formula della distanza euclidea sostituendo alle ascisse le larghezze e alle ordinate le lunghezze.

Per facilitare il recupero delle immagini in base alla disposizione spaziale vengono creati degli indici su grandezze multidimensionali come i punti bidimensionali che identificano i centri delle regioni e i rettangoli che identificano gli MBR.

Per i primi viene creato un indice basato su un albero Quad-Tree per i secondi un albero R-Tree (Paragrafi 5.2 e 5.4).

Quindi una query su una singola regione, su tutte le feature considerate, calcola la distanza globale delle diverse immagini eseguendo la sommatoria, secondo pesi stabiliti dall'utente, delle diverse distanze calcolate per ogni feature definita.

Per una query su più regioni, ogni regione definita viene elaborata singolarmente e il risultato finale viene calcolato come l'intersezione dei risultati delle query delle singole regioni.

Inoltre i risultati possono essere ordinati in funzione della disposizione relativa delle varie regioni.

6.4.2 Interfaccia Utente.

L'interfaccia utente di Visualseek è suddivisa in una serie di zone ciascuna con funzioni ben distinte [**?**].

Lo scopo principale di questa interfaccia è la composizione delle query visuali. Questa composizione coinvolge la selezione di colori e tessitura e l'assegnamento di proprietà spaziali come la forma, la posizione, le dimensioni e il movimento.

L'unità elementare di query impiegata in Visualseek prende il nome di POD (Point Of Denotation) che corrisponde ad una regione all'interno dell'immagine con qualche proprietà visuale degna di nota.

Il POD non deve essere identificato con una zona della stessa forma all'interno dell'immagine oppure con una regione prodotta da un algoritmo di segmentazione, ma indica solamente una zona che può possedere una serie di proprietà come il colore, la texture o la forma.

Per ogni query è possibile immettere fino a tre POD distinti. In altre parole l'utente può indicare fino a tre regioni all'interno dell'immagine, con le loro proprietà visuali, per ottenere le immagini più simili a quelle desiderate.

I POD, con tutte le proprietà definite dall'utente, vengono posizionati all'interno di una griglia bidimensionale detta pannello di posizionamento (LayoutGrid). All'interno del pannello di posizionamento oltre all'inserimento di nuovi POD è possibile attivare, ridimensionare, e riposizionare quelli già definiti.

Ogni POD è delimitato da un rettangolo; se durante la query una generica regione all'interno dell'immagine cade all'interno del rettangolo, questa e il POD sono considerate spazialmente coincidenti.

A fianco di questa griglia si trova il pannello delle proprietà (Propriety Panel) che indica per ogni POD quali proprietà definite sono attivate. Il pannello possiede una serie di pulsanti, uno per ogni proprietà, organizzati su tre colonne, una per ogni POD; se un pulsante è attivato la corrispondente proprietà è presa in considerazione nella query per il POD selezionato.

Le proprietà attivabili all'interno del pannello delle proprietà sono sette e riguardano il colore, la tessitura, la forma, il movimento, il testo, lo spazio e la visibilità. Le prime quatto specificano se la corrispondente proprietà debba essere considerata nella query, mentre text[2] abilita le stringhe di testo associate ai POD, lo spazio indica l'attivazione del rettangolo che circonda il POD come confine dello stesso, la visibilità include o esclude il POD dalla query.

Sotto al pannello di posizionamento è presente la zona dei bottoni (Button Area) in questa zona è possibile selezionare diverse modalità di ricerca, lanciare la query e resettare tutti i parametri immessi nell'interfaccia. In particolare si possono specificare le strategie di recupero, per ognuno dei tre POD, sia rispetto al colore che all'estensione nello spazio.

Le strategie definite sono "exact", "best match" e "none".

Per la disposizione spaziale specificando "exact" vengono recuperate tutte le immagini che contengono POD simili nella stessa posizione.

Con "best match" invece viene assegnato un punteggio in funzione delle caratteristiche di ogni POD specificato nella query, il punteggio è dato dalla somma delle funzioni distanza per ogni POD definito. La funzione distanza associa il valore 0 se il POD coincide esattamente con quello dell'immagine da recuperare, il valore 1 se il POD è solo parzialmente sovrapposto a quello dell'immagine ed il valore 2 se il POD è disgiunto da quello dell'immagine. Le immagini vengono ordinate e visualizzate come risultato della query secondo valori crescenti del punteggio.

Con "none" invece le informazioni sulle disposizioni spaziali vengono semplicemente ignorate.

Per il colore "exact" recupera immagini che contengono zone dello stesso colore dei POD definiti dall'utente. Con "best match" vengono ritornate sia

[2] Al momento non ancora implementa

quelle ritornate con "exact" che quelle contenenti colori simili secondo le metriche classiche sulle similitudini dei colori.

Con "none" le informazioni sul colore vengono semplicemente ignorate. Nella parte inferiore della schermata dell'interfaccia utente di Visualseek sono disponibili una serie di strumenti per specificare le proprietà dei POD:

- Il primo è la pila dei colori (ColorWell): una serie di ruote colorate sovrapposte che vengono visualizzate in successione agendo su due frecce che permettono di far scorrere verso l'alto o verso il basso, rispetto alla posizione corrente, tutta la serie di ruote dei colori.

  La pila dei colori è una rappresentazione discretizzata dello spazio dei colori HSV (Hue, Saturation, Value) e permette di definire il colore da associare al POD attivo nella griglia di posizionamento.

- Il secondo è il selettore delle tessiture (TextureSpread) contiene riproduzioni in formato ridotto di una serie di tessiture di esempio e permette di associarne una al POD attivo semplicemente con un doppio click sulla texture selezionata.

- Il terzo strumento (ShapeSketcher) permette di disegnare una forma arbitraria da associare ad uno dei tre possibili POD definibili.

- L'ultimo (MotionPad) permette di disegnare una linea retta che rappresenta il movimento del POD all'interno di un filmato.

Dopo aver espresso le proprietà che i risultati devono verificare non resta che premere il pulsante che manda in esecuzione la query ed attendere la visualizzazione dei risultati sotto forma di versioni miniaturizzate delle immagini originarie.

6.5 Metaseek.

Quando siamo interessati nella ricerca di immagini o filmati con particolari caratteristiche visuali si rende necessario interrogare un motore di ricerca basato sul contenuto.

Questi sistemi di solito forniscono metodi per recuperare immagini digitali utilizzando esempi oppure schizzi manuali. Per raggiungere lo scopo desiderato questi sistemi estraggono da ogni immagine archiviata una serie di feature tipo:

il colore, la tessitura o la forma in congiunzione con descrizioni testuali.

Lo scopo di queste operazioni è codificare in qualche maniera il contenuto visuale dell'immagine. Attualmente la proliferazione dei motori di ricerca ha sostituito il problema del recupero delle immagini con quello di scegliere il giusto

motore di ricerca, conoscendone pregi e difetti, e di come impiegarlo per condurre in porto le nostre ricerche.

Nasce quindi l'esigenza di integrare sotto una sola interfaccia motori di ricerca con caratteristiche, funzioni e particolarità diverse.

I motori di meta–ricerca sono punti di accesso che in maniera trasparente collegano l'utente a diversi motori di ricerca mettendoli in competizione tra loro allo scopo di estrarre il meglio da ognuno di essi.

Metaseek [**?**] è un motore di meta–ricerca, cioè un sistema utilizzato per il recupero di immagini e filmati, basato sull'analisi di feature visuali, che è stato progettato per selezionare ed interfacciare motori di ricerca con caratteristiche diverse presenti in rete.

Per ogni query Metaseek seleziona ed interroga diversi motori di ricerca soppesando i successi e gli insuccessi che si sono verificati in condizioni di ricerca simili. Il sistema tiene traccia e aggiorna costantemente le prestazioni dei vari sistemi integrando i giudizi degli utenti riguardo i risultati ottenuti.

6.5.1 Architettura interna.

La struttura interna di Metaseek è costituita da tre parti fondamentali (Figura 6.6):

Il traduttore delle query: (query translator), traduce la query immessa dall'utente nel linguaggio standard di metaseek in una serie di script compatibili con ogni motore di ricerca definito all'interno del sistema;

Il distributore delle query (query dispatcher), seleziona tra i motori di ricerca definiti quelli più adatti all'esecuzione della query;

L'interfaccia di visualizzazione: (display interface), fonde i risultati provenienti dai vari motori di ricerca, rimuovendo i duplicati e visualizza i risultati in maniera uniforme.

Dopo aver ricevuto un'interrogazione, il distributore delle query seleziona i motori di ricerca da interrogare consultando il database delle prestazioni (performance database) che risiede localmente a Metaseek.

Questo database contiene i punteggi sulle prestazioni in relazione ai successi e agli insuccessi di query passate per ogni modalità di ricerca (search option).

Poi il traduttore delle query traduce le query in adeguati script conformi alle interfacce dei vari motori di ricerca supportati ed infine l'interfaccia di visualizzazione fonde, ordina e presenta i risultati all'utente.

Metaseek valuta la qualità dei risultati per ogni modalità di ricerca prendendo in considerazione i giudizi espressi dell'utente. Queste informazioni vengono impiegate per modificare i valori contenuti nel database delle prestazioni.

Il criterio fondamentale impiegato nella progettazione di Metaseek è l'efficiente impiego delle risorse della rete, ciò implica che solo i motori che in passato hanno dato, in simili condizioni, i risultati migliori vengono interrogati e le immagini vengono scaricate dai database remoti dove risiedono solo quando sono effettivamente necessarie.

Metaseek correntemente supporta quattro motori di ricerca accessibili liberamente in rete:

- VisualSEEK;
- Webseek;
- QBIC;
- Virage.

ognuno con le sue caratteristiche individuali e con le sue limitazioni.

VisualSEEK, QBIC e Virage forniscono metodi per il recupero di immagini digitali sulla base di feature visuali. QBIC e VisualSEEK supportano ricerche personalizzate permettendo l'immissione di schizzi o immagini esterne. Virage implementa circa le stesse feature di QBIC, ma in più permette di assegnare un peso proporzionale all'importanza per ognuna delle feature definite. QBIC fornisce la possibilità di recuperare le immagini in base a parole chiave.

Webseek d'altro canto è un motore di ricerca che, in maniera semi automatica, cataloga immagini e filmati dalla rete, e supporta sia ricerche basate sul contenuto fornendo, feature sul colore, che ricerche basate su campi testuali [3].

Nella versione corrente di Metaseek l'interfaccia utente permette di visualizzare immagini casuali e di ricercarne di simili, esprimendo query basate sul contenuto oppure su parole chiave, da ognuno dei database supportati.

Le query vengono definite specificando un'immagine d'esempio, oppure immettendo un indirizzo URL che localizza un'immagine esterna. Dopo aver selezionato l'immagine d'esempio le sue feature vengono estratte e confrontate con quelle dei database esterni che Metaseek ritiene adeguati per recuperare le immagini più simili a quella d'esempio.

Le feature calcolate sono l'istogramma dei colori e la tessitura e possono essere ricercate una alla volta o insieme. Il blocco di distribuzione delle query è in grado di determinare quali motori di ricerca siano o no in grado di portare a termine le query con le opzioni di ricerca definite, per cui Metaseek può mandare in esecuzione più opzioni di ricerca sullo stesso motore e su più motori solamente attivando il rispettivo traduttore.

[3] In Metaseek Webseek viene interrogato solo per le ricerche su parole chiave

L'utente può definire il numero di opzioni di ricerca che possono essere mandate in esecuzione in parallelo in modo da adattarsi alla banda disponibile così da non sovraccaricare il collegamento. È possibile specificare un tempo massimo per l'esecuzione di ciascuna opzione di ricerca in modo da evitare situazioni di stallo del sistema nel caso che qualcuno dei motori interrogati non sia attivo al momento della query.

È possibile specificare anche una stringa di testo in modo da selezionare solo le immagini che contengono i termini specificati su cui operare le successive analisi sul contenuto. Dopo aver definito in tutti i suoi aspetti la query il sistema la passa al distributore che seleziona i motori di ricerca e le opzioni di query da interrogare avvalendosi per la loro esecuzione del traduttore.

Le decisioni vengono prese in base alle caratteristiche dei vari motori ed alle prestazioni calcolate mantenendo memoria delle query passate.

Ad esempio QBIC, Virage e VisualSEEK supportano la scelta di immagini casuali, QBIC e Webseek supportano le query con parole chiave, ecc. . .

Per le query sul contenuto la procedura per la selezione del motore di ricerca si basa sull'analisi del database delle prestazioni. Questo database contiene i risultati di come si è comportato un generico motore di ricerca per una generica opzione di query per tutte le query passate.

Un'opzione di query è una data metodologia di ricerca su un dato motore, ad esempio interrogare VisualSEEK sulle tessitura. Un'immagine d'esempio e un insieme di feature definiscono una query visuale; dopo aver ricevuto queste informazioni Metaseek cerca nel database delle prestazioni l'immagine d'esempio fornita e se questa esiste legge i dati sulle prestazioni passate e invia la query ai motori con i punteggi più alti.

Se una data immagine non è presente nel database delle prestazioni il sistema non compie scelte casuali, ma si cerca di legare le query nuove a quelle passate di cui si possiedono già una serie di informazioni.

Poiché le immagini impiegate nelle query vengono inserite nel database delle prestazioni e dato che viene applicato un raggruppamento in categorie (clustering) in base alla similitudine delle varie feature, allora Metaseek elabora l'immagine d'esempio, non ancora presente nel Database delle prestazioni estraendone le feature, poi determina le categorie con caratteristiche simili e visualizza un'immagine rappresentativa di ogni categoria in modo che l'utente possa scegliere quella più simile da cui prelevare i punteggi sulle prestazioni.

Dopo di che la nuova immagine viene inserita nel database delle prestazioni con i dati derivati da quelli dell'immagine rappresentativa della categoria che l'utente ha selezionato come più rilevante per i suoi scopi.

L'approccio utilizzato per gestire le categorie è il K-means per le sue caratteristiche di semplicità e ridotto calcolo computazionale.

L'algoritmo viene eseguito ogni 10 nuove immagini inserite nel database delle prestazioni. Per la feature sul colore viene estratto l'istogramma dei colori, per la tessitura invece viene utilizzato l'algoritmo di Tamura (Paragrafo 2.1) e la distanza tra due generiche feature è la classica distanza Euclidea (Paragrafo 3.1).

Il database delle prestazioni che contiene i valori delle feature e i punteggi sulle performance per ogni immagine impiegata per definire la query è realizzato in maniera gerarchica.

Al primo livello si trovano le categorie semantiche (semantic categorization) che corrispondono alle categorie selezionabili in fase di query (animali, arte, trasporti, ecc.).

Al secondo livello vi sono i raggruppamenti per feature (feature grouping) che comprendono tre categorie: colore, tessitura e tessitura&colore. Al terzo livello ci sono le categorie generate dall'algoritmo K–Means (clustering class) che raggruppa immagini con valori di feature simili. All'ultimo livello ci sono i dati sulle immagini (image query).

Per ogni immagine viene creata una tabella che contiene i seguenti dati:

- L'URL dell'immagine;
- Una serie di punteggi di performance:

  QBIC percentuali di colore
  QBIC disposizione dei colori
  QBIC texture
  VisualSEEK percentuali di colore
  VisualSEEK disposizione dei colori
  VisualSEEK texture
  Virage Colore
  Virage Composizione
  Virage texture

- Vettore delle feature sul colore (n-upla 166 numeri reali);
- Vettore delle feature sulla tessitura (tripla di numeri reali).

I punteggi sono numeri interi con segno:

numeri positivi equivalgono a buone prestazioni, numeri negativi cattive prestazioni, un valore NA indica un'opzione di query non disponibile per un dato motore.

Il database delle prestazioni viene aggiornato dinamicamente chiedendo agli utenti di esprimere le loro preferenze sui risultati ottenuti.

Le varie prestazioni per un'immagine vengono aggiornate ogni volta che l'utente utilizza un'immagine come esempio per una query. Se un'immagine recuperata da un dato motore di ricerca viene visitata dall'utente sul sito dove

risiede il punteggio nel database delle performance per l'immagine d'esempio viene incrementato di un'unità per il rispettivo motore.

Se l'utente esprime il suo gradimento per un'immagine recuperata il punteggio del motore che l'ha recuperata viene ulteriormente incrementato di un'unità, se non la gradisce il punteggio viene decrementato di un'unità.

Il problema fondamentale nella gestione del database delle prestazioni non è tanto l'aggiornamento dei dati quanto l'avviamento cioè la determinazione dei valore iniziali da assegnare alle prestazioni nel database.

Una soluzione consiste nell'eseguire un periodo di prova, prima che il sistema diventi totalmente operativo, in cui viene generato un set iniziale di valori per le prestazioni. Metaseek svolge questo periodo di avviamento con la costante presenza dell'utente che invia una serie di immagini d'esempio ai vari motori.

L'utente seleziona 40 immagini per ogni categoria semantica definita per essere usate come test di avviamento. Ogni immagine in questi set viene posta con tutte le opzioni di ricerca possibili su tutti i motori di ricerca e i relativi risultati vengono scaricati per ulteriori elaborazioni.

Per ogni immagine nel risultato della query che presenta una distanza rispetto a quella d'esempio minore di una data soglia viene incrementato di un'unità il punteggio del relativo motore di ricerca che l'ha recuperata e viene decrementato per tutte le immagini la cui distanza e maggiore della soglia.

La soglia viene determinata manualmente valutando la rilevanza dei risultati delle query. Quando la fase di test è terminata l'algoritmo di classificazione è applicato alle immagini di prova sia sul colore, che sulle tessitura, che su entrambi.

Il passo finale nell'esecuzione della query consiste nella visualizzazione dei risultati in cui Metaseek presenta all'utente la fusione dei migliori risultati dei singoli motori eliminando gli eventuali doppioni.

La strategia di visualizzazione dipende dal tipo di query effettuata. Nel caso di query su parole chiave i risultati provenienti dai vari motori vengono semplicemente mescolati, mentre nel caso di query basate sul contenuto la visualizzazioni dei risultati è più complessa.

Metaseek alterna i risultati delle query provenienti dai vari motori tenendo conto del peso che questi presentano all'interno del database delle prestazioni, per cui il numero di immagini visualizzate sarà proporzionale al punteggio assegnato al motore per quella query specifica [**?**].

Indirizzo HTTP dei sistemi analizzati	
Motore	Indirizzo HTTP
QBIC	http://wwwqbic.almaden.ibm.com
WindSurf	Non ancora disponibile in rete
WebSeek	http://www.ctr.columbia.edu/webseek
VisualSEEk	http://www.ctr.columbia.edu/VisualSEEk

Tabella 6.1: Indirizzi HTTP dei motori di ricerca disponibili su WWW.

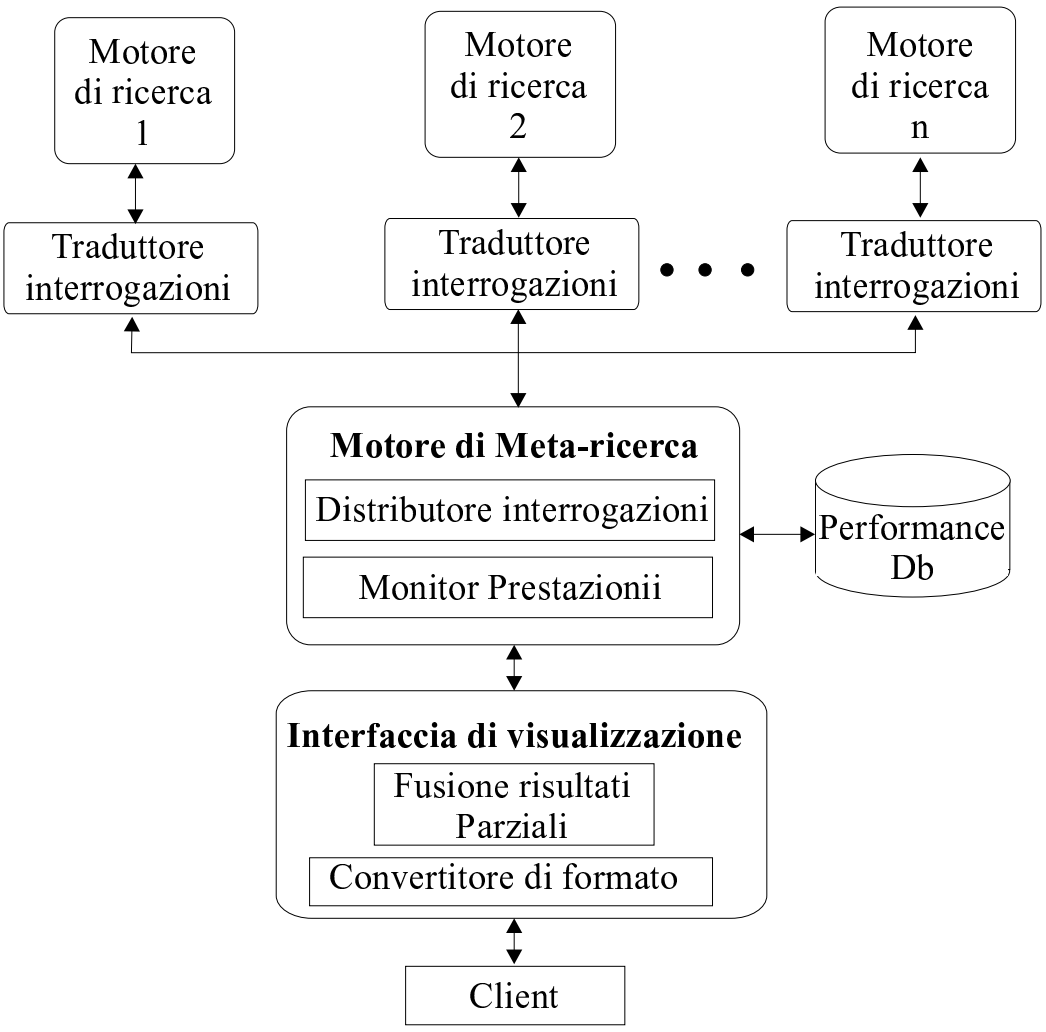


Figura 6.6: Architettura del motore di meta–ricerca Metaseek.

Capitolo 7

L’ambiente di sviluppo di QBIC.

QBIC, oltre a fornire una serie di programmi pronti all’uso per la gestione dei Database di Immagini e l’esecuzione delle interrogazioni basate sul contenuto (CBVQ), fornisce anche una serie di strumenti di sviluppo che permettono all’utente di integrare le funzionalità di recupero e gestione all’interno dei propri programmi.

In particolare vengono, forniti una serie di strumenti a diversi gradi di complessità, sviluppati in linguaggio C++ [**?**], che permettono all’utente di utilizzare gli oggetti e le classi all’interno dei propri programmi.

Utilizzando un approccio TOP-DOWN, al livello di maggiore complessità, si localizzano i programmi eseguibili a linea di comando in ambiente MsDos **QbMkDbs**, **QbQBE**, **QbMkThumb**, **QbDumpDb** che implementano le operazioni più complesse come la creazione dei cataloghi e le interrogazioni. Questi programmi vengono utilizzati assieme ad un Browser Internet, ad alcune applet Java, ed al linguaggio HTML per realizzare la Demo, fornita da IBM, che illustra le potenzialità di QBIC.

Ad un più basso livello invece, vengono fornite le librerie di classe C++ che definiscono la struttura degli oggetti ed il loro comportamento e che possono essere facilmente integrate nei programmi sviluppati degli utenti.

A livello di librerie di classe possiamo distinguere: la classe **QbicWrapClass** che fornisce una serie di metodi che combinano sottoinsiemi di classi di basso livello per compiere una serie di operazioni complesse, ma più frequenti ed una gerarchia di classi di basso livello per eseguire compiti particolari e personalizzazioni.

I programmi quindi conterranno per le operazioni di routine semplici chiamate ai metodi della classe QbicWrapClass, mentre per le operazioni particolari e specifiche del programma combinazioni più complesse dei metodi di basso livello.

7.1 I programmi eseguibili a linea di comando.

QBIC fornisce un insieme di programmi che realizzano le funzionalità di creazione dei cataloghi, interrogazione della base di dati, generazione delle miniature delle immagini e visualizzazione delle feature utilizzando per il passaggio dei parametri un'interfaccia a linea di comando.

I programmi forniti sono i seguenti:

QbMkDbs per la creazione dei cataloghi;

QbQBE per l'interrogazione dei cataloghi di QBIC;

QbMkThumb per la generazione delle miniature delle immagini originali;

QbDumpDb per la visualizzazione del contenuto delle feature che popolano il catalogo.

Tutti questi programmi utilizzano il DBMS dbm di UNIX che fissa le seguenti convenzioni:

1. Connettere il Database significa creare una directory con lo stesso nome del Database;

2. Creare un catalogo consiste nel generare un gruppo di file nella directory del punto 1. con l'estensione uguale al nome del catalogo.

Tutti i programmi si aspettano sempre una serie di parametri di ingresso, se questi non sono presenti oppure sono errati viene visualizzato un riepilogo della sintassi dei comandi così da poter determinare quella più corretta allo scopo prefissato. Illustriamo i parametri principali utilizzati dai vari programmi rimandando al riepilogo in linea per maggiori dettagli.

QbMkDbs oltre a creare un catalogo, operazione che consiste nel calcolare le feature che descrivono il contenuto delle immagini ed aggiungerle alle rispettive tabelle, fornisce funzioni per l'aggiornamento, l'inserimento e la cancellazione di ulteriori immagini nel Database. I parametri disponibili sono:

-d [Nome del Database]	nome del Db che contiene il catalogo all'interno del quale inserire le feature delle immagini.
-c [Nome del Catalogo]	nome del catalogo all'interno del quale inserire le feature delle immagini.
-f [Nome della feature]	nome della feature da estrarre e da inserire nel catalogo, possono essere più di una.
-i, -u, -U	definiscono la modalità di aggiornamento dei dati precedentemente inserti (-i=Incrementale aggiunge le feature mancanti ad un dato oggetto, -u aggiorna tutte le feature anche quelle esistenti, -U aggiorna tutte le feature anche se i dati non esistono).
-l[Nome File]	permette di specificare un file contenete una lista di immagini da elaborare (una per riga).

QbQBE viene impiegato per interrogare il Database di QBIC e per ottenere altre informazioni sul suo comportamento. Può essere utilizzato per interrogare il database e recuperare le immagini più simili ad una contenuta esternamente od internamente al Db ed identificata dal nome del file o dalla chiave rispettivamente.

Oltre a queste informazioni permette di visualizzare le feature disponibili nella versione corrente del programma e nel catalogo aperto. I parametri principali sono:

-d[..] -c[..]	come nel caso precedente.
-f [Nome della feature]	come nel caso precedente più una serie di parametri separati dai due punti che specificano con W=numero reale tra 0 e 1 il peso della feature in una query Multifeature e con O=0,1 il tipo di ordinamento in una query di tipo Multipass.
-i[Chiave], -r[Nome File]	specificano l'immagine da usare come esempio nell'interrogazione nel primo caso interna al Db ed identificata dalla relativa chiave e nel secondo esterna al db ed identificata dal nome del file.
-l e -L	visualizza l'elenco delle feature contenute nel catalogo e quelle complessivamente disponibili rispettivamente.

QbMkThumb estrae la miniatura di un'immagine e richiede due nomi di file il primo è il nome dell'immagine di ingresso, il secondo il nome dell'immagine d'uscita. Inoltre possono essere inclusi i seguenti parametri:

-l [Nome File]	come nel caso di QbMkDBS specifica una lista di immagini da “miniaturizzare”.
-o[Nome Directory]	specifica una directory diversa da quella di Default per la miniatura.
-s[Suffisso]	cambia il suffisso delle miniature.
-t	genera miniature in “True Color”.

QbDumpDb visualizza il contenuto delle tipologie di feature specificate, per un dato catalogo e per un dato Db, sullo schermo in modalità ASCII.

I parametri utilizzabili (-d, -c, -f) con lo stesso significato di quelli in QbMkDbs o QbMkQBE.

7.2 La classe di alto livello QbWrapClass.

La classe QbWrapClass di alto livello, singola, che non appartiene alla gerarchia delle classi di basso livello fornisce una serie di metodi che permettono l’esecuzione di operazioni per la popolazione e l’interrogazione del Database di QBIC.

Questa classe permette di eseguire in maniera semplificata operazioni complesse che richiederebbero l’utilizzo di molti metodi appartenenti alla gerarchia di classi di più basso livello.

In particolare questa classe contiene anche una serie di metodi che forniscono le stesse funzionalità e la stessa l’interfaccia dei programmi a linea di comando precedentemente descritti. I metodi della classe possono essere raggruppati nelle seguenti categorie:

- **Metodi sulle feature:**

  1. **Metodi per il calcolo delle feature:** GetFeatureDataImage, GetFeatureDataKey, GetFeatureDataPicker, GetFeatureDataString;

     Permettono di calcolare le feature delle immagini identificate in maniere diverse.

  2. **Gestione dei parametri delle feature:** GetParameters, GetParameterForFeature;

     permettono di visualizzare i parametri impostati per le feature;

  3. **Gestioni delle feature nei cataloghi:** InsertFeature, DeleteFeaure, ListAllFeature, ListCatFeature;

     permettono di inserire, eliminare ed elencare le categorie di feature utilizzate nell’interrogazione e nella popolazione del Database.

- **Metodi per l’interrogazione dei cataloghi del Database:**

1. **Definizione degli input della query:** QbicWrapQueryImage, QbicWrapQueryKey, QbicWrapQueryPicker, QbicWrapQueryString.

   Permettono di definire l'immagine d'esempio necessaria per l'esecuzione dell'interrogazione.

2. **Definizione degli output della query:** QbicWrapSetReturnedKeys;

   Permette di definire il numero di immagini contenute nel risultato della query.

3. **Esecuzione della Query:** QbicWrapQueryDB, QbicWrapRandomQueryDB;

   Permettono di eseguire una query, con i parametri specificati attraverso i metodi precedenti, oppure di ottenere un insieme casuale di risultati.

- **Metodi per la popolazione dei cataloghi del Database:**

  1. **Gestione delle immagini nei cataloghi:** QbicWrapInsertImage, QbicWrapInsertSubImage, QbicWrapDeleteImage, QbicWrapDeleteSubImag;

     Permettono di inserire od eliminare, immagini o oggetti estratti dall'immagine in maniera semiautomatica, per la gestione della popolazione del catalogo.

  2. **Gestione delle parole chiave associate alle immagini nel catalogo:** QbicWrapGetKeyWord, QbicWrapSetKeyWord.

     Permettono di leggere e scrivere all'interno del catalogo le parole chiave associate alle immagini.

- **Metodi per la gestione delle miniature delle immagini:**

  1. **Gestione dei parametri associati all'estrazione delle miniature:** QbicWrapSetPad, QbicWrapSetThumb24ColorMethod, QbicWrapThumbXY;

     Permettono di gestire le opzioni riguardanti la generazione delle versioni miniaturizzate delle immagini.

  2. **Estrazione delle minature:** QbicWrapThum.

     Permette di estrarre la miniatura dall'immagine originale.

- **Metodi per l'implementazione dei programmi a linea di comando:**

  1. **Definizione della modalita di visualizzazione:** SetScreenPrint.

     Permette di visualizzare i risultati dei programmi a linea di comando sullo schermo.

2. **Esecuzione dei programmi a linea di comando:** QbicWrapDBS, QbicWrapQBE, QbicWrapDumpDB, QbicWrapThm;

   Permettono di implementare le funzioni degli equivalenti programmi a linea di comando.

Approfondiamo il comportamento dei metodi introdotti descrivendone nei particolari la funzione svolta illustrando i parametri di ingresso/uscita necessari al loro funzionamento. Tutti i metodi della classe QbicWrapClass che alla loro invocazione restituiscono un numero intero, ritornano zero in caso di successo e meno uno in caso d'errore.

Metodi per il calcolo delle feature.

```
int GetFeatureDataImage(char *imgName, char
*maskName, QbStringClass *str);
```

Estrae le feature dell'immagine esterna al Database identificata dal nome del file memorizzato nella locazione di memoria puntata da *imgName.

```
int GetFeatureDataKey(char *imgKey, QbStringClass
*str);
```

Estrae le feature dell'immagine contenuta nel Database identificata dal nome della chiave memorizzato nella locazione di memoria puntata da *keyName.

```
int GetFeatureDataPicker(char *imgName, char
*maskName, QbStringClass *str);
```

Con picker si intende un'immagine disegnata dall'utente contenente una serie di forme di colore uniforme. In questo caso vengono considerate per l'estrazione delle feature solamente le parti effettivamente disegnate in modo che l'utente possa specificare solo i particolari interessanti senza dover necessariamente disegnare tutta l'immagine.

Il Picker è memorizzato nel file esterno avente nome memorizzato nella locazione di memoria puntata da *imgName.

```
int GetFeatureDataString(char *strDes,
QbStringClass *str);
```

strae le feature dell'immagine contenuta nel Database identificata da una serie di parole chiave memorizzate nella locazione di memoria puntata da *keyName.

Altri parametri comuni a tutti i metodi:

`*maskname:` punta ad una locazione di memoria che memorizza il nome di un file contenente la maschera, Cioè una matrice binaria della stessa dimensione dell'immagine dove gli elementi a valore nullo indicano la parte di immagine da ignorare, mentre la parte della maschera con valori diversi da zero (non necessariamente ad uno) rappresenta un oggetto su cui vengono calcolate le feature.

`*str:` puntatore ad un oggetto di tipo QbStringClass, estensione del tipo

base stringa, che contiene i valori delle feature estratte dal metodo dall'immagine o dall'oggetto delimitato dalla maschera.

Metodi per la gestione dei parametri delle feature:

```
int GetParameters(QbStringClass *str);
```

Estrae i parametri caratteristici delle feature implementate per l'oggetto su cui il metodo viene invocato. Il metodo restituisce i parametri delle feature alla locazione puntata da str.

```
QbParameterClass *GetParameterForFeature(char
*featureName);
```

Analogamente al precedente vengono estratti i parametri dalla feature con nome memorizzato all'indirizzo puntato da *featureName. I parametri dell'oggetto su cui è invocato il metodo vengono restituiti in un oggetto appartenete alla classe **QbParameterClass**.

Gestione delle feature nei cataloghi:

```
int InsertFeature(char *featureName);
int DeleteFeaure(char *featureName);
```

permettono di inserire/eliminare la feature, con nome puntato da *featureName, dall'oggetto su cui è invocato il metodo, ai fini dell'esecuzione delle interrogazioni o per la popolazione del Database.

```
char *ListAllFeature(void);
char *ListCatFeature(void);
```

restituiscono un puntatore ad una locazione di memoria contenete una lista di tutte le feature disponibili nella versione di QBIC utilizzata, e tutte le feature inserite nel catalogo con la funzione precedente rispettivamente.

Definizione degli input della query:

```
int QbicWrapQueryImage(char *imgName, char
*maskName);
```

assegna le specifiche di un'interrogazione utilizzando un'immagine esterna al Database il cui nome è memorizzato alla locazione di memoria puntata da imgName e da ed una maschera opzionale. In questo caso, prima di eseguire l'interrogazione bisogna estrarre le relative feature.

```
int QbicWrapQueryKey(char *keyName);
```

assegna le specifiche di un'interrogazione utilizzando un'immagine già contenuta del Database identificata dalla chiave memorizzata nella locazione di memoria puntata da *keyName. In questo caso le feture sono già state estratte, bisogna solo recuperarle dal Db.

```
int QbicWrapQueryPicker(char *keyName, char
*maskName);
```

assegna le specifiche di un'interrogazione utilizzando un Picker contenuto in un file esterno al Database il cui nome è memorizzato alla locazione di memoria puntata da imgName e da ed una maschera opzionale.

```
int QbicWrapQueryString(char *str);
```

assegna le specifiche di un’interrogazione utilizzando la descrizione di un Picker sotto forma di stringa di testo memorizzata nella locazione di memoria puntata da *str.

Metodi per la definizione degli output della query:

```
int QbicWrapSetReturnedKeys(int nhits);
```

permette di definire il numero di immagini, pari ad nhits, contenute nel risultato della query.

Metodi per esecuzione della Query:

```
int QbicWrapQueryDB(QbStringClass *str);
```

esegue l’interrogazione con i parametri impostati tramite i metodi precedentemente illustrati.

```
int QbicWrapRandomQueryDB(QbStringClass *str,
QbBoolean bool);
```

Esegue l’interrogazione recuperando un set di immagini del Database in maniera casuale. Il parametro bool permette di indicare se si vogliono ottenere, nelle successive invocazioni del metodo, set di immagini sempre diversi dai precedenti, il valore di default di bool è TRUE.

I risultati, per entrambe le modalità di interrogazione, sotto forma di stringa di caratteri sono memorizzate alla locazione di memoria puntata da *str.

Metodi per la gestione delle immagini nei cataloghi:

```
int QbicWrapInsertImage(char *imgName);
int QbicWrapDeleteImage(char *imgName);
```

permettono di inserire/eliminare le feature di un’immagine il cui nome è contenuto nella locazione di memoria puntata da *imgName, definite con i metodi precedenti(InsertFeature), dal catalogo associato all’oggetto su cui è invocato il metodo.

```
int QbicWrapInsertSubImage(char *imgNames);
int QbicWrapDeleteSubImag(char *imgNames);
```

Analoghi alla coppia precedente solo che permettono di inserire/eliminare le feature associate una serie di oggetti derivati da una serie di maschere appartenenti ad un’immagine “padre”. *ImgNames è un puntatore ad una stringa di caratteri che contiene l’immagine padre e delle maschere contenenti gli oggetti nel formato:

ImmaginePadre (Maschera1, Maschera2, ..., MascheraN)

Gestione delle parole chiave associate alle immagini nel catalogo:

```
char *QbicWrapGetKeyWord(void);
void QbicWrapSetKeyWord(char *kw);
```

permettono rispettivamente di leggere/scrivere le parole chiave associate alle immagini e memorizzate all’interno del Database. I parametri dei due metodi

rappresentano gli indirizzi di stringhe di caratteri che contengono parole chiave separate da spazi.

Metodi per la gestione dei parametri associati all'estrazione delle miniature.

```
Void QbicWrapSetPad(QbBoolean pad);
```

Setta in funzione della variabile boolean pad il riempimento delle zone laterali della miniatura se l'immagine da cui deriva non ha un rapporto tra le dimensioni del tipo $1:1$

```
void QbicWrapSetThumb24ColorMetod(QbBoolean Bool);
```

Permette di generare le miniature in "True Color" in funzione della variabile boolean bool.

```
Void QbicWrapThumbXY(int width, int height);
```

Setta le dimensioni della miniatura, dove width e height rappresentano il numero di pixel nelle direzioni delle ascisse e delle ordinate rispettivamente. Il valore di default è 100×100.

Metodi per la creazione delle miniature.

```
Int QbicWrapThumb(char *inImage, char *outImage);
```

I due parametri rappresentano rispettivamente i puntatori alle locazioni di memoria contenenti il nome del file dell'immagine originale e della relativa miniatura.

Metodi per l'esecuzione dei programmi a linea di comando:

```
  int QbicWrapDBS(int argc, char **argv,
QbStringClass *str=NULL);
  int QbicWrapQBE(int argc, char **argv,
QbStringClass *str=NULL);
  int QbicWrapDumpDB(int argc, char **argv,
QbStringClass *str=NULL);
  int QbicWrapThm(int argc, char **argv,
QbStringClass *str=NULL);
```

Questi metodi implementano le stesse funzioni dei rispettivi programmi MS-DOS in particolare utilizzano per la definizione dei parametri la stessa modalità a linea di comando.

I parametri di ingresso sono gli stessi della funzione main(int argc, char **argv) basta quindi includere all'interno della funzione main una chiamata a questo metodo su un oggetto della classe QbicWrapClass e leggere al termine dell'esecuzione i risultati nella stringa puntata da str.

Tramite il metodo

```
void SetScreenPrint(int sp);
```

è possibile specificare la visualizzazione sullo schermo $sp = 1$ oppure no $sp = 0$.

7.3 Gerarchia di classi di basso livello.

Al livello più basso QBIC fornisce una serie di classi C++ astratte ed alcune totalmente implementate per permettere operazioni come: la connessione ad un Database, la gestione dei cataloghi, il calcolo delle feature e così via.

Le classi astratte in C++ sono classi che utilizzano funzioni virtuali pure per definire l'interfaccia dei metodi costringendo le classi derivate a definire l'implementazione. I programmi non possono creare istanze delle classi astratte, solamente i puntatori alle classi astratte sono permessi. Queste classi sono state pro-

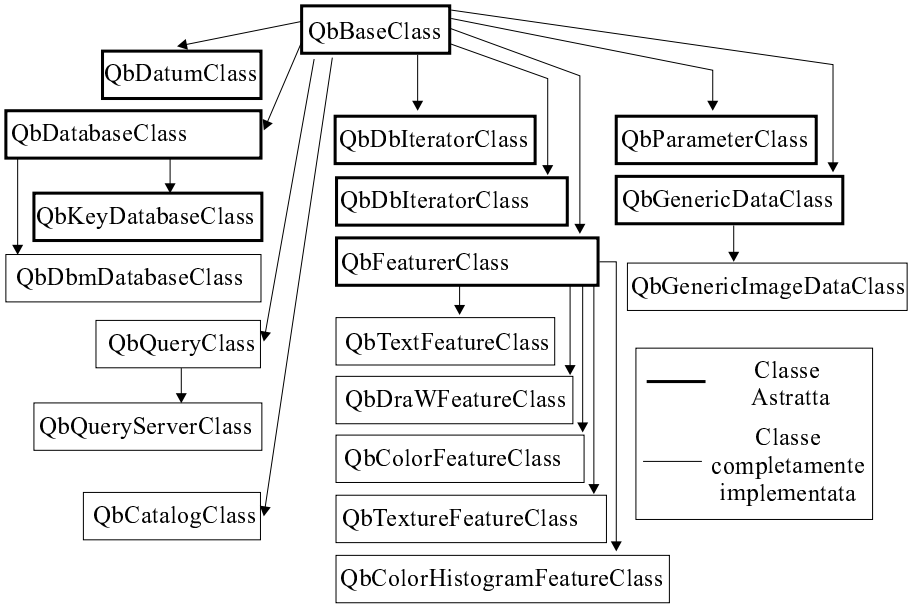


Figura 7.1: Schematizzazione della gerarchia di classi di basso livello implementata dalle API di QBIC.

gettate per essere estensibili, in particolare è possibile inserire classi per integrare nuovi tipi di contenuti multimediali come i suoni od i filmati oppure è possibile implementare nuove tipologie di feature, nuove metriche di distanza e così via.

La gerarchia di classi definita da QBIC utilizza solamente l'ereditarietà singola (Figura 7.1) in particolare esiste una convenzione sui nomi delle classi del tipo QbFraseDescrittivaClass. Le funzioni membro sono hanno tutte le iniziali delle parole maiuscole, mentre le variabili hanno l'iniziale della prima parola minuscola e la altre maiuscole.

Descriviamo ora le caratteristiche e le funzioni principali di ciascuna delle classi contenute nella API di QBIC rimandando al manuale utente [**?**] i particolari sui singoli metodi.

QbBaseClass è una classe astratta madre di tutte le altre classi nella gerarchia da cui derivano tutti i metodi definiti. Le classi derivate possono utilizzare i metodi della superclasse per determinare il tipo di interfaccia e monitorare lo stato degli oggetti.

Il tipo di interfaccia è una variabile enumerazione che può assumere i seguenti valori: genericData, feature, connect, database, keyDatabase, databaseIterator, parameter, catalog, query, ecc. . . , mentre lo stato degli oggetti e controllato dal metodo IsOk.

QbCatalogClass è una classe derivata da QbBaseClass che gestisce tutte le operazioni relative ai cataloghi come la creazione, la cancellazione di tabelle e l'aggiunta, la rimozione e l'aggiornamento delle feature per le immagini di input.

In questa versione di QBIC si utilizzano i files dbm di UNIX, come sistema di memorizzazione permanente, dove il nome del database è un sotto direttorio nel file system e i vari cataloghi sono i file con nome derivato dalla relativa classe di feature e con estensione uguale al nome del catalogo.

Per ogni catalogo QBIC genera un file di indice (FeatIdx.*nomeCat*) che permette di recuperare più velocemente le feature.

QbConnectClass è derivata dalla classe padre QbBaseClass ed è utilizzata per ottenere la connessione ad un Database o ad un Filesystem. Il Metodo Connect stabilisce una connessione al Database od esegue il Mount del Filesytem che sono operazioni dispendiose in termini di tempo e vengono eseguite una volta per tutte all'inizio dei programmi.

Il metodo Disconnect esegue il processo inverso. Entrambi i metodi richiedono il nome del database e la modalità si apertura in lettura "r" ed in scrittura "w".

QbDatabaseClass deriva dalla classe QbBaseClass e definisce un'interfaccia astratta ad un Database per archiviare i dati.

L'utente deve creare una specifica implementazione delle interfaccie delle funzioni virtuali come Open, Close, Insert, Update, Commit, ecc$ldots$ in modo da poter dialogare correttamente con i diversi Database Manager (IBM Db2, Oracle, Sysbase, dbm e così via).

In questa versione è implementata la classe **QbDbmDatabaseClass** che deriva dalla precedente e definisce il contenuto delle funzioni virtuali in modo da operare correttamente con il Database Dbm di UNIX. In particolare i metodi open e close aprono o distruggono una connessione al Db, la creano se non esiste e richiedono la definizione della modalità di apertura in lettura o scrittura.

Le operazioni che riguardano l'inserimento, l'aggiornamento o la cancellazione di record lavorano su coppie chiave(nome del file in Dbm) valore (ad esempio una feature); non sono ammessi due record con lo stesso valore di chiave.

QbDatumClass deriva dalla classe base e definisce un puntatore ad una stringa di byte, la sua lunghezza ed il formato del dato memorizzato. Questa

classe è utilizzata come "mezzo di trasporto" per trasferire i risultati dell'elaborazione di un metodo ad un altro e fornisce le operazioni di base per scrivere e leggere i dati privati della classe.

QbDbIteratorClass deriva dalla classe base e definisce quello che nei Database viene chiamato cursore (iteretor) che è necessario nella gestione liste di dati. Ad esempio nel caso della query si ottiene quasi sempre come risultato una lista di chiavi.

Per poter eseguire una serie di operazioni su ognuna di esse si crea un oggetto della classe QbDbIterator che punta al primo elemento della lista ed implementa i metodi: More che informa sull'esistenza di ulteriori elementi nella lista, Next che fornisce l'elemento successivo nella lista e Reset che riporta il cursore all'inizio della lista.

QbFeatureClass è una sotto classe della superclasse QbBaseClass e viene impiegata per incapsulare i valori delle feature. Implementa i metodi per l'estrazione delle feature e per il calcolo della distanza tra coppie di feature. Al fine di gestire le feature specifiche come l'istogramma dei colori oppure le tessiture vengono create le seguenti sottoclassi:

- QbTextFeatureClass per il recupero in base alle stringhe di testo;
- QbDrawFeatureClass per il recupero in base alla distribuzione delle zone di colore;
- QbColorFeatureClass per il recupero in base al colore medio;
- QbTextureFeatureClass per il recupero in base alla caratterizzazione delle tessiture;
- QbColorHistogramFeatureClass per il recupero in base all'istogramma dei colori.

QbGenericDataClass deriva dalla classe QbBaseClass e definisce tipologie di dati generiche come: immagini, video, testo e audio. Questa classe incapsuala i data nei passaggi tra il procedimento di estrazione delle feature, l'interfaccia utente e i file di dati esterni.

Questa classe è completamente implementata e fornisce modalità standard per il passaggio di interrogazioni o di informazioni specifiche tra client e server.

Questa classe non fa nulla per definire il formato dei tipi di dati specifici, per far ciò bisogna creare una sottoclasse. Ad esempio **QbGenericImageDataClass** permette la gestione specifica delle immagini.

QbGenericImageDataClass è una sottoclasse di QbGenericDataClass ed include tutti i metodi della superclasse più una serie di metodi specifici per l'elaborazione delle immagini come la lettura dalla memoria o dal disco, la de-

terminazione delle dimensioni dell'immagine, la gestione delle LUT e delle maschere.

I formati di immagini gestiti nella presente versione includono i file con estensione bmp, gif, jpg, pgm, ppm, tif e tga.

QbKeyDatabaseClass è una sottoclasse di QbDatabaseClass che viene impiegata per restituire i risultati di un'interrogazione oppure per specificare una "Restriction list" di chiavi.

La "restriction list" è un sottoinsieme di chiavi, tipicamente derivato da una query precedente, che viene utilizzato nelle successive per restringere il dominio di ricerca ed lo strumento attraverso il quale, a più alto livello, vengono implementate le query multi-feature e multi-pass.

La **QbDatabaseClass** è un archivio di coppie chiave/valore di distanza, la creazione di cursori è necessaria per recuperare gli elementi in ordine crescente di distanza.

I metodi implementati per questa classe comprendono quelli per la lettura e la scrittura del valore della distanza associata alla chiave,quelli per recuperare il numero di chiavi presenti nell'oggetto ed il valore massimo di distanza presente.

QbParameterClass è derivata dalla classe QbBaseClass ed incapsula i parametri associati con la determinazione di una feature di QBIC o di una interrogazione. E' totalmente implementata e crea un vettore di coppie nome/valore che ogni classe derivata da QbFeatureClass può utilizzare.

Ogni volta che viene creato un oggetto feature di una certa classe vengono assegnati anche i parametri di default che possono essere "aggiustati" secondo le esigenze degli utenti.

QbQueryClass deriva da QbBaseClass ed è una classe astratta che definisce il metodo Evaluate che permette l'esecuzione dell'interrogazione e necessita di una serie di parametri di ingresso comprendenti: il numero di risultati richiesti, il catalogo da interrogare, le eventuali liste di restrizione e maschere, il numero e il tipo di feature coinvolte, i dati dell'immagine d'esempio da utilizzare nella ricerca e i parametri di query necessari.

Per il sistema Dbm viene fornita una sottoclasse **QbQueryServerClass** che definisce l'implementazione dei metodi virtuali della relativa superclasse.

In molte delle classi vengono inclusi due metodi detti **ToQueryString** e **FromQueryString** che trasformano un oggetto di tipo QbDatumClasse in una stringa e viceversa, permettendo il passaggio dallo spazio di processo del sistema "client" a quello del sistema "server" nel caso che siano fisicamente residenti su sistemi diversi.

Capitolo 8

Implementazione di un IDB tramite QBIC

In questo capitolo si fornisce la descrizione dell'ambiente e del linguaggio di programmazione utilizzati, un'analisi degli strumenti software acquisiti da altri programmatori ed una descrizione dell'interfaccia e delle funzioni del programma (QbicProg.exe).

8.1 Ambiente e linguaggio utilizzato.

Il sistema è stato sviluppato in ambiente Windows 9x ed il linguaggio utilizzato per l'implementazione è il Visual C++ Versione 5.0.

I vantaggi offerti dall'uso di Windows sono principalmente legati alla comodità dell'ambiente grafico fornito. Con l'utilizzo di questo ambiente sono possibili funzionalità quali un'interfaccia grafica standard, il Multitasking, l'approccio alla programmazione Object Oriented, il controllo della memoria, l'indipendenza dall'Hardware installato e l'uso di librerie a collegamento dinamico (DLL).

Il linguaggio Visual C++ permette con relativa facilità di progettare GUI (Graphics User Interface) per mezzo della quale l'utente può interagire con il sistema. Il fatto che tutte le applicazioni Windows condividano lo stesso sistema di finestre di dialogo e menù e che queste siano disponibili al programmatore, che le può inserire nel proprio programma, fa si che l'utilizzo del sistema risulti molto intuitivo.

8.2 La libreria di Classi MFC.

Il compilatore Visual C++ di Microsoft è corredato di una libreria MFC (Microsoft Fundation Class) che contiene una serie di strumenti di sviluppo orientati agli

oggetti. Poiché l'applicazione sviluppata gestisce documenti multipli, per mezzo di MFC sono stati definititi i quattro moduli principali per la gestione di questo tipo di applicazioni [**?**]:

- **CMainFrame:** contiene le classi utilizzate per controllare le funzioni dell'interfaccia del documento 'main'. Attraverso i menù del documento 'main' l'utente può scegliere il tipo di operazione che intende eseguire: inserimento e cancellazione delle immagini nel Database, interrogazioni, ecc. . .

- **CChildFrame:** contiene la classe utilizzata per controllare le funzioni di interfaccia del documento 'figlio' che contiene l'output del sistema, cioè l'insieme delle immagini che costituiscono il risultato dell'interrogazione.

- **CDoc:** contiene la classe che specifica il documento dell'applicazione, il suo scopo è quello di memorizzare i dati grezzi: stringhe, numeri interi e floating point, ecc. . .

- **Cview:** fornisce il supporto alla visualizzazione dei dati memorizzati nel documento: è in questa classe che si implementa il codice per la visualizzazione delle immagini.

8.3 Architettura del sistema.

Nel programma sviluppato possono essere identificate le seguenti unità funzionali:

- libreria di Classi fornita dalla IBM che implementa la gestione del Database QBIC descritta nel Capitolo 7;

- libreria di classi CIMAGE[1] utilizzata per la visualizzazione nella vista dei risultati dell'interrogazione;

- il programma QbicProg.exe con le relative classi che tramite i due moduli precedenti implementa le funzioni di inserimento e cancellazione di immagini nel catalogo e permette la definizione dell'interrogazione, la sua esecuzione e la visualizzazione dei risultati.

[1]La Cimage è una libreria sviluppata da Alejandro Aguilar Sierra ed è disponibile all'indirizzo: **http://web.ukonline.co.uk/julian.smart/code/cimage/cimage.zip**

8.4 La libreria CImage.

La CImage è una libreria scritta in linguaggio C++ in grado di gestire immagini in formati diversi che viene impiegata nella fase di visualizzazione dei risultati dell'interrogazione.

In questa versione (1.2) la libreria permette di leggere e salvare i formati bmp, jpg e png e leggere solamente il formati gif. I metodi principali utilizzati dalla classe per la visualizzazione delle immagini sono:

- `void CImage(void);` è il costruttore che permette di inizializzare un oggetto appartenete alla classe CImage;

- `BOOL ReadFile(const CString& filename, int imageType);` legge un'immagine, dove filename identifica il suo nome nel filesystem, e imageType definisce il tipo di immagine da leggere. ImageType può assumere i valori CIMAGE_ FORMAT_ BMP, CIMAGE_ FORMAT_ GIF, CIMAGE_ FORMAT_ JPEG, CIMAGE_ FORMAT_ PNG oppure -1 che determina il tipo di immagine dall'estensione del file.

- `BOOL Draw(CDC *dc, int dx, int dy, int dw, int dh, int sx, int sy);` disegna l'immagine letta con ReadFile sul device context puntato da *dc. Il device context è una struttura dati di MFC che tiene traccia degli attributi della superficie di disegno di una finestra ed inoltre realizza l'indipendenza dal hardware installato. Gli altri parametri specificano:

  dx, dy l'angolo in alto a sinistra del rettangolo di destinazione dell'immagine;

  dw, dh le dimensioni del rettangolo che contiene l'immagine da disegnare;

  sx, sy l'angolo in alto a sinistra da cui iniziare a disegnare l'immagine.

- `BOOL Stretch(CDC *dc, int dx, int dy, int dw, int dh, int sx, int sy, int sw, int sh)` come la funzione `Draw`, solamente che in questo caso si possono variare le dimensioni dell'immagine visualizzata: in questo caso sw ed sh rappresentano le dimensioni dell'immagine sorgente che possono differire da quelle del rettangolo da disegnare. Questa funzione viene utilizzata dal programma per creare le versioni miniaturizzate delle immagini appartenenti al risultato dell'interrogazione.

8.5 QbicProg: gestione delle interrogazioni.

Il primo passo obbligatorio per l'esecuzione di un'interrogazione, ma anche per le operazioni di popolazione del Db, consiste nella apertura di un catalogo all'interno di un Database.

In Figura 8.1 è riportata la finestra di dialogo visualizzata attraverso l'opzione di menù apri/crea Catalogo che contiene due caselle di testo per specificare il nome del database ed il relativo catalogo da aprire, nella casella database è contenuto un Db di default.

La coppia di pulsanti di opzione permette di definire la modalità di apertura in lettura od in scrittura (per le query normalmente si apre in lettura). Chiudendo la finestra tramite il pulsante Ok viene visualizzato il risultato dell'operazione di apertura (successo o insuccesso). La seconda operazione necessaria all'ese-

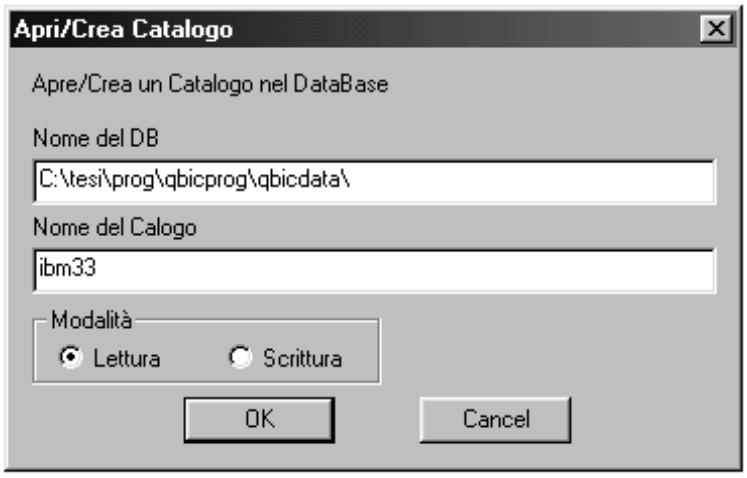


Figura 8.1: Finestra di dialogo per l'apertura/creazione di un nuovo catalogo in un Database.

cuzione di una query consiste nel definire l'insieme delle feature da utilizzare e la modalità con cui queste interagiscono nel recupero delle immagini.

In Figura 8.2 è riportata la finestra di dialogo visualizzata attraverso l'opzione di menù Inserisci feature che contiene una casella di riepilogo che permette di scegliere le diverse tipologie di feature implementate da QBIC, una casella di testo per specificare il peso da attribuire alla feature nel caso di query di tipo multi-feature ed una coppia di caselle di controllo che permettono di selezionare la modalità di calcolo della distanza e l'abilitazione della modalità di interrogazione multi-pass. Infine una coppia di pulsanti di opzione permette di specificare se la feature viene inserita ai fini dell'interrogazione o della popolazione del Db.

Al termine delle operazioni di inserimento dei parametri della feature premendo Ok viene visualizzato il risultato dell'operazione compiuta (feature inserita o no con successo) ed un riassunto dei parametri inseriti.

Per modificare i parametri associati ad una feature già inserita basta reinserirla modificando i necessari parametri. Nel caso in cui una feature da inserire non sia contenuta all'interno del catalogo aperto viene presentato un messaggio di errore dato che l'eventuale interrogazione, su quella feature, non ritornerebbe alcun risultato. Una finestra analoga a quella di Figura 8.2, ma senza la casella di testo

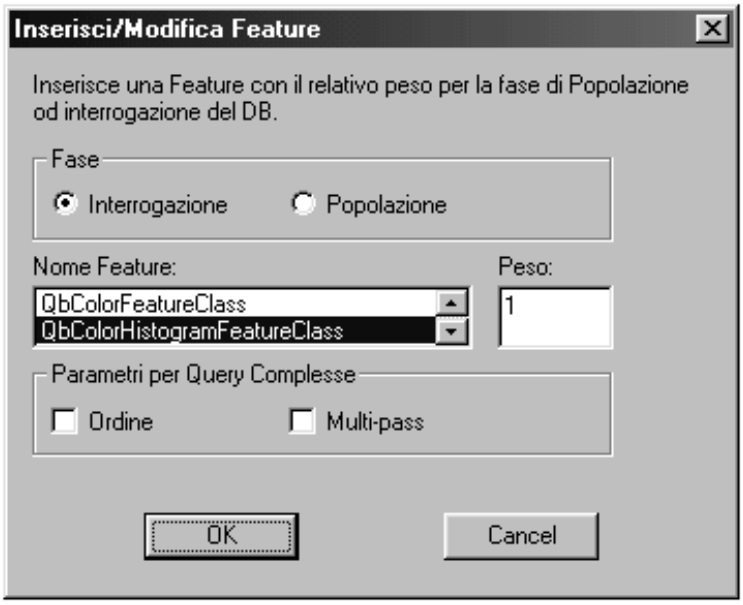


Figura 8.2: Finestra di dialogo per l'inserimento di una nuova feature nel catalogo.

e senza le caselle d'opzione, permette di eliminare una feature precedentemente inserita.

Premendo Ok viene visualizzato il risultato dell'operazione eseguita ed una lista delle feature ancora attive, oppure viene visualizzato un messaggio d'errore se la feature da cancellare non è stata precedentemente inserita.

Un'altra operazione fondamentale per l'esecuzione di una interrogazione consiste nel definire un'immagine d'esempio in modo da poter recuperare quelle più "simili".

Questa operazione viene effettuata attraverso la finestra di dialogo (Figura 8.3) visualizzata selezionando la voce Seleziona immagine dal menù QBIC Query Db che contiene un campo casella di testo dove specificare, in funzione dei pulsanti

di opzione, o un valore di chiave per un'immagine interna al Db oppure il nome di un file per un'immagine esterna.

Una discussione a parte richiede la spiegazione del pulsante d'opzione Picker, che permette di specificare come immagine d'esempio il Picker che sarà introdotto nel Paragrafo 8.7. Anche in questo caso premendo Ok viene illustrato il

Figura 8.3: Finestra di dialogo per la definizione dell'immagine d'esempio su cui eseguire la query.

risultato dell'operazione (successo o insuccesso), in particolare viene visualizzato un messaggio negativo nel caso che la chiave od il nome del file specificato non esista.

Dopo aver specificato il catalogo da aprire, inserito le feature necessarie e specificato l'immagine d'esempio è possibile mandare in esecuzione l'interrogazione selezionando la voce esecuzione Query nel menù QBIC Query Db.

Questo comando provoca l'apertura della corrispondente finestra di dialogo di Figura 8.4 che contiene una casella di testo con il relativo dispositivo di incremento/decremento che permette di variare il numero, compreso tra 1 e 100, delle immagini visualizzate nel risultato dell'interrogazione. Spuntando la casella di controllo sottostante è possibile ottenere in risposta all'interrogazione un insieme casuale di immagini, questa opzione necessita solamente che sia stato aperto un catalogo nel Database.

In questo caso premendo il tasto Ok oltre ad essere visualizzati eventuali messaggi d'errore, causati da un'errata definizione dei parametri delle finestre

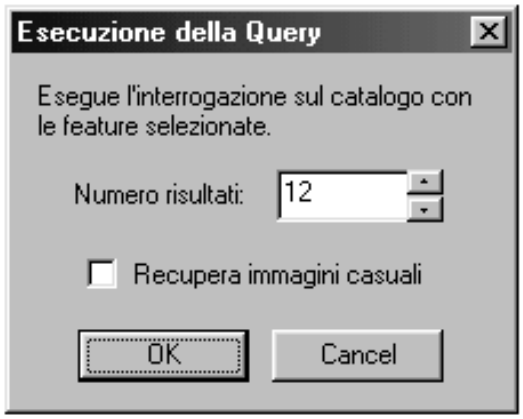


Figura 8.4: Finestra di dialogo per l'esecuzione dell'interrogazione.

precedenti, vengono visualizzate nella vista corrente le miniature delle immagini recuperate nel numero specificato.

Per ogni immagine visualizzata viene fornito il nome e la relativa distanza dall'immagine d'esempio utilizzata nell'interrogazione. Nel caso che il numero di immagini ecceda le dimensioni della finestra di visualizzazione appaiono le classiche barre di scorrimento che permettono di visualizzare le immagini nascoste.

In figura 8.5 sono visualizzate le prime sei immagini ottenute interrogando il catalogo ibm33 con l'immagine flower04.jpg utilizzando l'istogramma dei colori. Ridimensionando la finestra le immagini vengono disposte dinamicamente in modo da occupare sempre il minor spazio possibile.

8.6 QbicProg: popolazione del Database.

Per poter eseguire un'interrogazione bisogna prima inserire una serie di immagini nel database; questa operazione prende il nome di popolazione del Database.

In questa fase è possibile sia aggiungere nuove immagini ad un catalogo esistente sia crearne uno nuovo: in entrambi i casi è necessario tramite la finestra di dialogo apri/crea Catalogo in Figura 8.1 aprire il catalogo e specificare l'opzione di apertura in scrittura.

Analogamente al caso dell'interrogazione bisogna definire quali tipologie di feature verranno estratte per le immagini da inserire, questa operazione si effettua sempre attraverso la finestra di dialogo Inserisci feature in Figura 8.2 specificando l'opzione "popolazione". In questo modo è possibile inserire le feature anche se

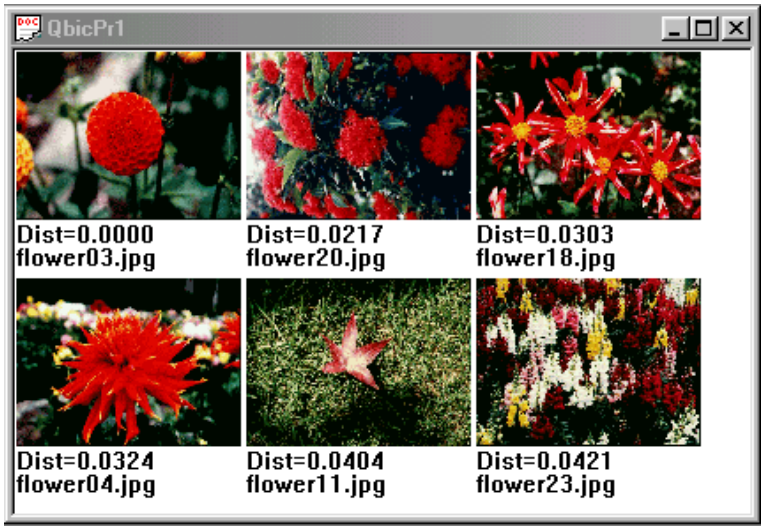


Figura 8.5: Risultati dell'interrogazione su flower04.jpg utilizzando l'istogramma dei colori.

non sono già presenti nel catalogo, operazione che risulterebbe del tutto inutile nel caso di un'interrogazione.

I valori dei parametri come il peso, l'ordine e la modalità multi-pass sono del tutto ininfluenti ai fini dell'inserimento delle immagini nel catalogo. I nomi delle immagini da inserire nel catalogo aperto, per le opzioni di feature selezionate, vengono immesse attraverso la finestra di dialogo Inserisci immagini, attivabile attraverso il menù QBIC Popola Db.

Questa finestra di dialogo (Figura 8.6) contiene una coppia di caselle di testo: nella prima viene inserito il percorso nel filesystem che individua la directory contenente le immagini, la seconda casella invece permette di definire la lista delle immagini da inserire. Questa consiste in una serie di nomi di file con relativi caratteri 'jolly' (? e *) separati da almeno uno spazio.

La chiave che identifica in maniera univoca ogni immagine inserita nel catalogo del database è ottenuta estraendo dal percorso completo il nome più l'estensione del file che lo memorizza.

Premendo il pulsante Ok tutte le immagini vengono inserite nel catalogo estraendo le feature specificate nel passo precedente, se un'immagine da inserire non esiste oppure è in un formato non gestito da QBIC viene visualizzato un messaggio d'errore.

Attraverso la finestra di dialogo Elimina Immagine in Figura 8.7 dello stesso menù è possibile eliminare tutte le feature di una data immagine inserite in un certo catalogo. Nella casella di testo contenuta nella rispettiva finestra di dialogo

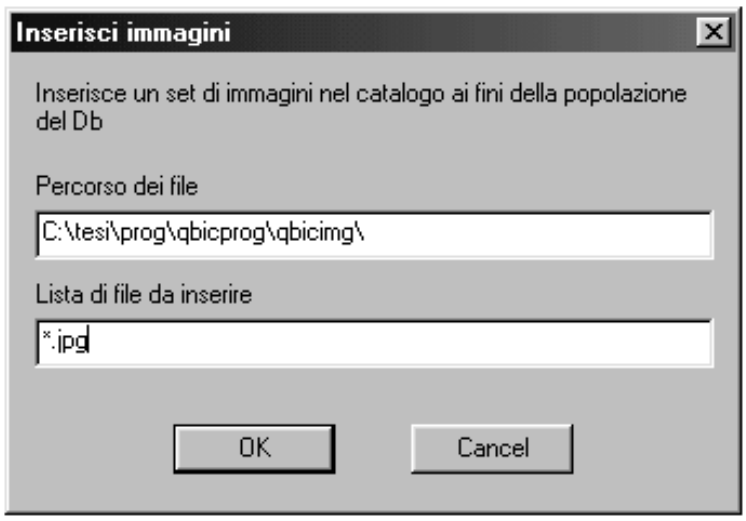


Figura 8.6: Finestra di dialogo per la definizione delle immagini da inserire nel catalogo.

viene inserita la chiave dell'immagine da eliminare. Premendo Ok è possibile visualizzare il risultato (successo o insuccesso) dell'operazione di cancellazione.

Utilizzando la voce Conta Immagini del menù QBIC Popola Db in Figura 8.8 è possibile ottenere il numero di immagini inserite nel catalogo per una data feature. La feature da utilizzare per il calcolo viene definita attraverso una casella di riepilogo che contiene i nomi delle feature gestite dalla versione esistente del programma. Selezionandone una e premendo Ok si visualizza il numero delle immagini per le quali, nella fase di popolazione, è stata estratta la feature specificata.

Queste sono le funzioni di base necessarie alla gestione del Database di immagini di QBIC che permettono di inserire, eliminare e ricercare le immagini inserite. È possibile sviluppare funzioni evolute sia per la fase di popolazione che per quella di interrogazione: in particolare è possibile definire interrogazioni attraverso i Picker cioè stringhe di testo che definiscono criteri di ricerca su una serie di zone di colore da individuare all'interno delle immagini, oppure, per la fase di popolazione, è possibile definire, attraverso opportune maschere binarie, una serie di oggetti all'interno delle immagini da cui estrarre un set di feature autonome.

8.7 Esecuzione di interrogazioni tramite Picker.

Uno strumento molto potente, per l'esecuzione di interrogazioni visuali, fornito dal sistema QBIC, è il "Color Picker". Questo permette di specificare l'immagine

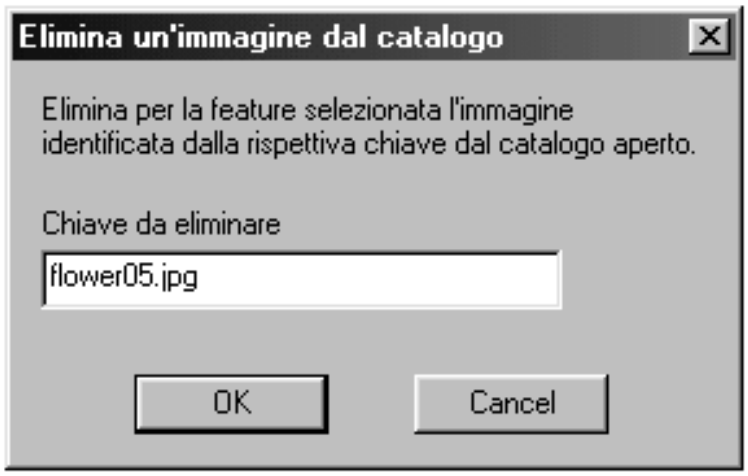


Figura 8.7: Finestra di dialogo per la definizione della chiave dell' immagine da eliminare dal catalogo.

di query, nella versione attuale, definendo la posizione di una serie di rettangoli di colore omogeneo.

La particolarità del sistema è che non è necessario specificare tutti i rettangoli che compongono l'immagine, ma solamente le parti che interessano maggiormente, mentre quelle lasciate in bianco vengono semplicemente trascurate. QBIC memorizza il Picker attraverso una stringa di testo avente la seguente struttura:

Dwidith,height:Rulx,uly,rwidth,rheight,R,G,B:R. . .

dove:

D specifica l'area di disegno definita attraverso lunghezza e larghezza e con l'angolo in alto a sinistra nell'origine;

: è il separatore che delimita i vari rettangoli che si vogliono utilizzare come esempio ai fini della definizione dell'interrogazione;

R specifica un rettangolo con l'angolo superiore sinistro di coordinate ulx e uly, lungo rwidith e largo rheight con il colore definito nello spazio RGB dai tre interi R,G,B.

Nel caso in cui venga inserito più di un rettangolo, l'ordine di inserimento determina la modalità di sovrapposizione, in particolare i rettangoli successivi al primo vanno a coprire, se sovrapposti, i precedenti. Il programma Qbicprog.exe implementa visivamente l'inserimento dei rettangoli all'interno del Picker. All'interno voce di menù Color Picker sono contenuti tutti gli strumenti necessari alla definizione del Picker, che viene visualizzato all'interno della finestra attiva.

Figura 8.8: Finestra di dialogo per il calcolo del numero di immagini presenti per una data feature.

Tramite la voce Dimensione Picker del precedente menù si accede alla finestra di dialogo di Figura 8.9 che contiene due caselle di testo al cui interno è possibile specificare le dimensioni del picker (larghezza e altezza) in pixel.

Premendo Ok viene disegnato un rettangolo col bordo di colore nero che delimita l'area all'interno della quale è possibile inserire i rettangoli che definiscono l'interrogazione.

Il passo preliminare all'inserimento di un qualunque rettangolo all'interno dell'area di disegno del Picker consiste nel definire la tonalità di colore che il rettangolo deve assumere. QBIC richiede che il colore sia definito attraverso la terna dello spazio RGB e l'immissione del colore selezionato avviene attraverso la finestra di dialogo Seleziona colore dal menu color Picker.

Questa finestra di dialogo (Figura 8.10) contiene tre barre di avanzamento e tre caselle di testo una per ogni canale di colore nello spazio RGB.

Spostando ognuna delle barre disponibili si va ad agire sulla rispettiva casella di testo che riporta il valore intero compreso tra 0 e 255 assegnato a ciascun canale di colore.

Se la finestra di dialogo non viene aperta o viene premuto Cancel il colore assegnato di default è il bianco.

Dopo aver selezionato un colore è necessario disegnare un rettangolo e ciò avviene tramite la voce di menù Inserisci Rettangolo che non apre nessuna finestra di dialogo, ma permette di selezionare tramite il mouse i due vertici su lati opposti del rettangolo da inserire.

La selezione avviene premendo il tasto sinistro del mouse in corrispondenza

Figura 8.9: Finestra di dialogo per la definizioni delle dimensioni dell'area di disegno del Picker.

del punto che identifica il vertice, avendo cura di inserire prima quello in alto a sinistra e poi l'altro in basso a destra del rettangolo.

Dopo aver disegnato i rettangoli, all'interno del picker, necessari alla definizione dei criteri che devono rispettare i risultati dell'interrogazione occorre, come nei casi precedenti, informare il sistema che si intende impiegare il picker. Ciò avviene attraverso la finestra di Figura 8.3descritta nel Paragrafo 8.5 settando il pulsante di opzione Picker.

Premendo Ok viene visualizzata la stringa utilizzata da QBIC per rappresentare il picker nella fase di interrogazione.

8.8 Modalità di compilazione del codice.

In questo paragrafo sono riportati i riferimenti per recuperare il codice, in parte reperito presse terze parti, utilizzato per produrre il programma Qbicprog.exe.

Il codice prodotto in questa tesi è disponibile direttamente all'indirizzo ftp del server del DSI:

ftp.sparc20.dsi.unimo.it

Nel direttorio:

/export/home/progetti.comuni/tesi/torricel/prog

In particolare nei seguenti sotto direttori sono contenuti file relativi a:

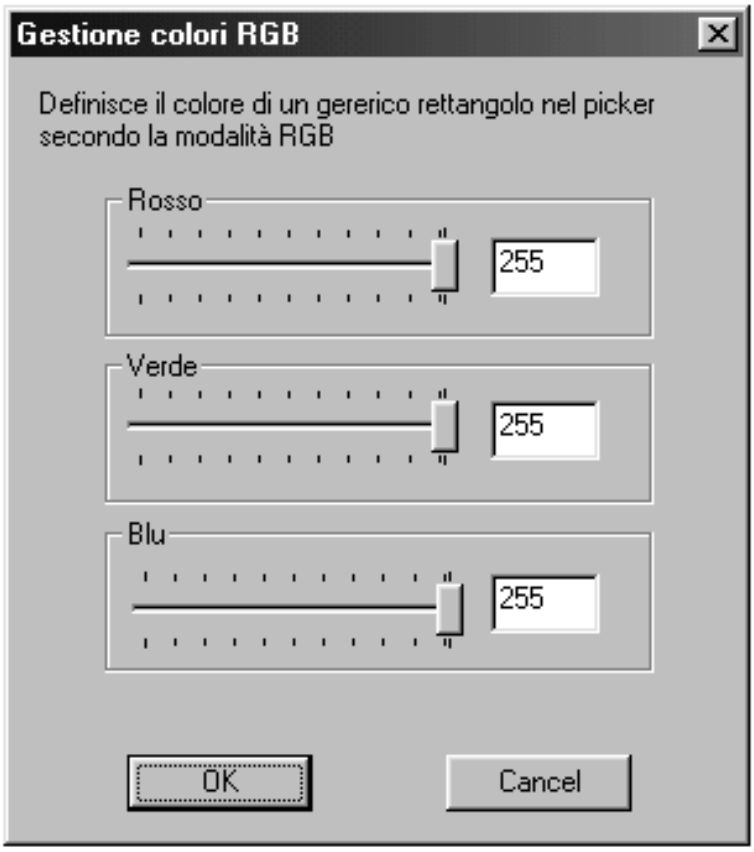


Figura 8.10: Finestra di dialogo per la selezione del colore nello spazio RGB.

/Cimage	Libreria di classi per la visualizzazione in una finestra Windows delle immagini in versione miniaturizzata;
/jpeg	Librerie di classi per l'apertura ed il salvataggio delle immagini in formato compresso jpg;
/png	Librerie di classi per l'apertura ed il salvataggio delle immagini in formato png;
/zlib	Librerie per la gestione della compressione nel formato png;
/Qbic	Librerie di classe per la gestione delle funzionalità di interrogazione e popolazione del Database di immagini di QBIC;
/Qbicprog	File del programma in Visual C++ 5.0, che ho sviluppato, contenente le primitive MFC per la realizzazione dell'ambiente a finestre utilizzato nella Tesi.

In particolare nel direttorio /Qbic sono contenute i sotto direttori /lib per le

librerie, /include per i file include, /dll contenente le librerie a collegamento dinamico ed il file QbicWrap.cpp contente il file sorgente delle classi di più alto livello.

Per ottenere l'eseguibile è necessario disporre dell'ambiente di programmazione Microsoft Visual C++ 5.0 e copiare i file contenuti nel direttorio /prog a partire da:

C:\tesi\prog

All' interno dei direttori /cimge, /jpeg, /png, /zlib sono presenti i file di progetto con estensione *.dsw che permettono di settare tutti i parametri da utilizzare nella compilazione per ottenere le librerie necessarie al funzionamento del programma QbicProg.exe.

Per i file nel sotto direttorio /Qbic non sono forniti i file sorgenti però sono disponibili le librerie già compilate ed i file dll che devono essere copiati nel direttorio contenete l'eseguibile finale.

I file nel sottodirettorio /qbicprog contengono i file necessari alla compilazione del eseguibile sviluppato in questa tesi, aprendo il file QbicProg.dsw si settano i parametri per la compilazione ed il link di tutte le librerie utilizzate.

Nel file di include global.h sono inseriti i riferimenti ad una serie di percorsi nel filesystem per recuperare i dati sulle immagini e sulle feature estratte: se si decide di modificare qualche percorso bisogna riflettere i cambiamenti in questo file.

All'interno del sottoalbero del direttorio /qbicimage sono contenute sottodirettori che memorizzano i file delle immagini su cui estrarre le feature che QBIC si occupa di memorizzare come sottodirettori del direttorio /qbicdata.

I file contenenti i capitoli della presente tesi realizzata attraverso il programma di composizione tipografica LaTeX sono disponibili sempre sul server nel direttorio:

/export/home/progetti.comuni/tesi/torricel/tex

La versione demo del Database QBIC è nel file qbnt_ 3.zip, il pacchetto completo di Cimage è in cimage.zip e il Database di immagini WindSurf è in projimage.zip tutti nel direttorio /prog.

Capitolo 9

Risultati sperimentali con QBIC e WindSurf

In questo capitolo vengono analizzate, confrontate e discusse le prestazioni dei motori di ricerca QBIC e WindSurf eseguendo una serie di test pratici sulle versioni disponibili.

In particolare, mentre per il primo è disponibile una versione dimostrativa prelevabile su Internet, per il secondo invece si dispone del software completo sviluppato dall'Università di Bologna.

9.1 Modalità di valutazione.

Sebbene siano state definite misure per la valutazione dell'efficacia e dell'efficienza dei sistemi di ricerca basati sul contenuto (Vedi Paragrafo 1.4), in realtà non esistono criteri oggettivi in grado di valutare, in maniera adeguata, la bontà dei risultati ottenuti.

In generale la bontà del risultato dipende dalle valutazioni soggettive dell'utente. Da quanto visto in precedenti lavori in ambito CBVQ le valutazioni fatte sono di tipo 'comparativo', ovvero si cerca di stimare la bontà relativamente ad altri sistemi che adottano tecniche differenti per l'estrazione delle feature.

Lo scopo delle valutazioni è determinare quali benefici introduce l'adozione di strumenti per l'analisi tempo/frequenza (cioè quelli adottati da WindSurf) rispetto ai più tradizionali strumenti basati sulla daterminazione degli istogrammi (cioè quelli adottati da QBIC).

Le valutazioni effettuate mirano principalmente a comparare la bontà dei risultati ottenuti trascurando totalmente il confronto tra i tempi necessari all'estrazione delle feature ed al confronto delle stesse durante l'esecuzione delle interrogazioni.

Infatti il sistema WindSurf sotto questo aspetto risulta nettamente peggiore sia

per il fatto che le tipologie di feature da estrarre sono più complesse da calcolare rispetto agli istogrammi, sia perché allo stato attuale del progetto non è ancora stata implementata alcuna politica di indicizzazione per il recupero delle feature durante la fase di interrogazione del database.

In particolare attualmente viene impiegata la scansione sequenziale, ma si sta valutando l'introduzione di un'organizzazione di più alto livello basata sugli alberi metrici [**?**].

9.2 Analisi delle capacità di interrogazione.

La versione "demo" di QBIC, ottenuta al sito della IBM, contiene una serie di limitazioni: in particolare non sono incluse le funzioni necessarie alla gestione degli oggetti che QBIC può estrarre in maniera semi-automatica attraverso l'interazione con l'utente. Ogni oggetto estratto viene così trattato alla stessa maniera delle immagini calcolando tutte le feature implementate e permettendo il recupero da parte delle interrogazioni.

Nella versione "demo" è possibile effettuare le seguenti interrogazioni:

Istogramma dei colori: è possibile recuperare 12 immagini alla volta in base alla similitudine dell'istogramma dei colori;

Disposizione: l'immagine è scomposta attraverso la sovrapposizione di una griglia con un numero fisso di celle all'interno delle quali vengono estratte feature sul colore,come il colore medio, che viene confrontato, attraverso le solite funzioni di distanza, con le feature della corrispondente cella nell'immagine di query;

Texture: si recuperano via via lotti di 12 immagini più simili in base alle feature sulla tessitura;

Ibrido: si recuperano le immagini attraverso una query di tipo multi–feature considerando, per la determinazione della similitudine, sia le feature sul colore che sulla tessitura.

In WindSurf, disponibile in versione "completa", invece è possibile specificare una serie di parametri per l'estrazione delle feature, come il tipo ed il livello di trasformata Wavelet, lo spazio di colore da utilizzare nella trasformazione e la metrica impiegata dall'algoritmo K–Means per l'estrazione delle regioni.

9.3 Descrizione del Test-set.

Le immagini impiegate nella valutazione delle prestazioni dei due sistemi (WindSurf *vs.* QBIC) sono state estratte dal CD-Rom numero 7 appartenente ad IMSI PHOTOS [1]; le immagini contenute in questo CD-Rom hanno dimensioni variabili tra 320×240 e 400×300 e sono organizzate in classi di soggetti semanticamente omogenee.

Per semplificare le valutazioni abbiamo creato un Test–Set di circa 570 immagini ottenute selezionandone, in maniera casuale, un numero variabile all'interno di ciascun raggruppamento semantico presente nel CD-Rom.

I Test–Set utilizzati sono elaborati da WindSurf specificando come classe di trasformata la Wavelet di Daubechies_11 iterata tre volte sullo spazio di colore HSV ed applicando per il Clustering la metrica di Mahalanobis.

9.4 Prove e Risultati.

Le prove riportate in questo paragrafo sono il sottoinsieme più significativo di un più vasto insieme di test a cui sono stati sottoposti entrambi i sistemi.

Come prima prova interroghiamo entrambi i sistemi fornendo un'immagine di query rappresentante la bandiera americana (301200.jpg). All'interno del Test–set sono presenti ben 9 immagini che rappresentano una bandiera americana completa, cioè con le strisce bianche e rosse e le stelle bianche sullo sfondo blu, riprese da angolazioni diverse, con illuminazioni diverse e con in trasparenza altre immagini. Inoltre sono presenti ulteriori 6 immagini rappresentanti altrettanti particolari della bandiera.

Nelle Tabelle 9.1 e 9.2 sono riportate le miniature delle prime 12 immagini appartenenti al risultato dell'interrogazione sui sistemi QBIC e WindSurf.

Come si può osservare in Tabella 9.1, i risultati migliori si ottengono utilizzando per l'interrogazione l'istogramma dei colori, che permette di recuperare solamente 4 immagini valide, mentre impiegando l'opzione riguardante la distribuzione spaziale dei colori si ottengono risultati peggiori per il fatto che l'immagine, presentando una dinamica dei colori fortemente variabile, non trova giovamento dalla suddivisione in celle. I risultati ottenuti con WindSurf in Tabella 9.2 sono migliori di quelli di QBIC per il fatto che vengono recuperate ben 6 immagini contro le 4 di QBIC.

In questa interrogazione come immagine di query impieghiamo un'immagine che rappresenta un aeroplano ad elica di colore rosso su sfondo azzurro ripreso da diverse angolazioni(B10374.jpg). Nel Test–Set precedentemente definito sono presenti 9 immagini valide come risultato per questa query.

[1] IMSI Master Photos 50.000 disponibili all'indirizzo Internet: **http://www.imsisoft.com**

Nelle Tabelle 9.3 e 9.4sono riportate le miniature delle prime 12 immagini appartenenti al risultato dell'interrogazione sui sistemi QBIC e WindSurf.

In questo caso, eseguendo l'interrogazione sul motore di ricerca QBIC, si recuperano, in Tabella 9.3, ben 5 immagini valide ottenute utilizzando l'opzione di query riguardante la disposizione spaziale di zone di colore, che questa volta ottiene risultati migliori, al contrario della precedente, rispetto all'utilizzo dell'istogramma. Altre 4 immagini contenute nella risposta presentano un'elevata predominanza di tonalità di colore vicine al rosso.

La motivazione di questo comportamento è probabilmente dovuta al fatto che, in questo caso, sono presenti solo due zone di colore rilevanti: la zona centrale rappresentante l'aeroplano e lo sfondo azzurro.

Il risultati ottenuti con WindSurf in Tabella 9.4 sono migliori di quelli di QBIC per il fatto che vengono recuperate ben 8 immagini contro le 5 di QBIC.

In quest'ultima interrogazione utilizziamo un'immagine d'esempio che rappresenta un dirigibile della II guerra mondiale. Questo è, rispetto ai precedenti, un caso particolare perché utilizza un'immagine a toni di grigio.

Nel Test–Set precedentemente definito sono presenti 8 immagini valide. Nelle Tabelle 9.5 e 9.6 sono riportate le miniature delle prime 12 immagini appartenenti al risultato dell'interrogazione sui sistemi QBIC e WindSurf.

Questa interrogazione eseguita da QBIC, utilizzando la distribuzione delle zone di colore, restituisce le immagini in Tabella 9.5 e mette in luce i profondi limiti del sistema nella gestione delle immagini a toni di grigio, infatti, in questo caso, vengono recuperate solamente 3 immagini di molto inferiori alla totalità di quelle presenti nel Data-set.

È da notare che, pur essendo in numero ridotto le immagini a toni di grigio presenti nel data set, il sistema ha recuperato molte immagini a colori, il che porta a pensare ad un'errata gestione di questa tipologia di immagini.

Al contrario, il risultato ottenuto da WindSurf in Tabella 9.6 è ottimo perché vengono recuperate tutte e 8 le immagini valide presenti nel Test–Set e l'unica immagine a colori inclusa ha predominanti sfumature di grigio.

			
301200.jpg	B10373.jpg	301500.jpg	301000.jpg
			
B10372.jpg	B35772.jpg	B10370.jpg	B35770.jpg
			
300900.jpg	B43947.jpg	B43946.jpg	B43119.jpg

Tabella 9.1: Risultati ottenuti interrogando QBIC sull'immagine 301200.jpg.

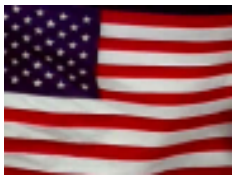
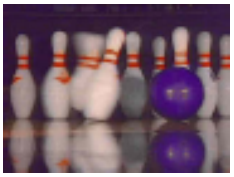
			
301200.jpg	301500.jpg	301000.jpg	300900.jpg
			
301900.jpg	B10370.jpg	301600.jpg	B11151.jpg
			
B42265.jpg	665700.jpg	B11153.jpg	B41889.jpg

Tabella 9.2: Risultati ottenuti interrogando WindSurf sull'immagine 301200.eps.

			
B10374.jpg	B10372.jpg	B10373.jpg	B10371.jpg
			
B10370.eps	B9966.jpg	B9888.jpg	B47697.jpg
			
B9887.eps	B35730.jpg	B9893.jpg	680000.jpg

Tabella 9.3: Risultati ottenuti interrogando QBIC sull'immagine B10374.jpg.

			
B10374.jpg	B10367.jpg	B10372.jpg	B10368.jpg
			
B43948.jpg	B41889.jpg	B43949.jpg	B10373.jpg
			
B10371.jpg	B9966.jpg	B103746.jpg	B10370.jpg

Tabella 9.4: Risultati ottenuti interrogando WindSurf sull'immagine B10374.jpg.


B13870.jpg


B43514.jpg


B13973.jpg


B13867.jpg


B13868.jpg


B9909.jpg


B30314.jpg


B10580.jpg


B41492.jpg


B10389.jpg


667100.jpg

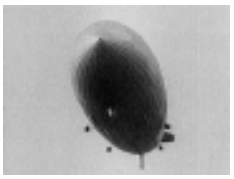

571100.jpg

Tabella 9.5: Risultati ottenuti interrogando QBIC sull'immagine B13870.jpg.


B13870.jpg


B13869.jpg


B13973.jpg


B13873.jpg


B13867.jpg


B13868.jpg


B13872.jpg


B13874.jpg


B35320.jpg


B13871.jpg


B13799.jpg


B12437.jpg

Tabella 9.6: Risultati ottenuti interrogando WindSurf sull'immagine B13870.eps.

In tutte le query eseguite l'opzione fornita da QBIC riguardante le tessiture ha fornito risultati deludenti, recuperando un numero di immagini valide sempre minore rispetto a quelle della altre opzioni disponibili, mentre la modalità di interrogazione ibrida riportava gli stessi risultati di quella sull'istogramma.

Nella maggior parte dei casi in QBIC l'impiego dell'opzione riguardante la distribuzione spaziale delle zone di colore ha ottenuto risultati migliori rispetto al più semplice istogramma, anche se fatica a competere con l'algoritmo di estrazione automatica di regioni implementato da WindSurf.

Al fine di valutare più approfonditamente le potenzialità dei due algoritmi nel riconoscimento delle tessiture creiamo un nuovo Test–Set contenente una serie di campioni di tessitura prelevati dalla collezione Brodatz[2].

Brodatz in [**?**] ha raccolto in una pubblicazione centinaia di immagini rappresentanti le tessiture di elementi naturali e manufatti umani che sono impiegati da tutti gli sviluppatori per testare le potenzialità dei propri sistemi.

Brodatz Texture		
Nome File	Contenuto	Cod.
1101.jpg	Erba	(D9)
1102.jpg	Corteccia	(D12)
1103.jpg	Paglia	(D15)
1104.jpg	Tessuto a spina di pesce	(D15)
1105.jpg	Tessuto di lana	(D19)
1106.jpg	Pelle di vitello pressata	(D24)
1107.jpg	Sabbia di mare	(D29)
1108.jpg	Acqua	(D38)
1109.jpg	Segatura	(D68)
1110.jpg	Raffia	(D84)
1111.jpg	Pelle di cinghiale	(D92)
1112.jpg	Muro di mattoni	(D94)
1123.jpg	Bolle di sapone	(D103)

Tabella 9.7: Descrizione del contenuto di alcune tessiture contenute nel secondo Test–Set.

In Tabella 9.7 sono riportate, accanto ai nomi dei file contenenti i campioni di tessitura, le relative descrizioni e la catalogazione di Brodatz per alcune delle immagini presenti nel Data–Set creato.

Per ognuna delle tessiture presenti nel Data–Set iniziale, ai fini di testare le capacità di recupero dei sistemi QBIC e WindSurf, sono state inserite le versioni

[2]reperibili all'indirizzo: **sipi.usc.edu/services/database/Database.html**

ruotate rispettivamente di 90 e 180 gradi, quelle con contrasto aumentato del 30% e luminosità aumentata dal 20% (identificate nel nome del file dai suffissi r90, r180, C30 e B20 rispettivamente).

Come primo test consideriamo l'immagine rappresentante una corteccia d'albero (1102.jpg) e sottoponiamola ad entrambi i sistemi da analizzare specificando per QBIC l'opzione di query riguardante la tessitura.

Nelle Tabelle 9.8 e 9.9 sono riportati le prime 8 immagini su 12 del risultato dell'interrogazione sul file 1102.jpg sui sistemi QBIC e WindSurf rispettivamente.

In questo caso QBIC recupera solamente 3 immagini, in particolare oltre all'immagine utilizzata nella query viene recuperata l'immagine ruotata di $90°$ rispetto all'originale e quella con il contrasto aumentato del 30% mentre le rimanenti immagini rappresentano erba e bolle di sapone.

Windsurf invece recupera 4 tessiture, oltre a quelle recuperate da QBIC nella stessa interrogazione viene recuperata anche l'immagine ruotata di $180°$ rispetto all'originale e le rimanenti 4 appartengono tutte alla categoria del tessuto di lana.

A conferma dei risultati ottenuti nel test precedente consideriamo l'immagine rappresentante erba (1101.jpg) e sottoponiamola ad entrambi i sistemi da analizzare specificando per QBIC l'opzione di query riguardante la tessitura.

Nelle Tabelle 9.10 e 9.11 sono riportate le prime 8 immagini su 12 del risultato dell'interrogazione sul file 1101.jpg rispettivamente sui sistemi QBIC e WindSurf.

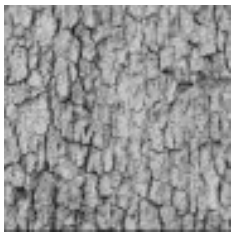
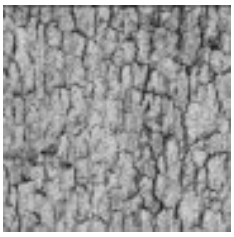
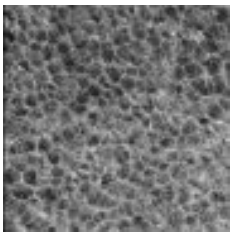
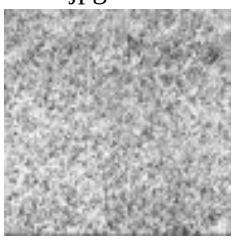
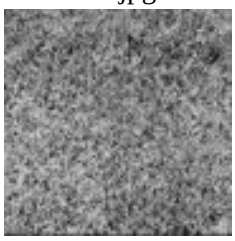
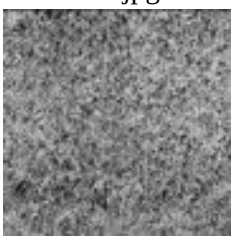
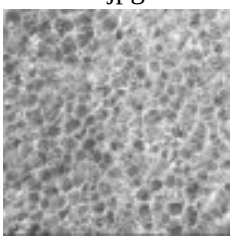
			
1102.jpg	1102B20.jpg	1102r180.jpg	1113r90.jpg
			
1101B20.jpg	1101.jpg	1101r180.jpg	1113B20.jpg

Tabella 9.8: Risultati ottenuti interrogando QBIC sulla texture 1102.jpg.

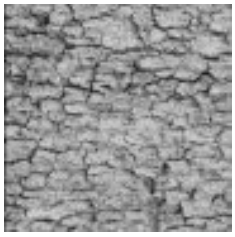
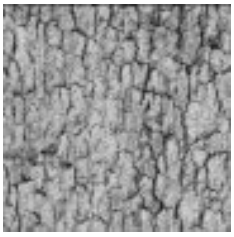
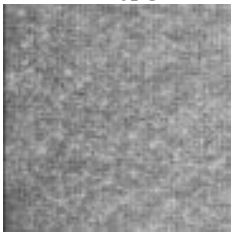
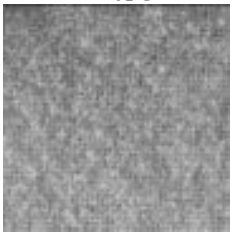
			
1102.jpg	1102r90.jpg	1102r180.jpg	1105C30.jpg
			
1102C30.jpg	1105.jpg	1105r90.jpg	1105r180.jpg

Tabella 9.9: Risultati ottenuti interrogando WindSurf sulla texture 1102.jpg.

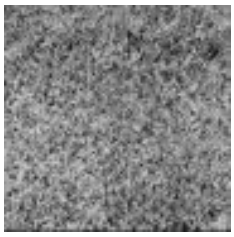
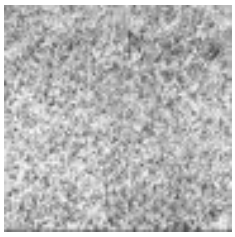
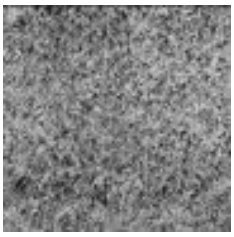
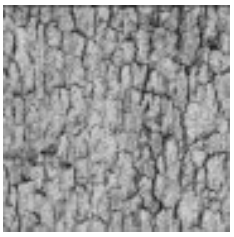
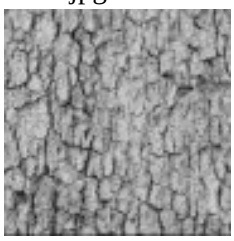
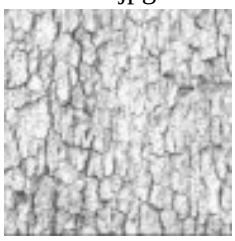
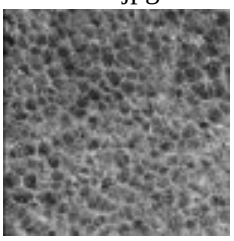
			
1101.jpg	1101B20.jpg	1101r180.jpg	1102r180.jpg
			
1102.jpg	1102B20.jpg	1103C30.jpg	1113r90.jpg

Tabella 9.10: Risultati ottenuti interrogando QBIC sulla texture 1101.jpg.

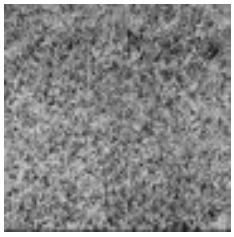
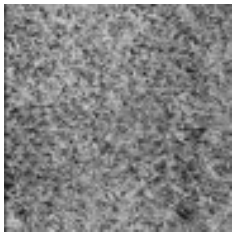
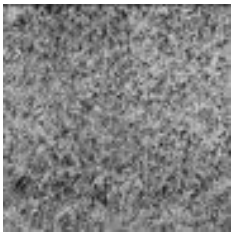
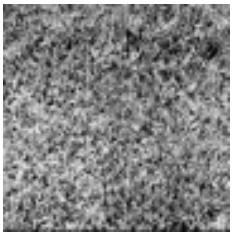
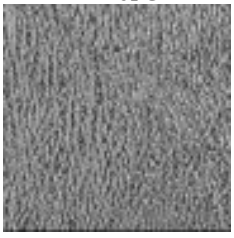
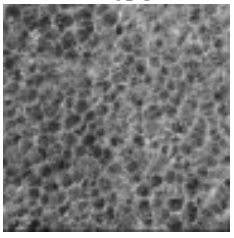
			
1101.jpg	1101r90.jpg	1101r180.jpg	1101C30.jpg
			
1402.jpg	1407.jpg	1106C30.jpg	1113.jpg

Tabella 9.11: Risultati ottenuti interrogando WindSurf sulla texture 1101.jpg.

Anche in questo caso QBIC Tabella 9.10 recupera solo 3 immagini valide, cioè l'immagine originale, quella con la luminosità incrementata del 20% e quella ruotata di 180° rispetto all'originale.

WindSurf, d'altro canto, recupera 4 immagini valide comprendenti entrambe le versioni ruotate e quella con il contrasto incrementato del 30%.

Windsurf in entrambe le interrogazioni recupera sempre le tessiture ruotate evidenziando la proprietà della trasformata Wavelet di essere praticamente invariante rispetto alle rotazioni.

9.5 Conclusioni.

Questa serie di test ci porta a concludere, almeno in prima analisi, la superiorità degli algoritmi di WindSurf rispetto a QBIC nel recupero delle immagini dei Data–Set creati. Ciò determina la superiorità dei sistemi di analisi tempo/frequenza rispetto a quelli basati sugli istogrammi o derivati da essi, giustificando la maggiore complessità delle operazioni di estrazione delle feature e l'aumento proporzionale dei tempi di elaborazione.

Per ciò che riguarda le tessiture invece, la differenza di prestazioni sembra essere minore, ma sempre a favore di WindSurf. Il motivo di questo risultato è da ricercare nell'utilizzo, da parte di QBIC, dell' algoritmo di Tamura, che è molto primitivo (estrazione di un numero ridotto di coefficienti) e datato ('78) e nell'impossibilità da parte di WindSurf di sfruttare le potenzialità dell'algoritmo di estrazione delle regioni, visto che le tessiture sono quasi sempre non clusterizzabili.

Conclusioni

L'utilizzo di strumenti per l'esecuzione di interrogazioni visuali pare la giusta soluzione al problema del reperimento corretto ed efficiente delle immagini con le caratteristiche desiderate.

L'analisi e la sperimentazione dei prototipi software attualmente disponibili per l'esecuzione di interrogazioni visuali hanno dimostrato discrete capacità nel recupero delle immagini desiderate.

In questi prototipi le tecnologie utilizzate nell'esecuzione delle interrogazioni non sono ancora pienamente consolidate, infatti ciascun sistema sviluppato e realmente implementato adotta modalità di recupero uniche con aspetti positivi e negativi diversi. Oltretutto i sistemi analizzati più approfonditamente in questa tesi (QBIC e WindSurf) risolvono il problema in maniera parziale, contenendo ancora molte limitazioni.

La prima osservazione è che QBIC è un prototipo rilasciato da un laboratorio IBM e quindi ha già le caratteristiche di uno strumento software maturo, mentre WindSurf è un prototipo sviluppato da ricercatori ed ha come unico obbiettivo quello di evidenziare metodi innovativi di estrazione di feature.

Al livello di gestione del Database, il sistema adottato da Windsurf non permette l'aggiornamento dinamico delle feature delle immagini memorizzate richiedendo, per l'aggiunta di nuove immagini, la ricostruzione globale del catalogo, operazione molto dispendiosa in termini di tempo.

A livello di feature, quelle estratte da WindSurf sono molto più efficaci di quelle di QBIC nel recupero delle immagini simili, come dimostrato nel capitolo 9, grazie all'introduzione di algoritmi di segmentazione che estraggono versioni locali delle feature rispetto a quelle globali di QBIC.

D'altro canto, le modalità attraverso la quali è stato sviluppato il codice di WindSurf non consentono un adeguato riutilizzo, nelle applicazioni sviluppate dagli utenti, dei miglioramenti ottenuti.

Inoltre in WindSurf non esistono strumenti di indicizzazione che permettano

di migliorare i tempi di recupero quando le dimensioni del Database crescono oltre il migliaio di immagini.

Mentre QBIC fornisce una serie di strumenti che l'utente può integrare nelle proprie applicazione per aggiungervi funzionalità di CBVQ, Windsurf è un'applicazione statica che non permette l'integrazione del proprio codice con quello dell'utente e nemmeno aggiunta di nuove feature.

QBIC, anche se non utilizza le feature più evolute, presenta un ambiente molto ben strutturato per la gestione del database delle immagini e delle relative feature, inoltre un aspetto interessante di QBIC è quello di essere un sistema aperto.

Uno sviluppo futuro di questa tesi potrebbe consistere nell'estrazione del codice di Windsurf, che implementa la trasformata Wavelet e l'estrazione delle regioni, ed integrarlo in QBIC che presenta una migliore organizzazione delle operazioni legate alle estrazione delle feature ad alla memorizzazione delle stesse.