UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria dell'Informazione

Skin Segmentation & Data Visualization in Java

Relatore: **Chiar.mo Prof. Enoch Peserico Stecchini Negri De Salvi**
Correlatore: **Ing. Alberto Silletti**

Laureando: **Maddalena De Biasi**

Anno Accademico: 2011-2012

alla mia nonna...

Indice

Capitolo 1

Introduzione

Il presente elaborato tratta due argomenti principali: la segmentazione della pelle in immagini fotografiche e la visualizzazione di dati medici attraverso grafici. L'esposizione verrà quindi divisa in due parti separate, in cui in ciascuna verranno esposti i problemi riscontrati e le soluzioni proposte.

La prima parte è dedicata alla Segmentazione della pelle nelle immagini. Segmentare un'immagine significa isolare quella parte dell'immagine che ha una determinata proprietà di interesse, in questo caso verranno segmentati i pixel di pelle. Una volta classificati i pixel di un'immagine il modo più rapido di visualizzare il risultato è di costruire una maschera in bianco e nero in cui vengono evidenziati i pixel con la proprietà osservata. Scopo di questa prima parte è quello di ottenere un classificatore semplice, veloce ed efficiente, implementato in Java, che come viene spiegato nel secondo capitolo, è la base del pacchetto software *fbs*. L'*fbs* fornisce gli elementi di elaborazione per un macchinario, il Full Body Scanner, proposto dal team Naevi in Silico, il cui prototipo è attualmente in fase di sviluppo all'Università di Padova e di cui viene mostrato il principio di funzionamento sempre nel secondo capitolo.
Il terzo capitolo riguarda lo stato dell'arte nella segmentazione della pelle nelle immagini: verranno presentati i lavori considerati e le motivazioni per cui sono stati scelti o scartati.
Il quarto capitolo fornisce i dettagli dell'implementazione dell'algoritmo di segmentazione, mentre nel quinto vengono esposte le conclusioni e i possibili miglioramenti futuri.

Nella seconda parte dell'elaborato verrà affrontata la visualizzazione di dati medici. A tale scopo è stata implementata una classe Java che permette di creare grafici per visualizzare questi dati. Nel sesto capitolo vengono presentate le motivazioni del progetto, nel settimo i dettagli della classe.

L'ottavo capitolo fornisce la spiegazione del wrapper JFreeChart, il software open source utilizzato come base per il lavoro svolto; nel nono capitolo sono esposte le conclusioni e i lavori futuri nell'ambito della gestione dei dati.

Capitolo 2

Full Body Scanner

Negli ultimi anni é stata osservata una sempre maggiore incidenza di lesioni cutanee dovute a melanomi. Il melanoma è, come ben noto, una forma tumorale aggressiva che causa la morte in molti dei casi. La prevenzione e l'analisi precoce per questo tipo di tumore sono fondamentali.
In un esame dermatologico atto all'individuazione di nevi sospetti, la prima fase consiste nella mappatura dei nevi presenti sulla pelle del paziente. Ció permette al medico di avere una prima visione globale della situazione e di concentrarsi successivamente sul caso specifico del singolo nevo. Questa prima parte dura all'incirca una decina di minuti per paziente, rappresentando un costo abbastanza elevato in termini di tempo utilizzato per la visita. L'idea é quindi quella di ridurre questa tempistica con l'utilizzo di un macchinario che, in automatico, svolga la mappatura dei nevi del paziente.

Il Full Body Scanner è un macchinario, formato da una parte hardware e una software, che serve come supporto dermatologico per la mappatura automatica dei nevi.
Nasce dalle necessità esposte in precedenza e ha l'obiettivo primario di abbreviare i tempi per la mappatura a circa un minuto a paziente, mantenendo però una qualità nella mappatura dei nevi molto simile a quella ottenuta dall' esame manuale da parte del medico.
La parte hardware del macchinario è composta da tre elementi principali:

- due aste in ferro, su cui su ognuna sono montate tre fotocamere digtali reflex Nikon D5000;
- due bluescreen con relative aperture per le fotocamere;

- un monitor per la visualizzazione delle immagini e dell'interfaccia grafica del software.

Il pacchetto software consiste nell'*fbs*. È scritto in due linguaggi di programmazione principali:

- java, soprattutto la parte riguardante il processing delle immagini fotografiche
- C++, per la parte di interfacciamento con l'hardware.

L'fbs permette l'interfacciamento con le fotocamere, l'elaborazione delle immagini e fornisce l'interfaccia grafica indispensabile al medico per utilizzare il macchinario.
Il primo step consiste nella mappatura dell'intera superfice corporea, che avviene scattando una serie di sei fotografie al paziente, tre nel lato front e tre nel lato back, in due posizioni standard. Ottenuto il posizionamento della superficie corporea nell'immagine, questa verrà nuovamente processata per ottenere la mappatura dei nevi.

Le due posizioni standard assunte dal paziente sono studiate in modo tale che l'area analizzata sia la più vasta possibile, in quanto ci sono zone che non possono essere fotografate comodamente, come gli spazi interdigitali e le piante dei piedi.

La presenza del bluescreen, come si vedà in seguito, è fondamentale per la mappatura della superficie corporea analizzabile, data dal metodo di segmentazione della pelle.

Un'altra applicazione utile fornita dal software è il database dello storico dei pazienti. Le visite precedenti di ogni paziente sono conservate in memoria e permettono di avere lo sviluppo nel tempo dello screening e il confronto con la situazione attuale. Ciò è molto utile per quanto riguarda il monitoraggio della comparsa di nuovi nevi, che tendenzialmente risultano essere pericolosi.

Capitolo 3

Skin Segmentation

La segmentazione della pelle è il primo, fondamentale passo nel processing delle immagini. Permette l'individuazione dell'area corporea del paziente, su cui verranno effettuate le successive elaborazioni. Risulta essere quindi un algoritmo fondamentale per il software fbs, e deve essere indispensabilmente real time. Segmentare un'area vuol dire classificare ogni pixel come avente o no una data proprietà, in questo caso significa determinare se è un pixel di pelle o di background.

La skin detection è utilizzata in molti campi di processing di immagini mediche e non, ad esempio per la face detection e la hand detection, per l'individuazione di immagini pornografiche su internet e altro ancora.
Il passo decisionale degli algoritmi permette la creazione di una maschera, solitamente in bianco e nero, dell'immagine in cui vengono evidenziati i pixel di interesse, isolandoli dal background.

3.1 State of the Art

Si possono trovare in letteratura svariati classificatori, che differiscono tra loro in maniera significativa. Si è tentato di trovare un metodo di categorizzazione generale, mettendo in luce gli elementi in comune ai vari algoritmi. Per ogni classe verranno fatti esempi esplicativi, scelti in base alle considerazioni effettuate per la decisione dell'algoritmo di base da elaborare e implementare nel software fbs. I vari metodi di classificazione possono essere suddivisi in due principali categorie [1]:

- pixel based, detti anche metodi colour based;
- region based, o metodi texture based.

Dei primi fanno parte tutti quei metodi che classificano ogni pixel come *pelle* o *non pelle* in base alle sole informazioni contenute all'interno, indipendentemente dai pixel presenti nell'intorno.
Dei secondi invece fanno parte tutti quei metodi che utilizzano le informazioni sulla posizione del pixel, quindi in base alle informazioni dettate anche dai pixel vicini.

3.2 Pixel Based skin detection

La computer grafica e le regole di visualizzazione digitale, che definiscono l'immagine come elemento bitmap, hanno permesso la comparsa di differenti colourspace.
Un colourspace è un un modello matematico astratto che permette la rappresentazione del colore attraverso tuple numeriche (in particolare 3 per l'RGB). La classificazione pixel based si basa sulle informazioni date dal colore del pixel, e quindi sull'elaborazione della tupla che lo definisce. I differenti classificatori pixel based presenti in letteratura, utilizzano vari colourspace, rispetto alle informazioni che vogliono ricavare dal colore del pixel (ad esempio la luminosità, il tono di rosso, l'intensità cromatica, ...) I classificatori pixel based, possono essere suddivisi a loro volta a seconda del modello di colore della pelle utilizzato.
Essi vengono suddivisi in metodi:

- a limiti definiti;
- a modello non parametrico;
- a modello parametrico;

Di seguito sono riportati le principali caratteristiche e degli esempi di ciascuno.

3.2.1 Explicitly Defined skin region

Sono metodi che definiscono i limiti della regione di pelle nel colourspace, questi sono prefissati e dettati da valori di soglia. La regola di decisione che viene adottata è molto semplice e uguale per ogni immagine presa in considerazione. L'algoritmo maggiormente citato in letteratura che fa parte di questa classe è l'algoritmo sviluppato da Peer et al [2]. L'algoritmo è stato ideato per essere utilizzato nell'ambito della face detection in occasione dell'installazione "15 Seconds of Fame", in cui venivano riportati all'interno di una cornice su un monitor LCD, per 15 secondi, i volti di chi visionava la

mostra. L'algoritmo utilizza il 3D colourspace (RGB) per determinare quali sono i pixel di pelle, basandosi su regole euristiche.
Viene così definito:

% the skin color at uniform daylight illumination ⇒

$R > 95$ AND $G > 40$ AND $B > 20$ AND
$max\{R, G, B\} - min\{R, G, B\} > 15$ AND

% RGB components must not be close together-
% grayness elimination

$|R - G| > 15$ AND

% also R and G components must not be close together
% otherwise we are not dealing with the fair complexion

$R > G$ AND $R > B$ % R component must be the gratest component

OR

% The skin color under flashlight or (light) daylight
% lateral illumination ⇒

$R > 220$ AND $G > 210$ AND $B > 170$ AND

$|R - G| \leq 15$ AND
% R and G components must be close together
$R > B$ AND $G > B$
% B component must be the smallest component

I limiti principali di quest'algoritmo sono:

- Il fatto di essere molto dipendente dal modello di pelle, stabilito a priori: l'algoritmo è impostato per una tipologia di carnagione chiara, quindi non è applicabile ad ogni persona. I valori di soglia sono empirici e calcolati per questo tipo di carnagione.
- Utilizzando il colourspace RGB, risulta essere dipendente anche dai valori della luminosità del pixel. Sebbene vengano prese in considerazioni due differenti tipologie di luminosità, queste non sono sufficienti a rendere l'algoritmo generale.

Il vantaggio è rappresentato dalla linearità: ogni pixel viene al più considerato un'unica volta e questo permette di avere una dipendenza lineare dalla dimensione dell'immagine. Inoltre, la tipologia di carnagione che viene presa in analisi, non rappresenta un limite, in quanto una mappatura automatica accurata dei nevi può essere effettuata solo su una carnagione in cui i nevi (scuri) sono in risalto, e quindi solo per pelli chiare.

La determinazione del colore della pelle è il primo passo fondamentale dal quale dipendono anche i risultati dei successivi. Per valori costanti e noti di luminosità, è stato dimostrato che complicare il modello, rendendolo indipendente da questa, non migliora in modo apprezzabile i risultati, ma peggiora le prestazioni dell'algoritmo.

3.2.2 Non Parametric skin distribution modeling

Sono una classe di metodi, la cui idea di base è di non avere un modello aprioristico del colore della pelle, bensì di stimarene una distribuzione probabilistica in base ai risultati ottenuti su dati di training. Il modello di colore della pelle è ottenuto quantizzando i colori di un'immagine (solitamente nel piano della crominanza); ogni valore quantizzato rappresenta un range di colori.

I pixel delle mmagini di training, rappresentanti pelle, vengono poi analizzati e viene creato un istogramma conteggiando quante volte un determinato colore si presenta. Infine i risultati ottenuti nell'istogramma vengono normalizzati in modo da ottenere la probabilità che un determinato colore sia di pelle.

$$Pskin(c) = \frac{skin[c]}{Norm}$$

dove skin[c] è il valore del conteggio e Norm è il coefficientte di normalizzazione (che a seconda del metodo può essere la somma totale dei valori presenti nell'istogramma, o il massimo valore presente).
In questo modo si possono implementare semplicemente criteri di verosimiglianza e, attraverso metodi empirici, ottenere valori di soglia accettabili.

La quantizzazione gioca un ruolo importante in quanto deve essere sufficientemente fine da non permettere che due colori distanti tra loro si ritrovino codificati con il medesimo valore, ma allo stesso tempo non deve essere trop-

po fitta, altrimenti la quantità di spazio richiesto per la memorizzazione delle informazioni risulta eccessivo.

Esempi di classificatori non parametrici si possono trovare anche tra quelli che utilizzano la regola di Bayes[4]:

$$P(skin|c) = \frac{P(c|skin)P(skin)}{P(c|skin)P(skin) + (c|\neg skin)P(\neg skin)}$$

Per classificare un pixel come pelle o non pelle si ha dunque bisogno della probabilità che il colore del pixel sia pelle, alla condizione che sia di un colore determinato.
Osservando più attentamente la costruzione dell'istogramma presentato in precedenza si può notare che la probabilità che si ottiene in effetti è la probabilità condizionata $P(c|skin)$.
Utilizzando lo stesso metodo si può ottenere per ogni colore anche $P(c|\neg skin)$ semplicemente prendendo come training immagini che rappresentano background possibili. In questo modo si tiene presente anche il fatto che un qualsiasi colore, anche se ricade nel range "rosa", potrebbe avere probabilità non nulla di non essere pelle.
Computando opportunamente la regola di Bayes si ottiene:

$$\frac{P(skin|c)}{P(\neg skin|c)} = \frac{P(c|skin)P(skin)}{P(c|\neg skin)P(\neg skin)}$$

e utilizzando come regola di decisione $P(skin|c) \geq \Theta$ si ottiene:

$$\frac{P(c|skin)}{P(c|\neg skin)} > \Theta$$

dove Θ rappresenta il valore di soglia, dipendente da P(skin) a meno di una costante moltiplicativa.

Si elencano in seguito i pregi dell'implementazione di metodi non parametrici per la skin detection:

1. dopo la fase di training sono molto veloci nell'uso;

2. sono indipendenti dalla distribuzione probabilistica del colore della pelle, ciò vuol dire che, utilizzando diversi tipi di immagini di training, possono essere prese in considerazioni diversi tipi di carnagione senza variare l'algoritmo.

3. tengono conto non solo della probabilità che un determinato colore sia di pelle, ma anche della probabilità che non lo sia.
   Questo implica, attraverso la scelta di un opportuno valore di soglia Θ una maggiore robustezza rispetto ai falsi positivi presenti nel background.

Gli svantaggi sono invece invece rappresentati da:

1. lo spazio di memoria richiesto per l'utilizzo è molto vasto, soprattutto se si utilizza una quantizzazione efficace;

2. l'incapacità di interpolare o generalizzare i dati di training (cioè i risultati risultano dipendenti dalla rappresentatività delle immagini utilizzate per il training).

3.2.3 Parametric skin distribution modeling

Questa classe utilizza modelli per il colore della pelle che dipendono da dei parametri probabilistici [5]. L'idea nasce dal rendere i metodi non parametrici indipendenti dalle immagini di training e dal ridurre lo spazio di memoria necessario per l'implementazione.
In questi modelli la distribuzione probabilistica (pdf) del colore della pelle può essere modellata come una normale Gaussiana. Viene definita come:

$$p(c|skin) = \frac{1}{2\pi|\Sigma_s|^{\frac{1}{2}}} e^{-\frac{1}{2}(c-\mu_s)^T \Sigma_s (c-\mu_s)}$$

dove c è il vettore dei colori e μ_s e Σ_s sono i parametri distribuiti, ottenuti dai dati di training.

Questo tipo di algoritmo ha lo svantaggio di essere maggiormente dipendente dal colorspace rispetto a metodi non parametrici. Ha però il vantaggio di ridurre notevolmente lo spazio di memoria utilizzato nell'implementazione. Rispetto agli scopi iniziali riguardanti l'indipendenza dalle immagini

di training, si è alla fine ottenuta solo un'indipendenza parziale, in quanto le immagini di training sono necessarie comunque per ricavare i parametri. Questi però tengono anche presente le possibil inesattezze e variabilità che si possono avere.

3.2.4 Dynamic skin distribution models

Questa classe di metodi si basa sulla considerazione che non esiste, in realtà un metodo di classificazione che possa essere considerato ottimo, perchè questo può variare in base alle differenti condizioni di luce, di carnagione, di fotocamera, di background, ...
È possibile valutare caso per caso quale sia il metodo di classificazione migliore, implementando un algoritmo che sia tempo variante.
Per ideare algoritmi efficienti bisogna mediare due aspetti:

- la velocità di training e classificazione;
- l'abilità di adattamento alle nuove condizioni.

Mediando questi due aspetti è possibile ottenere algoritmi che hanno un false positive rate inferiore rispetto ai modelli generali. Lo svantaggio principale dovuto a questo tipo di implementazione è l'aumento dei tempi di esecuzione, anche se mediato da un maggiore indice di correttezza.
Per applicazioni real time un approccio di questo tipo è perciò sconsigliabile.

3.3 Region Based Skin Detection

I metodi di classificazione region based utilizzano le informazioni prese dalla disposizione spaziale dei pixel della pelle durante la fase di rilevamento per migliorare le prestazioni.
Sono necessarie ulteriori conoscenze sulla texture della pelle per l'applicazione di questi metodi.

Sono generalmente utilizzati come supporto ai metodi pixel based, cioè l'uscita di un metodo pixel based è utilizzata come ingresso per un metodo region based.

Alcuni esempi di metodi region based si possono trovare nell'utilizzo di fuzzy system e di reti neurali. Non viene presentata una trattazione approfondita di questa tipologia di skin segmentation in quanto non sono stati presi in considerazione per il lavoro svolto.

Capitolo 4

Peer Algorithm Implementation

Per implementare un'applicazione da inserire nel software fbs, i principali obiettivi riguardano la solidità e i tempi di esecuzione approssimabili a real time.
In questo capitolo verranno descritti i dettagli dell'implementazione dell'algoritmo di skin segmentation e delle classi presenti nel package fbs.peer.
Per implementare l'algoritmo di segmentazione della pelle la scelta è ricaduta sull'algoritmo descritto da Peer et al. [2] per i seguenti motivi:

- perchè tra tutti gli algoritmi analizzati ha il pregio di avere un'implementazione real time: ogni pixel viene al massimo preso in considerazione una volta sola.
- non richiede spazio di memoria per il salvataggio del modello di pelle utilizzato.
- le foto vengono scattate in condizioni di ambulatorio, quindi di spazio indoor che è illuminato da luci artificiali; le specifiche dell'algoritmo sulla robustezza all'illuminazione dell'immagine possono quindi esserre rilassate.

L'algoritmo di Peer fornisce i limiti della skin region nel colurspace RGB.
In Java un'immagine è rappresentata come una matrice, ogni elemento della matrice rappresenta un pixel. I valori RGB di un pixel vengono rappresentati con un intero (36 bit). I valori dei canali R, G e B vanno da 0 a 255, quindi sono necessari 8 bit ciascuno per rappresentarene il valore($2^8 = 256$).
Per il modello RGB nella rappresentazione a intero sono utilizzati i 24 bit

meno significativi, i rimanenti 8 significativi sono dedicati al canale α alpha, che indica l'opacità del pixel e che non verrà preso in considerazione nella trattazione seguente. Le reflex digitali Nikon D5000 montate nel Full Body Scanner hanno un sensore CMOS da 12.9 MPixel. Vengono scattate tre fotografie per lato del paziente, che vengono successivamente composte tre a tre in modo da formare un'immagine total body. La dimensione dell'immagine risultante non è prefissata, ma ha delle variazioni che dipendono da vari fattori, come ad esempio l'esattezza della posizione assunta dal paziente.
In generale l'immagine risultante avrà dimensioni che sono in un intorno di 4600 x 8000 pixel (circa 36 MPixel). I tempi di esecuzione devono essere molto brevi, intorno al secondo, per riuscire a ottenere l'obiettivo prefissato di un minuto a paziente per la mappatura completa dei nevi. Data la dimensione delle immagini processate, questa specifica è abbastanza restrittiva.

4.0.1 Implementazione dell'algoritmo

L'algoritmo è stato implementato con un metodo statico della classe PeerClassifier:

```
public class PeerClassifier {
    public static int[] classifier(int width,
                                   int height,
                                   int[][] rgb)
            {...}//corpo del metodo
}
```

I parametri richiesti dal metodo sono

- la larghezza dell'immagine da segmentare
- l'altezza dell'immagine da segmentare
- la matrice di interi che rappresenta il colourspace RGB.

Ogni pixel viene considerato una sola volta, ottenendo quindi un'implementazione O(n) del metodo, dove n è il numero di pixel dell'immagine.
La maschera ottenuta classifica i pixel "pelle" con il colore bianco, mentre i pixel di background sono neri.
Vengono di seguito riportati alcuni risultati ottenuti con l'applicazione del metodo di classificazione.

(a) Immagine Originale

(b) Risultato del PeerClassifier

Figura 4.1: Esempio 1

(a) Immagine Originale (b) Risultato del PeerClassifier

Figura 4.2: Esempio 2

Anche se i risultati possono a prima vista sembrare soddisfacenti, da queste figure si possono notare i principali problemi dovuti all'applicazione del solo algoritmo di Peer:

- In primo luogo molti sono i falsi positivi; se sul background ci sono aree che hanno colorazione simile alla pelle umana esse verranno classificate come pelle. Ad esempio nell' esempio 1 la parte più illuminata dello stipite presente sulla destra dell'immagine o nell'esempio 2 il pavimento della stanza.

- In secondo luogo, l'area di pelle non è omogenea, cioè ci sono dei "buchi", che dovrebbero essere classificati come pelle, ma a causa di ombre e di non uniformità del colore (come macchie più scure o peli) non vengono classificati nella giusta maniera.

- Infine i contorni dell'area di pelle non sono sufficientemente uniformi e connessi, il che è un problema per le successive analisi dell'immagine.

Altri algoritmi, diversi da quello di Peer potrebbero portare un miglioramento decisivo al primo problema. Ad esempio utilizzando un algoritmo non parametrico, il numero di falsi positivi si ridurrebbe drasticamente. Purtroppo lo spazio in memoria e i tempi di esecuzione richiesti da un algoritmo di questo tipo sono inaccettabili per gli scopi prefissati.

La soluzione del problema (già anticipata nell'elenco delle risorse hardware del Full Body Scanner) è stata quella di mettere dei bluescreen come sfondo per le immagini. In questo modo si ha un netto contrasto tra il colore della pelle e il background, e il rischio di falsi positivi lontani dalla figura del paziente in condizioni corrette di scatto è nullo.
Per il secondo e terzo problema la soluzione adottata è la stessa ed è la seguente: vengono utilizzati dei metodi di erosione e di dilatazione sull'immagine, in modo da chiudere i buchi e rendere più omogenei i contorni dell'area segmentata.

4.0.2 Metodi di Dilatazione e Erosione

Per eliminare le non uniformità della pelle vengono utilizzati in combinazione i metodi di dilatazione e di erosione. Una dilatazione seguita da una erosione è detta operazione di chiusura. Esiste anche l'operazione di apertura che consiste in una erosione seguita da una dilatazione. Erosione, dilatazione, apertura e chiusura sono detti anche operatori morfologici.
La morfologia matematica è una disciplina che si è sviluppata nell'ambito dell'elaborazione delle immagini, con l'obiettivo di estrarre informazioni topologiche da un'immagine binaria. Detta A l'immagine binaria e B un'immagine binaria più piccola o maschera possiamo definire:

- **EROSIONE** di A da parte di B :

$$A \ominus B = \{h \in E | b_h \subseteq A\}$$

  dove B_h rappresenta la traslazione dell'immagine B sul punto h $\in$ A:

$$B_h = \{b + h | b \in h\}$$

  in pratica l'elemento strutturante B scorre su ogni punto h dell'immagine, eliminando gli elementi che non lo contengono interamente.

- **DILATAZIONE** di A da parte di B:

$$A \oplus B = \{h \in E | B_h \cap A \neq \emptyset\}$$

  Cioè $A \oplus B$ è l'insieme di tutti i punti di posizione h, per cui B_h ed A hanno almeno un punto in comune. In pratica l'elemento strutturante di B scorre sull'immagine di riferimento A estendendo il dominio di quest'ultima alle porzioni d'immagine che hanno almeno un punto in comune con A.

- **APERTURA** di A da parte di B:

$$A \circ B = (A \ominus B) \oplus B$$

- **CHIUSURA** di A da parte di B:

$$A \bullet B = (A \oplus B) \ominus B$$

Le operazioni di dilatazione e erosione in generale non sono reversibili.
Come maschera B sono state prese in considerazione due differenti tipologie: una romboidale e una circolare.
Il raggio della maschera è regolato attraverso un parametro statico.

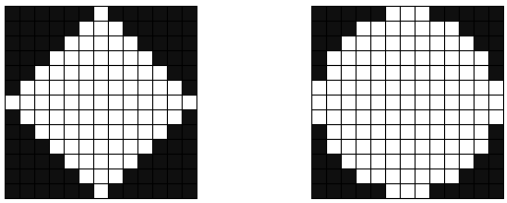


Figura 4.3: esempio di maschera romboidale e circolare con raggio 7

L'implementazione maggiormente efficiente dei metodi di dilatazione e erosione trovata è quella che prevede l'utilizzo dell'algoritmo Manhattan.

4.0.3 Manhattan Distance

La **Manhattan Distance** è una metrica per cui la distanza tra due punti è la somma tra i valori assoluti delle loro coordinate. È conosciuta anche come distanza rettilinea.

Ad esempio la distanza tra i punti P_1 e P_2 di coordinate (x_1, y_1) e (x_2, y_2) è :

$$D_{Manhattan} = |x_1 - x_2| + |y_1 - y_2|$$

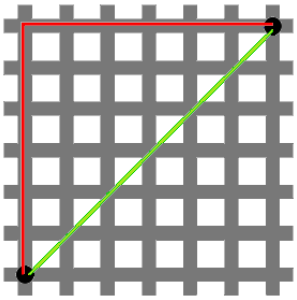
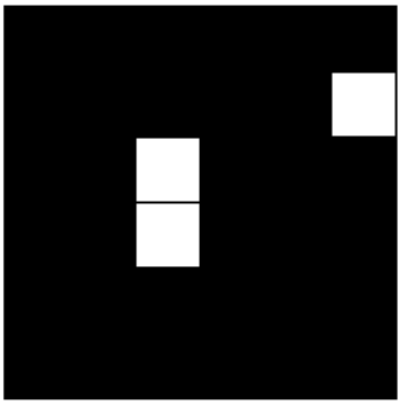


Figura 4.4: la linea rossa rappresenta la manhattan distance tra i punti P_1 e P_2, la linea verde invece la distanza euclidea.

Utilizzando l'algoritmo di distanza Manhattan su un'immagine bidimensionale è possibile ottenere una mappa di distanza. La mappa di distanza è data da una matrice in cui ogni elemento rappresena la distanza (Manhattan) dal pixel bianco più vicino (figura 4.5).



(a) Immagine binaria in input

4	3	2	3	2	1
3	2	1	2	1	0
2	1	0	1	2	1
2	1	0	1	2	2
3	2	1	2	3	3
4	3	2	3	4	4

(b) Mappa di distanza Manhattan

Figura 4.5:

la mappa di distanza manhattan è implementata in maniera semplice:

- per prima cosa è stato necessario implementare due diversi metodi per calcolare la mappa di distanza, uno per quella utilizzata per l'erosione e uno per quella utilizzata nella dilatazione. Infatti, nella prima viene calcolata la mappa di distanza per i pixel neri, nella seconda quella per i pixel bianchi.

- sono sufficienti due attraversamenti dell'immagine per il calcolo della singola mappa. Uno dall'angolo in alto a sinistra all'angolo in basso a destra per righe, e uno dall'angolo in basso a destra all'angolo in alto a sinistra per colonne.

I metodi di erosione e di dilatazione si possono ottenere semplicemente confrontando il valore contenuto negli elementi della mappa di distanza con il valore del raggio che si vuole dare alla dilatazione/erosione. In questo modo si ottiene una maschera B romboidale.
I metodi sono metodi statici della classe MorphologicalOperators:

```
public static void erode(int width, int height,int[][] image){
     //corpo del metodo
}

public static void dilate(int width, int height,int[][] image){
     //corpo del metodo
}

public static void close(int width, int height,int[][] image){
     //corpo del metodo
}

public static void dilate(int width, int height,int[][] image){
     //corpo del metodo
}
```

I parametri richiesti sono la larghezza e l'altezza dell'immagine da analizzare, e l'immagine binaria su cui eseguire le operazioni. Per l'apertura e la chiusura l'immagine è l'uscita del metodo di classificazione. I ritorni sono void in quanto viene modificata l'immagine data in ingresso.

I risultati ottenuti utilizzando un'implementazione di questo tipo per i metodi di dilatazione e di erosione sono molto buoni per quanto riguarda i tempi di esecuzione: il tempo per l'applicazione dei metodi di classificazione e chiusura rimane nei limiti di ciò che avevamo prestabilito, cioè circa un secondo. Non sono invece sufficienti per quanto riguarda l'ottenimento di contorni omogenei, come è possibile notare dalla figura 4.6.
Per la chiusura dell'immagine si devono utilizzare raggi per la maschera molto grandi, i bordi della regione di pelle anche se risultano molto più connessi rispetto alla classificazione originale sono seghettati; ciò è un problema in quanto aumenta il numero di falsi positivi.

Figura 4.6: chiusura con distanza manhattan

4.0.4 Euclidean Distance

La soluzione trovata per migliorare ulteriormente i risultati ottenuti è stata quella di utilizzare una maschera B circolare. Per modificare la mappa di distanza per ottenere il tipo di maschera richiesto è stato necessario utilizzare, invece della distanza Manhattan, la distanza Euclidea.
La mappa di distanza euclidea è implementata in maniera più complessa:

- Sono necessari quattro attraversamenti dell'immagine per calcolare una singola mappa:
  - dall'angolo in alto a sinistra a quello in basso a destra per righe;
  - dall'angolo in alto a destra a quello in basso a sinistra per righe;
  - dall'angolo in basso a sinistra a quello in alto a destra per colonne;
  - dall'angolo in basso a destra a quello in alto a sinistra per colonne.
- Inoltre al semplice calcolo della distanza manhattan (una somma), è sostituito il calcolo più elaborato della distanza euclidea, che richiede un numero maggiore di operazioni.

Figura 4.7: chiusura con distanza euclidea

- Anche il numero di confronti tra i valori contenuti che vengono effettuati per ogni attraversamento è maggiore.

I metodi sono stati implementati come metodi statici della classe MorphOpEuclidean. Richiedono un parametro in ingresso in più rispetto ai loro analoghi implementati con la Manhattan distance. Il parametro in ingresso aggiuntivo è una matrice delle stesse dimensioni dell'immagine da analizzare, utilizzata come supporto per le informazioni necessarie per il calcolo della distanza dal pixel più vicino da inserire nella mappa.

Un esempio dei risultati ottenuti utilizzando la mappa di distanza euclidea si può trovare in figura 4.7.

Dai risultati ottenuti si può osservare che i contorni sono meglio definiti e meno seghettati. Inoltre per ottenere la stessa efficacia nella chiusura dei buchi nell'immagine è possibile utilizzare un raggio minore.
Il metodo risulta però molto più lento a causa del maggior numero di operazioni che è necessario effettuare sull'immagine.

Capitolo 5

Conclusioni e future works

Non è possibile determinare in assoluto quale sia il migliore dei metodi di apertura e chiusura proposti; sia il metodo che utilizza la Manhattan distance che il metodo che utilizza la Euclidean distance hanno entrambi pregi e difetti.

Il primo è molto veloce e permette di rispettare le tempistiche che si sono fissate come obiettivo, però ha lo svantaggio di non avere una definizione dell'immagine soddisfacente.

Il secondo, invece, peggiora il tempo di esecuzione a favore di una maggiore definizione dell'immagine: i contorni dell'area di pelle sono migliori e la figura risultante è più congruente alla reale area corporea.
Entrambi i metodi sono stati inclusi nel software fbs e fanno parte del pacchetto fbs.peer.

Un miglioramento dei metodi proposti è possibile soprattutto per quanto riguarda la resa dell'immagine finale.
I contorni dell'immagine sono ancora abbastanza irregolari. Per ottenere maggiore regolarità si dovrebbe costruire un'algoritmo di interpolazione dei contorni e di definizione delle aree corporee, in modo da determinare quali regioni di pixel siano connesse e eliminare dalla classificazione i pixel al di fuori di esse.
La modifica proposta potrebbe, però, portare a prestazioni temporali peggiori di quelle ottenute utilizzando una chiusura.

Capitolo 6

Data Visualization

La seconda parte di questa trattazione è dedicata alla visualizzazione dei dati.
Si è riscontrata una mancanza nella applicazioni open source nel fornire software che fosse di immediato utilizzo per tale scopo.
In molte applicazioni sviluppate nell'ambito sanitario, sono indispensabili dei grafici, per permettere la visione da parte del medico dei dati. Questi, oltre a fornirene una visualizzazione chiara e precisa, dovrebbero avere anche un aspetto interessante dal punto di vista del layout.

L'idea di implementare del software che permettesse di creare grafici di questo tipo è nata dalla suddetta esigenza. È stata così ideata una classe in Java che permettesse la creazione veloce di grafici atti alla visualizzazione di dati medici.

Per implementare questa classe è stata utilizzata la libreria grafica JFreeChart, che permette la creazione e la personalizzazione di molte tipologie di grafici. JFreeChart è una libreria open source che verrà spiegata nel dettaglio nel capitolo 8.

Capitolo 7

LineChart

7.1 Caratteristiche

Con la classe LineChart è possibile creare grafici interattivi.
È stata ideata principalmente per la visualizzazione di dati che hanno uno sviluppo temporale.
Permette la creazione di due differenti tipologie di grafici:

- uno temporale: che ha come riferimento sull'asse delle ascisse le date, con cui è possibile visualizzare l'andamento nel tempo dei dati osservati;
- uno numerico: vengono visualizzati una collezione di dati numerici di tipo (x_i, y_i).

Il grafico è interattivo nel senso che, se inserito all'interno di un Frame, ha le seguenti funzionalità:

- selezionando un'area, questa viene automaticamente ingrandita;
- la possibilità di reimpostare lo zoom ai valori iniziali;
- la presenza di tooltip che evidenziano i dati inseriti;
- la visualizzazione del mirino per un particolare dato;

7.2 API

LineChart è una classe che estende JPanel e che permette la creazione di un grafico interattivo a interpolazione lineare dei dati.

```
java.lang.Object
    java.awt.Component
           java.awt.Container
                  javax.swing.JComponent
                         javax.swing.JPanel
                                LineChart
```

La classe è fornita di due metodi costruttori:

```
public HistoricalChart(java.lang.String name,
                       int width,
                       int height,
                       boolean isDate)
```

questo metodo costruttore ha come parametri di ingresso:

name - l'intestazione del grafico;
width - la larghezza del grafico;
height - l'altezza del grafico;
isDate - valore booleano che permette di creare un grafico
dove l'asse delle ascisse e' temporale se impostato a true,
l'asse delle acisse e' numerico se impostato a false.

il secondo costruttore differisce dal primo solo per il parametro name, che non è presente, in quanto l'intestazione del grafico viene impostata di default a "Historical Line Chart".

Questa classe definisce undici metodi:

```
public void createDataset(long[] xValue, double[] yValue)
Throws:
      java.util.zip.DataFormatException
```

Questo metodo serve per creare il dataset del grafico, cioè l'insieme di dati da visualizzare. Ha due parametri di ingresso:

[ ]xValue - array che rappresenta i valori delle ascisse;
[ ]yValue - array che rappresenta i valori delle ordinate.

I punti inseriti sono $(xValue_i, yValue_i)$. Se i due array hanno lunghezze diverse allora il metodo lancia un'eccezione.

```
public void newValue(long xValue,
                     double yValue)
```

Questo metodo permette di inserire un nuovo valore nel dataset, anche se questo è vuoto.
Il valore inserito sarà $(xValue, yValue)$.

```
public void setXLabel(java.lang.String xLabel)
```

```
public void setYLabel(java.lang.String yLabel)
```

Con questi due metodi è possibile impostare le label dell'asse delle ascisse e delle ordinate.

```
public void changeBackground(java.awt.Paint p)
```

```
public void changeBackground(Image im)
```

Con questi due metodi è possibile reimpostare il colore dello sfondo, o inserire un'immagine.

```
public void changeLinePaint(java.awt.Paint p)
```

Cambia il colore della linea di intersezione.

```
public void setRange(double lower,
                     double upper)
```

Permette lo zoom del grafico in un determinato range di valori [lower, upper].

```
public void saveAsJPEG(java.io.File file,
                       int width,
                       int height)
                throws java.io.IOException

    Throws:
        java.io.IOException


ublic void saveAsPNG(java.io.File file,
                     int width,
                     int height)
              throws java.io.IOException

    Throws:
        java.io.IOException
```

Salvano il grafico rispettivamente in formato JPEG o PNG, con altezza height e larghezza width, nel file specificato.
Viene lanciata un'eccezione nel caso che il file non esista.

```
public void defaultRange()
```

Riporta lo zoom al valore di default.

```
public void restoreDefaults()
```

Reimposta il colore dello sfondo, della linea e lo zoom ai valoro di default.

7.3 Examples

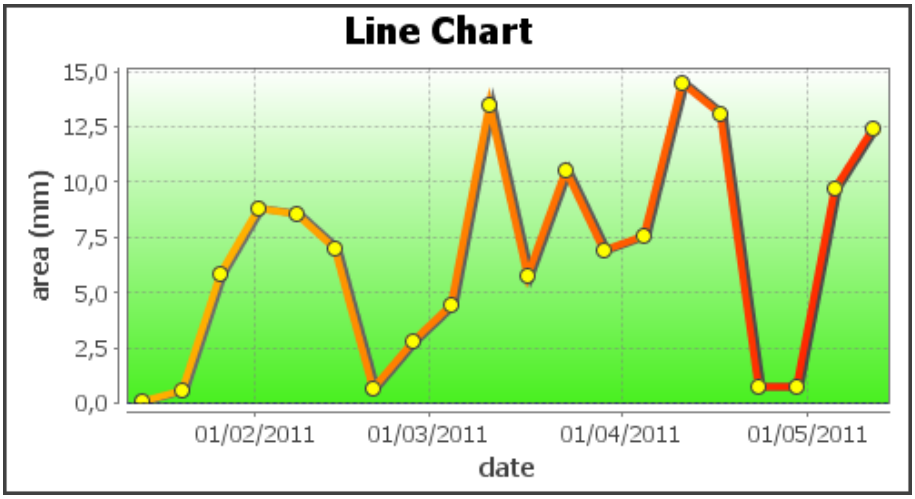


Figura 7.1: esempio di grafico temporale, salvato con il metodo saveAsPNG

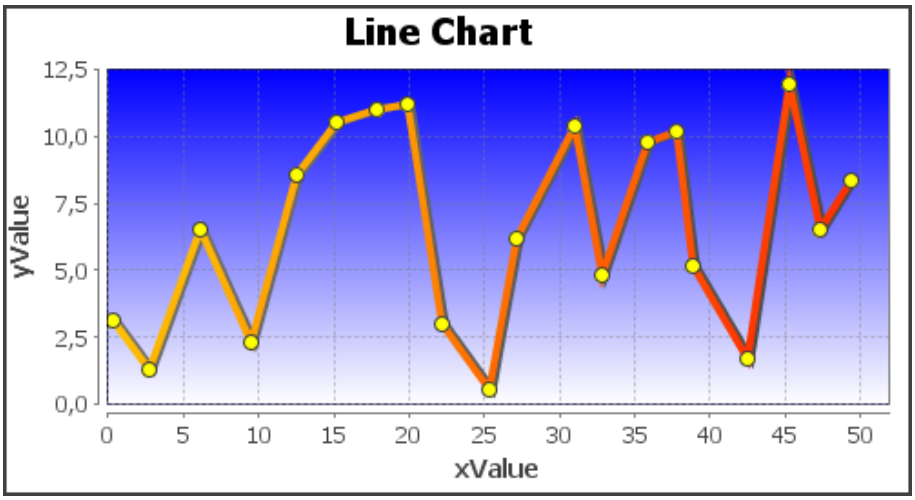


Figura 7.2: esempio di grafico numerico, salvato con il metodo saveAsPNG

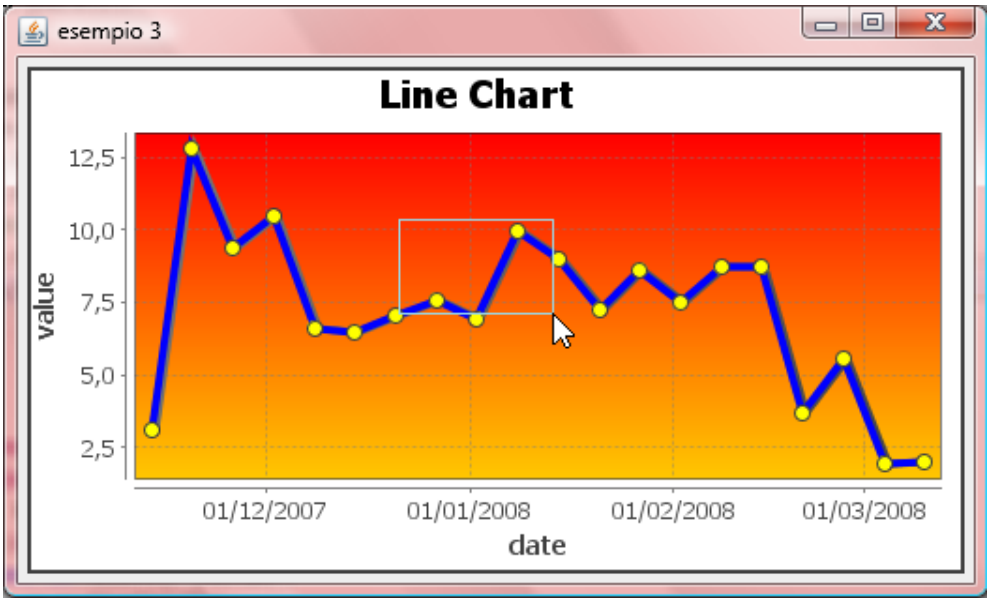


Figura 7.3: esempio di grafico in cui è stata selezionata un'area da ingrandire

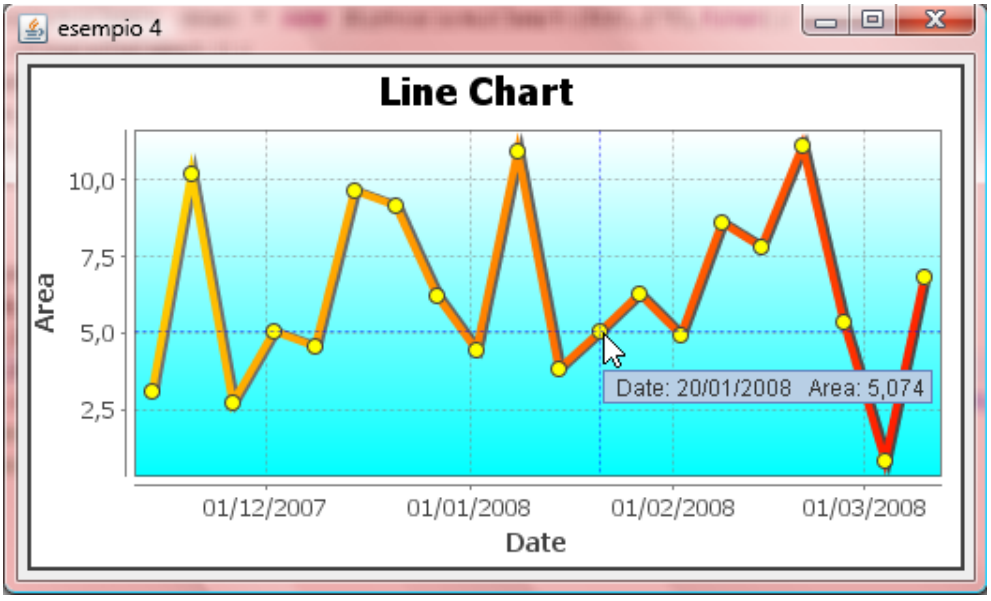


Figura 7.4: esempio di grafico in cui è stato selezionato un dato. Vengono visualizzati il mirino e il tooltip per quel dato

Capitolo 8

JFreeChart

JFreeChart è una libreria grafica open source, rilasciata sotto licenza GNU LGPL(Lesser General Public Licence o licenza di software libero), che permette la generazione di grafici a partire da dati. La versione utilizzata di questa libreria è la 1.0.14, che è anche la più recente, disponibile dal 21 novembre 2011. JFreeChart è una libreria che fa parte del pacchetto software JFree, fondata nel 2000 da David Gilbert e Thomas Morgner [11], del quale fanno parte anche molte altre librerie tra cui JCommon (utilizzata da JFreeChart), JWorkbook o JFreeReport. Oggi JFreeChart è la libreria per grafici maggiormente utilizzata per Java. Come tutte le librerie open source presenti su internet è stata sviluppata da moltissime persone; questo fatto la rende articolata, ma ciò non pone dei limiti alla chiara strutturazione dei suoi contenuti.

Con JFreeChart è possibile creare molte tipologie di grafici di base quali:

- grafici a torta;
- grafici a barre o istogrammi , orientati orizzontalemene o verticalmente;
- grafici lineari;
- scatter plot;
- grafici temporali;
- Gantt plot;
- combinazioni di grafici e molto altro ancora

Molti grafici inoltre possonno essere implementati in versione 2D o 3D, a seconda dell'effetto finale che si vuole ottenere.

JFreeChart rende possibile la creazione di grafici in maniera dinamica, cioè il grafico puó essere aggiornato anche in esecuzione dell'applicazione. Permette inoltre la creazione di grafici interattivi.
In figura 12 è possibile trovare un esempio di chiusura con l'utilizzo di distanza euclidea.

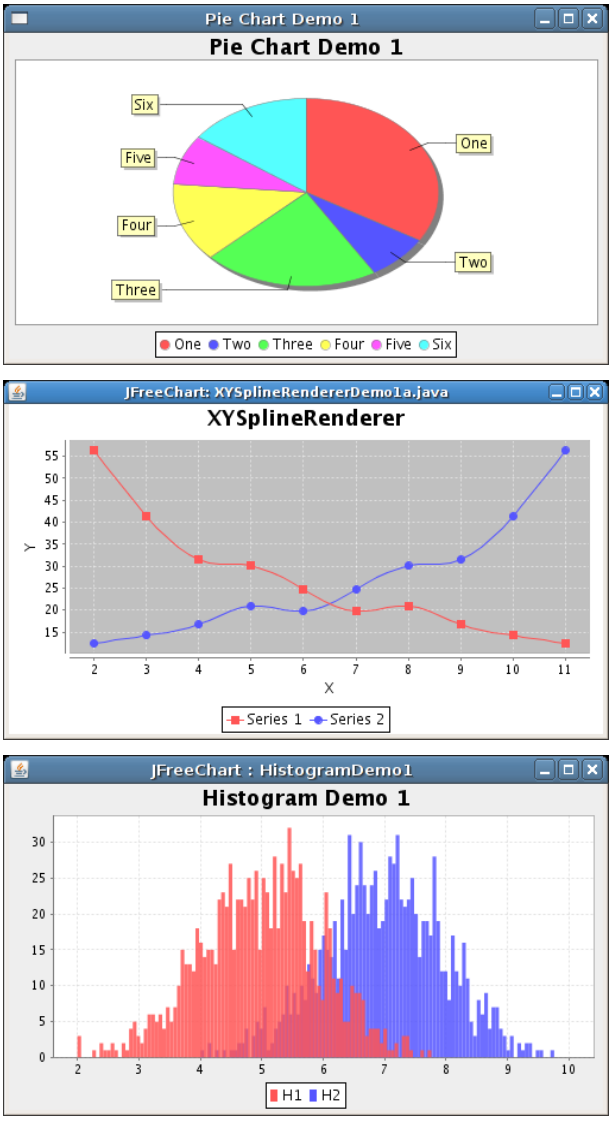


Figura 8.1: Esempi di Grafici ottenibili mediante JFreeChart

Tra le caratteristiche principali di JFreeChart ci sono:

- lo zoom interattivo;
- la possibilità di aggiungere dei tooltip ai grafici;
- la possibilità di gestire gli eventi del mouse sui grafici;
- la presenza di interfacce dataset dedicate all'accesso di dati provenienti da diverse sorgenti, come ad esempio dei database, disponibili nella libreria.
- la possibilità di esportare i risultati in diversi formati, come JPG, PNG, SVG, PDF e ogni altro formato con implementazione Graphics2D.
- la lavorazione anche all'interno di applicazioni, di servlets e applet;
- la disponibilità di tutto il codice sorgente.

L'implementatzine di grafici tramite JFreeChart è consigliabile, rispetto all'utilizzo delle sole API Oracole standard, in quanto rende immediatamente disponibili un'insieme di API, ben documentate, che ne danno diverse tipologie di strutturazione dei dati e di rendering, senza il bisogno della creazione (abbastanza laboriosa) di nuove API ad hoc.

Anche se la qualità dei grafici di base ottenibili da JFreeChart mediante l'uso della classe ChartFactory è molto buona, si è preferito, per l'implementazione proposta, personalizzare ulteriormente i risultati.
JFreeChart mette a disposizione molte interfacce e classi per la creazione personalizzata di grafici, e ciò ha permesso di modificarene ogni aspetto ritenuto opportuno.
Le tre istanze principali modificabili di un grafico sono:

1. gli attributi del diagramma;
2. il rendering del grafico;
3. la definizione dell'asse delle ascisse.

Per modificare gli attributi del diagramma è possibile utilizzare una delle classi che definiscono il plot, che fanno parte del package org.jfree.chart.plot. Cambiare il rendering del grafico permette di agire sul tipo di visualizzazione di dati (dimensioni, colore e texture della linea/barra/fetta ecc che si vogliono utilizzare, sul tooltip, ecc); ciò è possibile tramite una delle classi del package org.jfree.chart.renderer. Infine, per variare l'asse delle ascisse, è possibile utilizzare il package org.jfree.chart.axis.

Ad esempio per l'implementazione del "LineGraph" è stato scelto di utilizzare come renderer un'istanza della clase XYLine3DRenderer. Si sono utilizzate linee di interpolazione 3D, delle quali è stato modificato, attraverso la classe Stroke, lo spessore e il motivo della giunzione tra una linea e l'altra. Inoltre è stato possibile cambiare il colore sia della linea che dell'ombreggiatura.

Attraverso l'utilizzo della classe XYPlot per il plot si è variato il colore, il colore della griglia, il colore e il motivo del mirino e ogni altro particolare. Cambiamenti di analoga natura sono state fatte con l'utilizzo della classe DateAxis per l'asse delle ascisse e delle ordinate, nel caso di grafico temporale.

Un'altra importante modifica apportata alla versione base del grafico è stata quella dei Tooltip, resi più idonei all'obietti prefissato di utilizzo del grafico.

Capitolo 9

Conclusioni e future works

Il principale vantaggio che ha portato la classe implementata è stato quello di proporre dei metodi immediati per la creazione dei grafici. L'utilizzo di JFreeChart in maniera diretta, infatti, è abbastanza laborioso e prevede una conoscenza approfondita della libreria.

La creazione di un grafico mediante LineChart permette di avere subito disponibile un risultato ottimo, anche nel layout. Non permette tuttavia il grado di personalizzazione che offre la libreria JFreeChart. Per gli scopi previsti era però indispensabile mantenere la semplicità nell'utilizzo della classe da parte di terzi. Si è così preferito mantenere bassa la possibilità di modificazione del grafico, preferendo implementare metodi di facile applicazione.

Sviluppi futuri prevedono l'aggiunta di nuovi metodi alla classe implementata, in modo da avere ulteriori funzioni e una maggiore personalizzazione. Inoltre è prevista la creazione di altre classi, che permettano di produrre differenti tipologie di grafici e differenti utilizzi degli stessi, in modo da formare un pacchetto software completo per la gestione dei dati.

Bibliografia

[1] Vladimir Vezhnevets, Vassili Sazonov, Alla Andreeva. *A Survey on Pixel-Based Skin Color Detection Techniques*. Graphics and Media Laboratory, 2003. Faculty of Computational Mathematics and Cybernetics Moscow State University.

[2] Jure Kova, Peter Peer, and Franc Solina. *Human Skin Colour Clustering for Face Detection.*University of Ljubljana, 2003.

[3] Franc Solina, Peter Peer, Borut Batagelj, Samo Juvan. *15 seconds of fame - an interactive, computer-vision based art installation*. University of Ljubljana, Faculty of Computer and Information Science, SI-1000 Ljubljana, Slovenia. 2002.

[4] Son Lam Phung, Douglas Chai, Abdesselam Bouzerdoum. *Adaptive skin segmentation in colour images*. School of Engineering and Mathematics, Edith Cowan University, 2003.Perth, Australia

[5] Hao-kui Tang, Zhi-quan Feng. *Hand's Skin Detection Based on Ellipse Clustering*. International Symposium on Computer Science and Computational Technology, 2008.

[6] Ukil Yang, Bongjoe Kim, Kwanghoon Sohn. *Illumination Invariant Skin Color Segmentation*. Biometric Engineering Research Center. Dept. of Electrical and Electronic Eng. Yonsei University Seoul. South Korea. 2009.

[7] Jasjit S. Suri, David L. Wilson, Swamy Laxminarayan. *Segmentation Models*. Volume I, Part A, Handbook of Biomedical Imagr Analysis. Kluver Academic / Plenum Publisher. pg 371-381. 2005.

[8] Joseph Gil, Ron Kimmel. *Efficient Dilation, Erosion, Opening and Closing Algorithms*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.24, no. 12, december 2002.

[9] Xiaoping Lin, Zhihong Xu. *A Fast Algorithm for Erosion and Dilation in Mathematical Morphology.* WCSE'09. WRI World Congress on Software Engineering. 2009.

[10] Etienne Folio. *Distance Transform.* Laboratoire de Recherche et Dèveloppement de l'Epita. 2008.

[11] David Gilbert. *The JFreeChart Class Library, Reference Documentation.* Simba Management Limited, 2008.