UNIVERSITÀ DEGLI STUDI DI BRESCIA

Facoltà di Ingegneria

Corso di Laurea in Ingegneria dell'Informazione

Dipartimento di Ingegneria dell'Informazione

Studio di tecniche per l'inseguimento della posizione e dei movimenti oculari

Tesi di Laurea Triennale

Relatore

Dott. Nicola Adami

Correlatore

Dott. Sergio Benini

Laureando

Michele Svanera

Matricola

65329

Anno Accademico 2010-2011

Indice

Introduzione

Scenario

I dispositivi di *gaze tracking*, traducibile in italiano come tracciamento dello sguardo, sono definibili come le apparecchiature atte a misurare la direzione dello sguardo di un individuo.

Dispositivi di riconoscimento della posizione dello sguardo sono utilizzati in svariati ambiti, ad esempio all'interno della ricerca sull'apparato visivo, in psicologia, in linguistica cognitiva, nel settore militare, nella progettazione di prodotti, nel marketing e nell'assistenza ai disabili. È tuttavia in quest'ultimo ambito che si concentrano i maggiori sforzi. Le malattie che provocano l'impossibilità motoria sono essenzialmente due. La Sclerosi Laterale Amiotrofica, comunemente detta SLA, e la paralisi cerebrale infantile. Entrambe le malattie provocano una perdita irreversibile delle normali capacità motorie lasciando tuttavia inalterate le capacità visive.

I vantaggi derivanti dall'utilizzo di un sistema di gaze tracking sono notevoli. I più rilevanti sono: una semplicità d'uso e un'alta velocità d'esecuzione. Tali congegni presentano però molti limiti, fra cui quello di essere adoperati soltanto da utenti esperti ed "allenati" ed inoltre l'impossibilità di poter interfacciarsi a più di un individuo per volta. Uno dei difetti più rilevanti è il cosiddetto "tocco di Mida", ovvero non esiste un metodo per confermare l'intenzionalità dello sguardo. Un ulteriore svantaggio è la necessità di una fase iniziale di calibrazione prima dell'utilizzo di quasi tutti i dispositivi di

tracciamento dello sguardo. Un possibile importante svantaggio è dato dal livello d'interazione, o meglio l'invasività, del dispositivo; questo dipende dalla tipologia di dispositivo che si realizza.

Obiettivi e realizzazione

L'obiettivo principale del progetto è la creazione di un sistema capace di riconoscere la zona osservata dall'utente in una griglia di celle. Si è cercato di creare un sistema che fosse compatibile ad una moltitudine di utenti con differenti fisionomie e, ancora più importante, che richiedesse il minor contatto possibile con l'utente.

A seguito degli obiettivi posti, la scelta è ricaduta sull'utilizzo di una tecnica basata sulla luce bianca ovvero che richieda l'utilizzo di una normale fotocamera[1]. In particolare si è cercato di realizzare un algoritmo in grado di rilevare e tenere traccia del limbo[2].

Con lo scopo di ottenere la massima modularità ed espansibilità possibile, si è realizzato il sistema componendolo di una serie di algoritmi a cascata, formato dunque da una serie di livelli. In ingresso ad ogni livello c'è il prodotto di quello inferiore. Ogni livello serve ad elaborare l'immagine proveniente dal livello precedente e ad estrarne l'informazione richiesta.

I passaggi fondamentali sono il rilevamento del volto, il rilevamento degli occhi e la valutazione dello sguardo. Per realizzare quest'ultimo punto vengono sfruttate sei differenti caratteristiche, denominate d'ora in avanti *feature*. Queste features si dividono in due categorie: laterali e verticali. Le features laterali misurano la quantità di sclera ai lati delle iridi, mentre quelle verticali misurano la parte bianca al di sopra di esse. Un'attenta combinazione di questi valori permette la stima della direzione dello sguardo rispetto alle caselle della griglia.

[1] Queste tecniche si contrappongono a quelle basate sulla luce infrarossa che utilizzano fotocamere in grado di rilevare questa lunghezza d'onda.

[2] Il limbo è il confine tra la *sclera* (la parte bianca dell'occhio) e l'*iride* (membrana vascolare dell'occhio di colore variabile).

Il programma realizzato presenta una prima fase di calibrazione, dove l'utente è invitato a guardare ognuna delle caselle per un dato periodo. Questa procedura permette di studiare la fisionomia degli occhi dell'utente e consente di creare i riferimenti necessari per la fase di utilizzo. Nell'ultima fase avviene la stima dello sguardo: avvengono confronti tra i riferimenti presi durante la fase di calibrazione e quelli attuali. Il confronto che restituisce la distanza minore elegge la casella che si ritiene si stia guardando.

Nel programma, in risposta alla volontà di creare di un software che sia di supporto a pazienti affetti da gravi disabilità, è stato implementato un dizionario predittivo. In ognuna delle dieci caselle si possono trovare tre o quattro lettere: viene dunque impressa a video una griglia con illustrate all'interno tali lettere. In base alla selezione delle casella (e di conseguenza delle lettere impresse all'interno) si forma una lista di possibili parole: quella che occupa la prima posizione viene illustrata a video. Oltre alla selezione di lettere sono state implementate altre funzioni sempre inerenti al dizionario, come la possibilità di cancellare una lettera o di selezionare un'altra parola dal dizionario.

Descrizione del periodo di stage

I risultati ottenuti sono stati possibili a seguito di un periodo di stage durato tre mesi. Nella prima fase si è svolto lo studio che ha portato alla realizzazione del capitolo sullo stato dell'arte. Scelto il metodo si è realizzato il sistema vero e proprio.

Non ci sono dubbi nell'affermare che il valore formativo di questo progetto sia notevole, soprattutto per la metodologia con cui è stato approntato. Il periodo di stage ha permesso inoltre di apprendere, a un discreto livello, uno dei linguaggi base della programmazione: il linguaggio C.

Struttura della tesi

Dopo aver definito il termine *gaze tracking*, nel **primo capitolo** vengono presentate le tematiche e i campi d'applicazione dei dispositivi di tracciamento dello sguardo. In poche parole viene definito il contesto nel quale si intende lavorare, con particolare interesse al settore dell'assistenza ai disabili.

Nel **secondo capitolo** viene fatta una panoramica sullo stato dell'arte dei dispositivi di tracciamento dello sguardo. Vengono presentati i metodi e le tecniche realizzative più utilizzate, cercando di fare una panoramica il più possibile completa ed esaustiva. Si presentano inoltre un elenco di alcuni prodotti in commercio e un utile elenco di aziende, consorzi e università che operano nel settore.

Il sistema realizzato ed esposto in questa tesi è composto da una serie di algoritmi a cascata. Nel **terzo capitolo** vengono descritti in dettaglio tutti i moduli di tale programma. Dopo aver stabilito l'elenco dei requisiti richiesti al sistema, si passa alla descrizione dei metodi utilizzati per la stima dello sguardo. L'ultima parte del capitolo descrive brevemente l'implementazione nel programma del dizionario.

La fase di sperimentazione del sistema viene descritto nel **quarto capitolo**. Nella prima fase di sperimentazione si è testato il cascade generato per la fase di riconoscimento del volto. Nella seconda invece si è andato ad analizzare l'efficacia delle features utilizzate per la stima dello sguardo. Nell'ultima parte vengono richiamati i requisiti stabiliti all'inizio del progetto e ne viene analizzata la raggiungibilità e il grado di scostamento.

Nel **quinto** ed ultimo **capitolo** vengono proposti dei possibili sviluppi futuri da apportare al programma. Vengono mostrati per ogni problematica diverse estensioni che potrebbero contribuire ad un aumento delle prestazioni del sistema.

Alla fine viene redatto un capitolo a se stante con le **conlcusioni** del progetto, nel quale viene espressa una sintesi critica ed un giudizio conclusivo.

Capitolo 1

Il contesto

Introduzione In questo primo capitolo viene definito il termine *gaze tracking* e vengono presentate le tematiche e gli ambiti in cui lavorano i dispositivi di tracciamento dello sguardo. Viene definito il contesto nel quale si intende lavorare, con particolare attenzione al settore dell'assistenza ai disabili, importante ed esteso campo di applicazione in cui operano questi dispositivi.

1.1 Definizione del problema

Definizione

Il *gaze tracking*, traducibile in italiano come tracciamento dello sguardo, è definibile come il processo di misura che si propone di stimare la direzione dello sguardo di un individuo. Il termine tracciamento indica l'iterato processo di ricerca.

Dispositivi di riconoscimento della posizione dello sguardo sono utilizzati in svariati ambiti, ad esempio all'interno della ricerca sull'apparato visivo, in psicologia, in linguistica cognitiva, nel settore militare, nell'assistenza ai disabili, nella progettazione di prodotti e nel marketing.

Eye Tracking vs Gaze Tracking

Tra questi due termini, *eye tracking* e *gaze tracking*, intercorre una sottile differenza: con "eye tracking" si intende quel processo di misura utilizzato per la stima della posizione dell'occhio prendendo come riferimento il resto del capo. Il "gaze tracking" è, invece, quel processo di misura utilizzato per stimare la posizione dello sguardo utilizzando come riferimento l'ambiente circostante.

Per la stima dello sguardo si devono necessariamente prendere in considerazione il movimento relativo degli occhi e il movimento del capo; solo una combinazione di questi due porta all'individuazione della posizione dello sguardo. L'unica eccezione in cui i due processi sono identici è quando la testa è immobile; ciò si verifica quando, o per malattie o per dispositivi appositamente costruiti, ne viene limitano il movimento.

Si porta all'attenzione del lettore come spesso questa distinzione venga sfumata sia in letteratura che nelle pubblicazioni, internazionali e non. Nel prosieguo del documento si cercherà il più possibile di distinguere i due termini.

Breve descrizione di vantaggi e svantaggi

I vantaggi derivanti dall'utilizzo di un sistema di gaze tracking sono notevoli.

I più rilevanti sono: una semplicità d'uso e un'alta velocità d'esecuzione. Si presti attenzione a non confondere la velocità di input, alta in questi dispositivi, con la velocità di esecuzione dei programmi annessi a queste apparecchiature, come un programma di scrittura. Per quest'ultimi si raggiunge, solo con l'esperienza, una buona velocità di composizione delle parole. Sfruttando la rapidità dell'occhio si possono quindi scambiare notevoli quantità di informazioni.

Tali congegni hanno però molti limiti, fra cui quello di essere adoperati soltanto da utenti esperti ed "allenati" ed inoltre l'impossibilità di poter interfacciarsi a più di un individuo per volta.

Dispositivi di tracciamento dello sguardo presentano anche altri svantaggi rispetto agli altri dispositivi di input. Il più rilevante è il cosiddetto "tocco di Mida"[1], ovvero non esiste un metodo per confermare l'intenzionalità dello sguardo. Un altro svantaggio da segnalare è la necessità di una fase iniziale di calibrazione prima dell'utilizzo di quasi tutti i dispositivi di tracciamento dello sguardo.

Un possibile importante svantaggio è dato dal livello d'interazione, o meglio l'invasività, del dispositivo; questo dipende dalla tipologia di dispositivo che si realizza, come verrà spiegato in dettaglio in seguito.

1.2 Descrizione del tema

Vengono ora descritti, in breve, i cenni storici e i settori di utilizzo di questi dispositivi.

Cenni storici

Gli studi e le ricerche in materia di tracciamento dello sguardo iniziano già nei primi anni del XIX secolo, anche se si parla principalmente di studi sull'anatomia dell'occhio.

I primi rudimentali dispositivi di tracciamento dello sguardo utilizzavano tecniche di registrazioni di immagini su lastre fotografiche; bisogna tuttavia aspettare l'avvento dell'elettronica di consumo, anni '70-'80 del XX secolo, per trovare le prime realizzazioni cosiddette contemporanee.

[1] Mida, mitico re della Frigia, a cui Sileno diede il potere di trasformare in oro tutto ciò che toccasse. La stessa idea viene trasferita in questo ambito: ovunque lo sguardo si posa viene trasformato in input dal programma. Per approfondimenti si rimanda al capitolo 3.1.

Le prime rare ricerche sulla focalizzazione dello sguardo di un individuo, come ad esempio dove si concentra lo sguardo durante la lettura, si presentano intorno alla metà del XX secolo.

Quando si parla di tracciamento dello sguardo si pensa subito alle tecnologie utilizzate per l'assistenza ai disabili, ma come spesso accade alle nuove tecnologie, le prime realizzazioni in questo settore furono per scopi militari, come la gestione di dispositivi bellici, armi e veicoli, per sistemi di guida che richiedessero un controllo visivo oltre che manuale. I primi dispositivi commerciali nascono solo verso l'inizio degli anni '90.

Settori di utilizzo

I campi di applicazione di questi dispositivi sono, come accennato, molteplici.

Oltre ai già citati settori militari, il tracciamento dello sguardo, viene usato anche nel campo dell'analisi della percezione visiva per studi di psicologia e marketing. Nel campo della psicologia vengono utilizzati per determinare come un individuo reagisca agli stimoli. Ad esempio alcuni eye tracker furono utilizzati per monitorare i movimenti repentini e continui che un individuo fa durante delle fasi del sonno.

Nel campo del marketing viene sfruttato per svolgere studi e ricerche di mercato, per comprendere dove una persona focalizza lo sguardo durante uno spot pubblicitario o durante la navigazione sul web; oppure come posizionare prodotti all'interno di centri commerciali e supermercati, oppure ancora per la forma, i colori e il posizionamento di un marchio.

Un altro esempio d'utilizzo è il miglioramento della segnaletica stradale, dove questi dispositivi sono stati utilizzati per monitorare il livello di attenzione di un automobilista.

In linguistica cognitiva, scienza che analizza le espressioni linguistiche situate nel contesto, vengono utilizzati per determinare dove, nella lettura di

un testo, si concentri l'attenzione del lettore. Un risultato conseguito anche grazie a questi dispositivi, ha portato alla scoperta di come un lettore non concentri lo sguardo su tutte le parole di una frase ma solo su determinate parole chiave. Il campo dove però riscontra più applicazioni è sicuramente il settore medico, che verrà analizzato successivamente.

Settore medico

Oltre ai già citati campi di applicazione, i maggiori sforzi della ricerca si concentrano nel settore medico.

Questi dispositivi vengono utilizzati nell'assistenza alle persone affette da gravi malattie, le quali comportano l'impossibilità di interfacciarsi ad un computer con i normali dispositivi, come mouse o tastiera. Si tratta quindi di una branca della *human-computer interaction* (HCI), interazione uomo-macchina. Un "gaze tracker" permette loro non solo di utilizzare un personal computer, ma molte volte rappresenta l'unico metodo di comunicazione a loro disposizione.

È pensando anche a queste esigenze che ci si è spinti nel progetto di un dispositivo di tracciamento dello sguardo.

Passiamo ora a vedere, con brevi cenni, a quali malattie si rivolgono questi dispositivi.

1.3 Malattie a cui si rivolge

Le malattie che provocano l'impossibilità motoria, senza danneggiare la vista, sono essenzialmente due. La *Sclerosi Laterale Amiotrofica*, comunemente detta SLA, e la *paralisi cerebrale infantile*[2].

Si segnala che molti sono gli studi fatti, nel settore medico, che riguardano l'impatto che questi dispositivi hanno sulla vita dei pazienti. Si cita a titolo

[2]Per affrontare l'argomentazione in questo capitolo farò prevalentemente riferimento ai contenuti dei siti: [1] e l'Associazione Italiana Sclerosi Laterale Amiotrofica [2],[3] e [4].

d'esempio il lavoro svolto dal centro Molinette di Torino in collaborazione con il Politecnico [5], dove si descrive il significativo miglioramento dell'interazione con l'ambiente dei malati affetti da SLA.

Sclerosi Laterale Amiotrofica (SLA)

La *Sclerosi Laterale Amiotrofica* (SLA), conosciuta anche come "Morbo di Lou Gehrig", "malattia di Charcot" o "malattia dei motoneuroni", è una malattia neuro-degenerativa progressiva che colpisce i motoneuroni, cioè le cellule nervose cerebrali e del midollo spinale, che permettono i movimenti della muscolatura volontaria.

Fu descritta per la prima volta nel 1860 dal neurologo francese Jean-Martin Charcot, ed attualmente le sue cause sono ancora ignote.

L'etimologia della definizione "sclerosi laterale amiotrofica" chiarisce le caratteristiche della malattia: la parola "amiotrofico" è composta da tre termini greci, che sono *a* (corrispondente alla negazione), *mio* ("muscolo"), *trofico* ("nutrimento"), quindi significa che i muscoli del malato si atrofizzano per un nutrimento insufficiente; l'aggettivo laterale si riferisce alla zona del midollo spinale che ospita le cellule morenti; lentamente questa zona colpita dal morbo tende a indurirsi ed ecco spiegato l'utilizzo di sclerosi che significa "indurimento".

Il paziente affetto da SLA perde progressivamente i motoneuroni centrali e periferici, con un decorso del tutto imprevedibile e differente da soggetto a soggetto, con esiti disastrosi per la qualità di vita oltre che sulla sua sopravvivenza. Le conseguenze di questa malattia sono la perdita progressiva e irreversibile della normale capacità di deglutizione (disfagia), dell'articolazione della parola (disartria) e del controllo dei muscoli scheletrici, con una paralisi che può avere un'estensione variabile, fino alla compromissione dei muscoli respiratori, alla necessità di ventilazione assistita e quindi alla morte.

La SLA non altera le funzioni cognitive, sensoriali, sessuali e sfinteriali del malato.

Generalmente si ammalano di SLA individui adulti di età superiore ai 20 anni, di entrambi i sessi, con maggiore frequenza dopo i 50 anni. In Italia si manifestano in media tre nuovi casi di SLA al giorno e si contano circa sei ammalati ogni 100.000 abitanti.

Paralisi cerebrale infantile

La *paralisi cerebrale infantile* è un disturbo persistente ma non progressivo della postura e del movimento, dovuto ad alterazioni della funzione cerebrale infantile prima che il sistema nervoso centrale abbia completato il suo sviluppo.

La paralisi cerebrale infantile rappresenta l'esito di una lesione del sistema nervoso centrale che abbia comportato una perdita più o meno estesa di tessuto cerebrale. Le manifestazioni della lesione sono caratterizzate prevalentemente, ma non esclusivamente, da un'alterazione delle funzioni motorie. L'evento lesivo può aver avuto origine in epoca prenatale, perinatale o postnatale, ma in ogni caso entro i primi tre anni di vita del bambino, periodo di tempo in cui vengono completate le principali fasi di crescita e sviluppo della funzione cerebrale nell'essere umano. Il disturbo è definito come persistente, in quanto la lesione a carico del cervello non è suscettibile di "guarigione" in senso stretto, ma la patologia non tende al peggioramento spontaneo perché la lesione stessa, sostituita da tessuto cicatriziale, non va incontro a fenomeni degenerativi. Le manifestazioni della malattia, comunque, non sono fisse, perché i sintomi mutano nel corso del tempo, e possono beneficiare di un trattamento di tipo riabilitativo o, nei casi più gravi, anche chirurgico.

L'incidenza delle paralisi cerebrali infantili, che nei paesi occidentali risulta ormai stabile da alcuni anni, è di 2-3 casi ogni 1.000 nati vivi.

Conclusioni È stato presentato in questo capitolo lo scenario in cui lavorano questi dispositivi; sono stati definiti i settori di utilizzo ed è stato messo in particolare rilievo il settore medico. È proprio in questo ambito che ci si appresta a lavorare vista l'importanza filantropica e considerando come la maggior parte di pubblicazioni e di realizzazioni si concentrino in tale settore.

La descrizione dettagliata dei settori di utilizzo, redatta in questo capitolo, ha permesso di meglio comprendere le esigenze alle quali le realizzazioni devono sottostare. Nel capitolo successivo verrà illustrato in che modo il settore risponde a queste esigenze. Attraverso un minuzioso stato dell'arte verrà verificato il livello raggiunto e verrà fornita un'ampia panoramica riguardante tutti i metodi e tutte le tecniche utilizzate.

Capitolo 2

Stato dell'arte

Introduzione In questo capitolo viene fatta una panoramica sullo stato dell'arte dei dispositivi di tracciamento dello sguardo. Nonostante si presentino i metodi più utilizzati, il lavoro di ricerca svolto è solo uno scalfire la superficie. Uno stato dell'arte in tale ambito richiederebbe un intero lavoro di tesi. Una rapida ricerca sul web potrà dare l'idea dell'enorme numero di pubblicazioni al riguardo.

Si presentano inoltre un elenco di alcuni prodotti in commercio e un utile elenco di aziende, consorzi e università che operano nel settore.

2.1 Categorie

Il tracciamento dello sguardo è un settore dell'interazione uomo-macchina abbastanza recente. Tutte le ricerche e le pubblicazioni in materia si concentrano negli ultimi venti-venticinque anni, con un forte picco di attività nell'ultimo decennio.

Essendo un settore in forte e continuo sviluppo, molte sono le tesi e le pubblicazioni che si sono prefisse, negli anni, di fare ampie panoramiche sull'argomento. Un riferimento storico viene dall'ottimo lavoro di Glenstrup e Engell-Nielsen [6], 1995, uno dei primi grandi lavori di recupero informazioni

del settore. Nel tempo poi si sono seguiti altri lavori, come quello di Morimoto e Mimica [7], oppure la Tesi di Laurea Magistrale "Nuovi metodi di interazione con ambienti virtuali mediante gaze tracking", realizzata da Stefano Rosa [8]. V'è poi, come spiegheremo meglio in seguito, il network *COGAIN* che realizza periodici aggiornamenti sulle nuove tecniche e tecnologie utilizzate.

Verrà ora svolta una panoramica riguardante tutto il settore.

Esistono essenzialmente *tre grandi categorie* di tracciamento dei movimenti dell'occhio umano, queste categorie di differenziano per la metodologia utilizzata:

1. utilizzo di elettrodi per la misurazione del potenziale elettrico della pelle intorno all'occhio;
2. utilizzo di speciali lenti a contatto che misurano la rotazione dell'occhio;
3. rilevamento e misurazione della luce che incide sull'occhio tramite elaborazione grafica.

Vengono di seguito descritte in ordine decrescente di invasività.

Elettrodi

La prima categoria utilizza degli elettrodi posizionati intorno e sopra gli occhi per misurarne il potenziale elettrico.

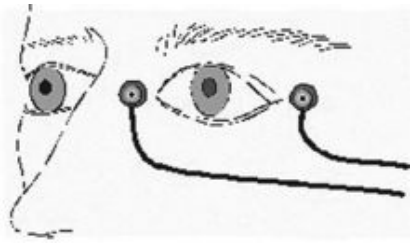


Figura 2.1: Elettrooculogramma (EOG) [9].

Gli elettrodi vengono posti a contatto sulla pelle intorno agli occhi, generalmente o sopra e sotto l'occhio o a sinistra e a destra di esso, e

misurano il potenziale che si viene ad instaurare tra la cornea e la retina misurando di fatto il campo elettrico generato. Un esempio è mostrato in figura 2.1.

Se l'occhio è spostato dalla posizione di centro verso un elettrodo, questo elettrodo "vede" il lato positivo della retina e l'elettrodo opposto "vede" il lato negativo. Di conseguenza, si verifica una differenza di potenziale misurabile tra gli elettrodi. Supponendo che il potenziale di riposo è costante e facilmente ricavabile, il potenziale registrato è una stima della posizione degli occhi.

Questo speciale esame è chiamato *Elettrooculogramma* (EOG) [10] e le sue principali applicazioni si riscontrano in diagnostica oftalmologica, branca della medicina che si occupa della fisiologia anatomia e le malattie degli occhi e, appunto, nella registrazione dei movimenti oculari.

È facile intuire quanto questo metodo sia estremamente invasivo e richieda un'elevata conoscenza dell'anatomia dell'occhio. Questo metodo permette però la registrazione dei movimenti oculari anche ad occhi chiusi, ed per questo motivo che è utilizzato nella ricerca durante le fasi del sonno. Opera anche in condizioni di assenza di luce.

Da segnalare come punto di partenza per lo studio di questa branca della medicina il lavoro svolto dal Royal Liverpool University Hospital [11]. Si segnala inoltre uno, dei molti, studi fatti in materia [12], dove viene fatta un'analisi del movimento dell'occhio tramite EOG, per il riconoscimento di movimenti dell'occhio come saccadi e fissazioni[1].

Lo studio appena illustrato rappresenta un esempio di dispositivi di tracciamento dello sguardo. Tuttavia in questa categoria di dispositivi si predilige il rilevamento della posizione dell'occhio piuttosto che la stima dello sguardo.

[1] Per approfondimenti si rimanda all'appendice (D.1).

Lenti a contatto

La seconda categoria utilizza dispositivi fisici da applicare all'occhio come speciali lenti a contatto. Rientrano in questa categoria anche quei dispositivi come speciali occhiali o sensori che rilevino il campo magnetico, con i quali è possibile misurare la rotazione dell'occhio durante i suoi normali movimenti. Un esempio commerciale è illustrato in figura 2.2.

Figura 2.2: Mobile eye tracking - Tobii Glasses [13].

Sono apparecchi di misura estremamente sensibili e sono quelli che offrono i maggiori risultati. Sono tuttavia abbastanza invasivi, anche se non al livello della categoria precedente. Sono utilizzati per brevi periodi e per specifici scopi come ad esempio dai ricercatori che studiano le dinamiche e la fisiologia alla base del movimento dell'occhio.

L'obbligo da parte dell'utente di indossare lenti a contatto speciali, eventualmente collegate a dei fili, rende la tecnica troppo ingombrante per le attività non strettamente di laboratorio. È tuttavia opinione comune che un giorno lenti a contatto per il tracciamento dello sguardo saranno comunemente utilizzate ed accettate nella vita di tutti i giorni.

Elaborazione grafica

La terza categoria è quella che richiede meno contatto fisico con l'occhio ed è per questo il metodo più utilizzato. Si basa essenzialmente sull'elaborazione

grafica e ed è la tecnica alla base di tutti i dispositivi, commerciali e non, adibiti all'assistenza ai disabili.

Le varie tecniche di questa categoria si suddividono in base alla componente hardware richiesta. È possibile distinguere due sottocategorie in base al hardware utilizzato: la prima utilizza fotocamere tradizionali, la seconda fotocamere in grado di rilevare luce a lunghezza d'onda infrarossa.

Queste tecniche prevedono come componente hardware solo una fotocamera, spesso con una buona definizione, o poco più, con lo scopo di catturare le immagini. Tale immagini vengono poi modificate ed elaborate via software per estrapolare la posizione e il movimento dell'occhio. È senza dubbio la tecnica meno invasiva e verrà qui di seguito trattata nei suoi molteplici aspetti, sia vantaggi che svantaggi. È la categoria scelta per la realizzazione del dispositivo proposto in questa tesi, in quanto si avvicina di più ai requisiti stabili nella sezione 3.1.

Nella trattazione, per completezza d'informazione e per dovere storico, sono state incluse tutte quelle tecniche che prevedono fotocamere montate su speciali supporti, come occhiali o apposite fasce montate intorno o in prossimità della testa. Storicamente, come già accennato nel capitolo precedente, le prime realizzazioni vennero fatte in ambito militare utilizzando proprio queste tecniche. Molte sono le ricerche e i risultati ottenuti in questa direzione.

2.2 Tecniche di elaborazione grafica

Come accennato in precedenza esistono due grandi sotto categorie dell'elaborazione grafica: quella che utilizza luce a lunghezza d'onda infrarossa o quella che utilizza luce bianca.

Le principali realizzazioni che riguardano l'elaborazione grafica con luce a lunghezza d'onda infrarossa prevedono lo sfruttamento del fenomeno fisico della luce riflessa. La riflessione della luce è quel fenomeno fisico per cui

quando la luce, che ricordiamo essere un'onda elettromagnetica, incontra una discontinuità nel mezzo viene generalmente divisa in due fasci, uno trasmesso e uno riflesso. La luce riflessa, catturata dalla fotocamera, contiene tutte le informazioni utili alla stima di distanza e posizione. L'utilizzo di quella particolare lunghezza d'onda è dovuta a particolari fenomeni che avvengono quando un luce ad infrarosso indice sull'occhio. In particolare si fa notare come il fenomeno della riflessione cambi caratteristiche al variare della lunghezza d'onda. Tali fenomeni verranno poi spiegati in dettaglio in seguito.

L'utilizzo della luce bianca prevede invece l'acquisizione di un'immagine da una normale fotocamera. L'immagine acquisita viene poi elaborata per estrapolarne tutte le informazioni volute, come ad esempio la posizione e il movimento dell'occhio. La maggior parte di queste tecniche si basano sulla geometria dell'occhio.

La quasi totalità dei dispositivi che utilizzano queste tecniche non può essere utilizzata con gli occhiali da vista, anche se, come vedremo in seguito, esistono alcune soluzioni a questo problema.

Vengono qui di seguito descritte tutte queste tecniche, suddivise per tipologia di luce utilizzata [6].

2.2.1 Tecniche basate sulla luce infrarossa

Tre sono le principali tecniche di rilevamento e tracciamento dello sguardo che utilizzano luce infrarossa riflessa dall'occhio:

- il monitoraggio del rapporto tra la cornea e la riflessione della pupilla;
- il monitoraggio delle immagini di *Purkinje-Sanson*;
- il monitoraggio del rapporto tra il riflesso corneale e l'immagine dell'occhio utilizzando una rete neurale artificiale.

L'utilizzo degli infrarossi, con lunghezza d'onda di circa 880 nm, ha essenzialmente due scopi: rilevare dei particolari fenomeni di riflessione chiamati immagini di *Purkinje*[2], oppure migliorare il contrasto tra la pupilla e l'iride.

Oltre ai già citati vantaggi l'elaborazione di immagini a lunghezza d'onda infrarossa è utilizzata per la possibilità di controllare i livelli d'illuminazione ambientale attraverso illuminazione attiva, ottenibile tramite appositi diodi LED. Gli altri vantaggi dell'infrarosso sono che non è visibile ad occhio nudo e quindi tale luce non distrae l'utente, ma è rilevabile da comuni fotocamere commerciali.

Tuttavia ci sono una serie di limitazioni agli approcci con luce infrarossa. Ad esempio, la presenza di luce ambientale ad infrarossi, o altri dispositivi che ne utilizzano la stessa lunghezza d'onda, possono causare disturbi nell'acquisizione delle immagini. Inoltre, le prestazioni dei sistemi a spettro infrarosso possono variare in modo significativo a causa di differenze individuali nella proprietà fisiologiche ed anatomiche dell'occhio.

Vengono di seguito descritti in dettaglio tutti i metodi che utilizzano luce infrarossa.

2.2.1.1 Tracciamento del rapporto tra la cornea e la riflessione della pupilla

Questa tecnica si basa sul metodo denominato PCCR (*Pupil Center - Corneal Reflection*). Un diodo (LED) emette raggi infrarossi a bassa potenza ed illumina l'occhio. Il fascio di luce produce due effetti sugli occhi. Il primo è una riflessione luminosa sulla superficie della cornea dell'occhio, detta prima immagine di *Purkinje* o anche "riflesso corneale" (figura D.1) e provoca il cosiddetto *glint* (luccichio).

Approssimando l'occhio con una sfera che ruota attorno al suo centro, il luccichio non si muove con la rotazione degli occhi. Per questo motivo il

[2]Per approfondimenti si rimanda all'appendice (D.3).

riflesso può essere preso come punto di riferimento.

Il secondo riflesso è chiamato effetto *bright-eye*, o *Bright Pupil* (occhio luminoso), che è un fenomeno molto simile all'effetto occhi rossi nelle fotografie. Si verifica infatti che la pupilla, fatta incidere da una luce, si illumina e diventa molto visibile poiché assume un forte contrasto con l'iride. Ciò è dovuto al fatto che la luce che passa attraverso la pupilla incide sulla retina e viene riflessa indietro provocando un'illuminazione della pupilla, come illustrato in figura 2.3.

Da notare come queste due riflessioni avvengano anche con luce bianca; tuttavia, solo con luce infrarossa si verificano con un'intensità tale da essere facilmente individuabili.

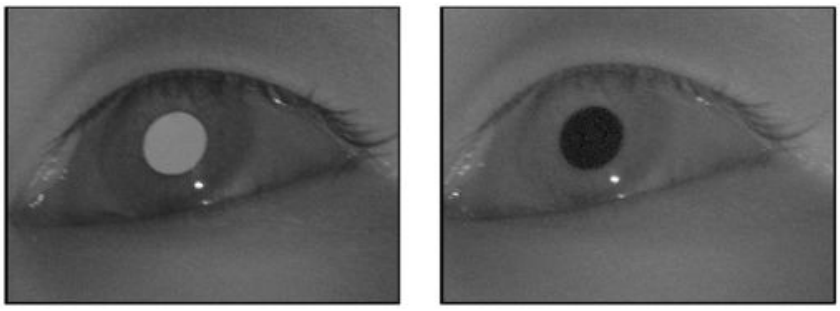


Figura 2.3: Bright Pupil e Dark Pupil. I luccichii, o riflessi corneali, sono facilmente individuabili come i puntini bianchi subito al di sotto della pupilla [14].

Il fenomeno Bright Pupil avviene solo quando la luce è coassiale con l'asse visivo dell'occhio, l'occhio agisce quindi come un retro-riflettore[3]. La pupilla si presenta quindi come un cerchio bianco acceso, facilmente individuabile tramite elaborazione. Una volta rilevato il cerchio ne viene calcolato il centro. Si misura poi la distanza tra il centro della pupilla (Bright Pupil) e il luccichio e ne viene estratto un vettore. Questo vettore rappresenta una stima della direzione dello sguardo.

[3]Per approfondimenti si rimanda all'appendice (D.2).

Ci sono diverse implementazioni di questi stessi principi, come tenere il Bright Pupil fisso e il luccichio variabile, al contrario di come precedentemente esposto. Tutte però si basano sugli stessi principi fisici.

Uno dei primi esempi completi apparsi su questa tecnica viene dal lavoro di Yoshinobu Ebisawa [15]. In questo lavoro viene proposta, in primo luogo, una tecnica di rilevazione delle pupille utilizzando due differenti sorgenti luminose. In secondo luogo, per gli utenti che portano gli occhiali, viene proposto un metodo per eliminare le riflessioni che si creano sul vetro della lente.

Una particolare variante viene proposta da Zhiwei Zhu e Qiang Ji [16]. Nel metodo proposto sono previsti due anelli concentrici di LED ad infrarossi posti intorno alla telecamera per produrre l'effetto, precedentemente discusso, sulla pupilla. I due anelli sono accesi e spenti alternativamente tramite un decoder, per "accendere" e "spegnere" la pupilla. L'alternanza di colore viene sfruttata per rilevare e tracciare la pupilla. Siccome il movimento della testa causa il mal rilevamento dello sguardo viene integrato con un tracciamento del volto. Vengono usati algoritmi di Reti Neurali (GRNNs) in fase di calibrazione.

L'utilizzo delle librerie OpenCV, come vedremo in seguito, facilità le fasi di elaborazione delle immagini[4]. Le stesse librerie vengono utilizzate nella realizzazione del software presentato nel lavoro di Manu Kumar [17]. Vengono, nella pubblicazione, esaminati i fattori che contribuiscono agli elevati costi dei sistemi di tracciamento dello sguardo e vengono proposte diverse tecniche e strategie atte a ridurne i costi. Il metodo utilizzato è sempre lo stesso degli infrarossi. Il tutto è realizzato tramite due webcam commerciali poco costose.

Una dimostrazione della ricerca nel vecchio continente viene dal lavoro presentato da A.Pérez e dai suoi collaboratori [18]. Quattro fonti di luce

[4]Per un approfondimento sulle librerie OpenCV si rimanda all'appendice (B).

infrarossa, sincronizzate con l'otturatore della fotocamera, vengono utilizzati per produrre riflessi corneali. La linea dello sguardo è determinato sempre nello stesso modo, utilizzando il vettore pupilla-riflesso. Viene implementato, tramite un'altra fotocamera il tracking della testa.

Variante: *Bright Pupil* vs *Dark Pupil*

In aggiunta alla tecnica Bright Pupil si aggiunge una seconda tecnica detta *Dark Pupil*, illustrata nella figura 2.3. Il metodo del Bright Pupil si ha, come già discusso, quando l'illuminazione è coassiale con il percorso ottico; in questo caso l'occhio agisce come un retro-riflettore. Se invece la fonte di illuminazione è fuori dal percorso ottico, la pupilla appare scura perché la retro-riflessione dalla retina è diretta lontano dalla fotocamera. Questo fenomeno è noto come Dark Pupil, pupilla scura.

Questa variante è utilizzata in molte soluzioni proposte negli anni passati, come integrazione al metodo bright pupil. Il vantaggio di ottenere il cosiddetto Dark Pupil è quello di rendere la pupilla ancora più scura, rispetto alle normali condizioni di luce, in modo che sia maggiormente rintracciabile ed evidenziabile tramite elaborazione grafica.

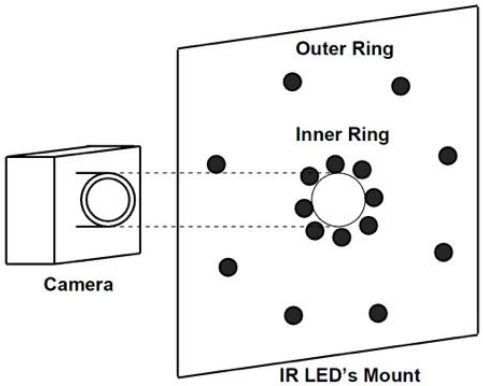


Figura 2.4: Hardware utilizzato: fotocamera con luci ad infrarossi.

L'esemplare realizzato e proposto da C.H. Morimoto [14], utilizza proprio questa variante. Viene presentato un veloce, robusto e a basso costo sistema

di rilevamento della pupilla che utilizza due sorgenti di luci infrarossi multiplexati sincronizzati con la frequenza di aggiornamento dei fotogrammi della telecamera. Anche in questo caso è previsto l'utilizzo di due anelli concentrici di LED ad emissione infrarossa; al centro è stata posta la fotocamera, come illustrato in figura 2.4.

Le due fonti di luce generano immagini chiare e scure della pupilla, che vengono utilizzate per la segmentazione di quest'ultima. Viene inoltre proposta una soluzione per le persone che utilizzano gli occhiali da vista.

La University of Copenhagen, Denmark, uno dei migliori centri di ricerca del settore, propone un'apprezzabile dispositivo, attraverso i ricercatori Hansen e Hammoud [19]. In questo lavoro vengono applicati sempre gli stessi principi dell'esempio precedentemente illustrato, con qualche piccola variante. Da notare come il rilevamento dell'occhio avvenga tramite *AdaBoost* e *Haar-like Features*[5].

Un altro lavoro svolto in questa direzione è quello presentato da Dongheng Li e David Winfield [20]. Questo lavoro si propone di rilevare i movimenti oculari tramite il monitoraggio della pupilla utilizzando l'algoritmo *Starbust*. L'obiettivo dell'algoritmo è quello di estrarre la posizione del centro pupilla e il riflesso corneale in modo da mettere in relazione la differenza vettoriale tra tali misure. L'algoritmo inizia con l'individuazione e la rimozione del riflesso corneale dall'immagine. Poi i punti sul bordo della pupilla vengono trovati usando un'apposita feature.

Figura 2.5: Dispositivo montato sul capo.

[5]Per approfondimenti si rimanda all'appendice (A.1).

In tale senso, nonostante si utilizzi luce infrarossa e si applichi il solito metodo del riflesso corneale, questo metodo rientra di diritto anche nella categoria di rilevamento delle pupille, trattata in seguito. Questo algoritmo promette di avere risultati migliori di qualsiasi feature estraibile dall'occhio. Si utilizza una webcam montata su speciali occhiali come si può vedere in figura 2.5. Questo metodo garantisce livelli di precisione di circa un grado di angolo visivo.

Considerazioni

Come per le altre tecniche, vasti movimenti del capo possono fare decadere le prestazioni in quanto il "luccichio" esce dalla zona di visibilità della telecamera. Per ovviare al problema molti lavori implementano un tracciamento del volto.

Questo metodo ha il vantaggio di essere molto più accurato rispetto ad un'elaborazione prettamente geometrica. Può inoltre lavorare su differenti condizioni di luminosità, che è il grosso svantaggio di tutte le tecniche che utilizzano luce bianca. In aggiunta al metodo Bright Pupil viene spesso utilizzato il metodo Dark Pupil.

Da segnalare come la richiesta di un intervento hardware aggiuntivo, una luce ad infrarossi, oltre ad una fotocamera in grado di rilevarli, presenti un grosso svantaggio in termini di tempo, di lavoro e di costi. Presenta anche lo svantaggio di stancare l'occhio dell'utilizzatore. Questo è causato dalla luce, che incidendo all'interno dell'occhio, ne disidrata la normale umidificazione, nonostante ovviamente la luce non sia nociva. L'utilizzo è limitato a qualche ora.

Sembra tuttavia il metodo migliore e che fornisce la miglior precisione possibile. Questa considerazione deriva anche dal fatto che praticamente tutti i dispositivi in commercio utilizzano questo metodo. La calibrazione è necessaria in quanto ogni utenza presenta occhi di diverse forme e dimensioni.

2.2.1.2 Il monitoraggio delle immagini di *Purkinje*

La prima e la quarta immagine di *Purkinje*, illustrate in figura D.1, possono essere rilevate ed utilizzate per il tracciamento della direzione dello sguardo con la tecnica *Dual-Purkinje Image*. Questa tecnica utilizza le posizioni relative di queste due riflessioni per calcolare la direzione dello sguardo.

Il lavoro di Micheal R. Clark [21], è uno delle poche realizzazioni esistenti che utilizzino questa tecnica. Questo articolo descrive un dispositivo di tracciamento dello sguardo bidimensionale che utilizza informazioni sulla posizione spaziale delle due riflessioni citate per determinare la posizione degli occhi.

Considerazioni

La tecnica Dual-Purkinje Image è in generale più accurata rispetto alle altre tecniche, e la frequenza di campionamento è molto alta, fino a 4000Hz. Lo svantaggio di questa tecnica è che la quarta immagine di Purkinje è piuttosto debole, quindi l'illuminazione circostante deve essere molto controllata.

2.2.1.3 Il rapporto tra il riflesso corneale e l'immagine dell'occhio utilizzando una rete neurale artificiale

Una delle tecniche di più recente sviluppo è quella che prevede l'utilizzo di una Rete Neurale Artificiale (*Artificial Neural Network*, ANN) che rappresenta un metodo per approssimare funzioni usando reti di calcolo composte da elementi elementari interconnessi.

L'ingresso a questo algoritmo è un'immagine video digitale dell'utente presa dalla fotocamera, che catturi l'intera testa dell'utente. Viene individuato l'occhio destro dell'utente tramite una luce ferma posta di fronte che causa un riflesso sull'occhio. Successivamente viene estratto un piccolo rettangolo dell'immagine video, in genere 40 $\times$ 15 pixel (dipende dalla risoluzione della

fotocamera), con centro nel riflesso, e la si immette in una ANN, come illustrato in figura 2.6. L'uscita del ANN sono le coordinate dello sguardo.

Questa tecnica non richiede, per fare il riflesso, l'utilizzo di luce ad infrarosso ma viene spesso accompagnata dall'utilizzo di questo tipo di luce che, come già accennato in precedenza, permette una visione migliore dell'occhio e diminuisce la dipendenza dalle condizioni d'illuminazione.

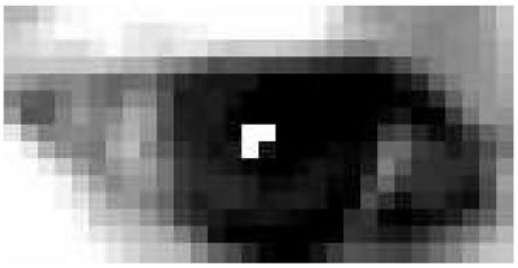


Figura 2.6: Immagine dell'occhio di 30 × 15 pixel, in bassa risoluzione; fa da ingresso al sistema ANN. Si può facilmente notare la riflessione al centro dell'immagine [6].

L'utilizzo di reti neurali è alla base del lavoro svolto da Li-Qun Xu, Dave Machin e Phil Sheppard [22]. Viene proposto in questo lavoro un tracciamento dello sguardo in tempo reale attraverso l'utilizzo di una rete neurale combinata con algoritmi robusti di elaborazione delle immagini. Il sistema apprende in modo efficace la direzione dello sguardo dell'utente attraverso la modellizzazione dell'anatomia dell'occhio umano, attraverso le posizioni relative della pupilla, e la riflessione della luce all'interno della cavità oculare. Questo lavoro non prevede l'utilizzo di hardware specifico.

Un italico esempio viene dal lavoro presentato da Diego Torricelli e colleghi [23]. Il metodo proposto non rientra solo nelle tecniche che sfruttano le Reti Neurali ma piuttosto combina tecniche diverse per far fronte a problemi di movimento della testa, d'illuminazione e di fruibilità nel quadro delle applicazioni a basso costo. Feature detection, come il rilevamento dei contorni, e algoritmi di tracciamento sono stati utilizzati per ottenere una configurazione automatica e per rafforzare la robustezza dell'algoritmo in

condizioni differenti di luce. Un'ampia analisi delle soluzioni che utilizzano reti neurali è stata effettuata per far fronte alla non-linearità associate allo sguardo in condizioni di ampi movimenti della testa. Nessun hardware specifico, come l'illuminazione a infrarossi o macchine fotografiche ad alta risoluzione, è necessario; solo una semplice webcam commerciale.

Considerazioni

Si nota subito come la precisione di questo sistema non è buona come per le altre tecniche utilizzanti luce infrarossa. Il grande vantaggio di questa tecnica è che prendendo solo un determinato rettangolo dell'immagine di base, è aumentata la mobilità della testa (fino a 30 cm). Ciò potrebbe renderla una soluzione implementabile in progetti in cui un alto grado di precisione non è essenziale, ma che richiedano un'elevata mobilità del capo.

Uno svantaggio è che il metodo ANN richiede più di una semplice calibrazione rispetto alla altre tecniche; esso deve essere addestrato a raccogliere immagini della testa e dell'occhio dell'utente per alcuni minuti mentre l'utente segue visivamente un cursore in movimento sul display. Questo è seguito da una sessione di allenamento automatico, che utilizza le immagini memorizzate precedentemente, della durata di parecchi minuti. Dopo di che il sistema non necessita di una nuova calibrazione.

Si può facilmente intuire come l'addestramento della rete neurale sia l'elemento cruciale del metodo e anche quello che ne definisce il grado di accuratezza.

2.2.2 Tecniche basate sulla luce bianca

Passiamo ora a vedere le tecniche di elaborazione grafica che utilizzano la luce bianca. Come già si può intuire dalle tecniche illustrate, i confini tra un metodo e l'altro sono piuttosto labili, e spesso le tecniche si fondono e si complementano.

Le tecniche che utilizzano luce bianca sono essenzialmente divisibili in due categorie:

- il rilevamento del limbo;
- il rilevamento delle pupille.

2.2.2.1 Il rilevamento del limbo

Il limbo è il confine tra la sclera (la parte bianca dell'occhio) e l'iride (membrana vascolare dell'occhio di colore variabile). Il colore bianco della sclera, tolti alcuni vasi sanguigni, e quello scuro dell'iride formano un contrasto facilmente individuabile e monitorabile otticamente. Spesso questa tecnica si basa sulla posizione del limbo in relazione alla posizione della testa; quest'ultima deve essere tenuta quindi abbastanza ferma o si deve predisporre l'apparecchio per essere fissato alla testa dell'utente risultano però abbastanza invasivo. In soluzione al problema si potrebbe implementare un tracciamento tridimensionale della testa, o almeno bidimensionale del volto, in modo da permettere maggior movimento all'utente.

Si includono in questa categoria anche tutti quei metodi di rilevamento della sola sclera e del solo iride.

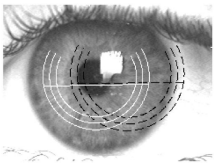


Figura 2.7: Immagine 240 × 180 pixel dell'iride scattata con illuminazione ambientale.

Il lavoro svolto da Ivins e Porrill [24], può essere incluso in questa tipologia di tecnica. Nell'elaborato si implementa una modellizzazione dell'iride umano per la misura accurata dello sguardo con movimenti oculari a tre dimensioni, partendo da sequenze di immagini video. Siccome l'espansione e la contrazione della pupilla riduce la precisione di questo metodo, è stato

sviluppata una modellizzazione dell'iride. Questo modello, illustrato in figura 2.7, può rilevare movimenti orizzontali, verticali, e di rotazione.

Un'altra implementazione di questa tecnica viene dalla ricerca di Betke, Mullally e Magee [25]. In questo lavoro viene proposto un metodo in real-time per il rilevamento della sclera, la parte bianca dell'occhio. Questo sistema si divide in due operazioni: il rilevamento e il tracciamento del volto, e il rilevamento della sclera, dove, se possibile, facendo uno zoom sull'occhio. Per la rilevazione della sclera viene applicata una teoria statistica decisionale, in particolare il *Bayes Decision Thresholds*, che determina, dopo una prima fase di training, se un determinato pixel del volto è della pelle o della sclera dell'occhio.

L'elaborato di Dongheng Li e Derrick Parkhurst [26], riprende una realizzazione ideata per la luce ad infrarossi e la converte all'utilizzo di luce bianca. Questo lavoro ha il preciso scopo di realizzare un eye tracking utilizzando esclusivamente luce bianca, soppiantando la luce ad infrarossi e i suoi limiti, come ad esempio lo scarso rendimento in ambienti esterni dovuto alla presenza di luce infrarossa naturale, oppure le scarse prestazioni dovute alle differenze individuali nelle proprietà fisiologiche dell'occhio. Questo lavoro è un upgrade del lavoro svolto l'anno prima sempre da Li, Winfield e Parkhurst [20]. A differenza dal precedente lavoro che, come spiegheremo in dettaglio di seguito, venne originariamente progettato per monitorare la pupilla in immagini a infrarossi, questo si occupa di rilevare e monitorare il limbo in condizioni di luce bianca. Anche questo elaborato, come il precedente, utilizza l'algoritmo *Starburst*. Viene anche previsto un algoritmo che permetta di eliminare i riflessi nell'occhio. La fotocamera è montata su un braccio applicato ad una mentoniera.

La ricerca redatta da Kourkoutis, Panoulas e Hadjileontiadis [27], è un importante esempio di utilizzo di questa tecnica. Essendo però la teoria di base per il progetto realizzato, verrà spiegato in dettaglio nella sezione 3.2.

Considerazioni

I movimenti della testa e il movimento delle palpebre svantaggiano questa tecnica o addirittura la rendono inutilizzabile, senza un algoritmo di tracciamento del volto. Richiede un software non troppo preciso, ma i risultati non sono eccellenti, in quanto spesso non si tiene l'occhio completamente aperto con parte bianca sopra e sotto (oltre che ai lati) ma semi aperto. Le parti bianche quindi sono solo ai lati, il che lo rende un ottimo metodo per rilevare i movimenti laterali (dx e sx) ma pessimo per rilevare i movimenti verticali (sopra e sotto). In tal senso verranno previste e adottate soluzioni.

2.2.2.2 Il rilevamento delle pupille

Il rilevamento delle pupille è simile al metodo precedentemente visto del tracciamento del limbo. In questo caso il confine monitorato è quello tra la pupilla (cerchio nero più interno dell'occhio) e l'iride (membrana vascolare dell'occhio di colore variabile). Anche in questo caso l'apparecchio deve essere posizionato in modo che si muova seguendo la testa.

Uno dei pochi esempi esistenti è la Tesi di Laurea di Isabella Signorile [28]. Sistemi ottici che sfruttando algoritmi di elaborazione dell'immagine, riescono a determinare le coordinate del centro pupilla, e in seguito a opportune considerazioni geometriche, a determinare il punto fissato sullo schermo dall'utente. Calcola la distanza tra il centro dell'occhio ed un riferimento fisso. Questo richiede l'applicazione di un pezzo di carta nero sulla fronte, il che, anche se lievemente, incide sulla invasività del metodo. In questo lavoro sono state usate le *Victor image processing Library* ed è stato scritto in codice C.

Considerazioni

Il vantaggio di questa tecnica rispetto al tracciamento del limbo consiste nel fatto che la pupilla è poco coperta da palpebre e permette di tenere

traccia dei movimenti anche in verticale. Altro vantaggio è che il bordo della pupilla è più spesso rispetto al limbo. Lo svantaggio principale è che la differenza di contrasto tra pupilla e iride è più basso che tra l'iride e sclera, rendendo così l'individuazione del confine più difficile.

Richiede un software (e un hardware) più preciso e complesso del precedente, ma offre senza dubbio prestazioni migliori. I problemi di invasività possono essere superati attraverso l'utilizzo di un tracciamento del capo.

2.2.2.3 Altre metodologie

Vengono ora presentate altre due soluzioni proposte negli anni che utilizzano luce bianca.

Un esempio sicuramente fuori dall'ordinario è presentato da Kruger, Happe e Sommerl [29]. In questo articolo viene presentato un metodo per il tracciamento dello sguardo che si basa su una rappresentazione *wavelet* di un modello di volto. La trasformata wavelet si riferisce alla rappresentazione di un segnale mediante l'uso di una forma d'onda oscillante di lunghezza finita (nota come wavelet madre). Questa forma d'onda è scalata e traslata per adattarsi al segnale in ingresso. La trasformata wavelet è spesso paragonata alla trasformata di Fourier, dove i segnali sono rappresentati come somma di sinusoidi. La differenza principale è che le wavelet sono localizzate sia nel tempo che nella frequenza mentre la trasformata di Fourier standard è localizzata solo in frequenza [30]. La rappresentazione wavelet permette deformazioni arbitrarie delle immagini della fotocamera, permette di generalizzare da un modello individuale di volto a un modello di volto generale e permette di adattare le esigenze di calcolo dell'algoritmo di tracciamento per le risorse di calcolo disponibili.

Un esempio non trascurabile è dato dal software open-source "TrackEye : Real-Time Tracking Of Human Eyes Using a Webcam" [31]. Questo lavoro, scritto in C++, fa largo uso delle librerie OpenCV. In questo la-

voro ci sono due implementazioni diverse: *Adaptive EigenEye Method* e *Template-Matching*. Quest'ultima implementazione si basa essenzialmente sul confronto tra situazioni prestabilite. In memoria ci sono una serie di immagini contenenti solo un occhio, il quale assume nelle diverse immagini diverse posizioni. Tramite la libreria OpenCV si effettua un matching, un confronto, tra l'immagine in tempo reale presa da una webcam e le varie immagini campione. Il confronto che offre più probabilità è scelto per determinare la posizione attuale dell'occhio. Questo metodo è di facile implementazione ma ha il grande svantaggio di essere poco lungimirante e inoltre richiede tanti confronti, quindi tanti calcoli. L'implementazione Adaptive EigenEye Method, invece, si basa sul conosciuto metodo del *EigenFaces*.

2.2.3 Considerazioni finali

Dopo aver elencato tutti i metodi, specificando di volta in volta vantaggi e svantaggi, verranno esposte ora alcune considerazioni di carattere generale, concentrandoci solo sulle tecniche ad elaborazione grafica.

La scelta di utilizzare una tecnica piuttosto che un'altra dipende da molti fattori. In primo luogo bisogna stabilire gli obiettivi da perseguire e i requisiti da richiedere al sistema. Stabiliti gli obiettivi, e ponderate le risorse disponibili (principalmente temporali, ma anche economiche) si sceglie il criterio con cui effettuare la scelta.

- Se l'obiettivo principale è realizzare un dispositivo il più possibile accurato, che permetta un'elevata mobilità del capo e che sia capace di rilevare un buon range di movimenti, la scelta ricade sicuramente sulle tecniche basate sulla luce infrarossa. Quest'ultime necessitano tuttavia di un significativo intervento hardware e di una rilevante quantità di risorse.

- Se invece l'obiettivo è ridurre i costi e ottenere un prodotto con un buon compromesso tra sforzi apportati e risultati ottenuti, la scelta adeguata può essere l'utilizzo delle tecniche basate su luce bianca.

Una volta scelto il tipo di luce con cui lavorare la scelta dei metodi non presenta una difficile decisione. Per quanto riguarda la luce infrarossa il tracciamento del rapporto tra la cornea e la riflessione della pupilla (sezione 2.2.1.1) è sicuramente il metodo migliore, considerando che tutti i dispositivi commerciali utilizzano questo metodo. Per quanto riguarda l'utilizzo di luce bianca la scelta ricade sul rilevamento del limbo (sezione 2.2.2.1).

Pare evidente che queste considerazioni siano state ponderate considerando un comune livello di realizzazione; si sottolinea come la bontà di un metodo rispetto ad un altro dipenda dalla fase realizzativa. Si presti attenzione anche al seguente commento: una combinazione di più metodi è la migliore scelta realizzativa possibile, risorse permettendo.

Un'ultima considerazione riguarda la distinzione tra valori continui e valori discreti. I valori discreti presentano un numero finito di riquadri in cui è possibile spostare lo sguardo. I valori continui invece assomigliano alla nota periferica mouse; tramite quest'ultimo il cursore può essere spostato in qualsiasi posizione all'interno dello schermo. Dalle pubblicazioni lette per realizzare questo capitolo non si evince questa distinzione. È quindi molto difficile intuire quali permettano di rilevare valori discreti e quali valori continui. A fronte dei movimenti oculari detti fissazioni[6], si ipotizza che la quasi totalità di realizzazioni prevedano valori discreti con molti riquadri. Questa scelta permette di aumentare sensibilmente la robustezza del metodo.

[6]Per approfondimenti si rimanda all'appendice (D.1).

2.3 Prodotti in commercio e software di supporto

Verranno ora elencati alcuni esempi di prodotti commerciali e di alcuni software di supporto. I primi vengono descritti per rendere l'idea di quale sia il livello raggiunti dai prodotti in commercio. I secondi per elencare le possibili estensioni da applicare ad un qualsiasi progetto.

2.3.1 Prodotti in commercio

Come già detto, i primi dispositivi commerciali nascono solo tra la fine degli anni '80 e l'inizio degli anni '90. Tali dispositivi vengono usati, principalmente, per l'assistenza ai disabili. Nel corso degli anni le tecnologie e i metodi si sono affinati al punto da poter offrire al mercato una varia gamma di prodotti.

Tali prodotti sono suddivisibili in due categorie principali: i dispositivi indossabili e i dispositivi fissi. Da quest'ultima categoria ne sono stati scelti quattro, accessibili nel mercato nazionale. I prezzi sono solo indicativi e sono soggetti a possibili cambiamenti. I prodotti sono:

- *TM3, TM4, VT1* : prodotti dalla *EyeTech Digital System* [32] con base in Arizona, USA. Sono tre tipologie poco differenti (figura 2.8) che si basano sul modello Dark Pupil quindi sulla proiezione di una luce infrarossa. Sono vendute con il solo blocco da collegare al PC o ad uno schermo. Richiede per il suo utilizzo una prima fase di calibrazione di qualche decina di secondi. Costo approssimativo di 10.000 €.

Figura 2.8: TM3, TM4, VT1. EyeTech Digital System.

- *EyeMax* : prodotto dalla *DynaVox* [33] : importante azienda americana, con sede a Pittsburgh, che commercializza tablet, prodotti e software per il supporto ai disabili. Il prodotto (figura 2.9) si presenta come uno schermo con integrata una fotocamera. Include molte funzionalità e molti software di supporto. Sfrutta sempre la stessa metodologia della luce ad infrarossi. Può essere usata anche da utenti con gli occhiali. Possibilità di noleggio. Il prodotto è definibile come portatile, in quanto include batterie ricaricabili. La calibrazione richiede una decina di secondi. Costo approssimativo di 10.000 €.

Figura 2.9: EyeMax. DynaVox.

- *Eyegaze Edge Desktop and Tablet*: prodotto dalla *LC Technologies Inc* [34] questo prodotto (figura 2.10) si presenta come uno schermo con una fotocamera posta al sotto di esso. La tecnica utilizzata si basa su luce ad infrarossi. La calibrazione richiede una quindicina di secondi. Una limitazione di questo dispositivo è la scarsa tolleranza ai movimenti della testa e la necessità che nella prossimità della telecamera non ci siano fonti di interferenza infrarosse come luci di elevata intensità. La LC technologies è stata la prima azienda a presentare nel settore commerciale un dispositivo di tracciamento dello sguardo. Costo approssimativo di 15.000 €.

Figura 2.10: Eyegaze Edge Desktop and Tablet. LC Technologies Inc.

- *MyTobii P10* : prodotto dalla *Tobii Technology* [13] società svedese che da anni si occupata di progettazione e realizzazione di hardware e software per dispositivi di tracciamento dello sguardo. Il prodotto mostrato in figura (2.11) presenta uno schermo 15" con telecamera integrata. Funziona anch'esso utilizzando tecnologia che si basa su luce ad infrarossi. Il software riesce a tenere traccia della testa fino ad un massimo di 10cm/s. Funziona bene in diverse condizioni di illuminazione e garantisce una precisione di stima del punto osservato di circa 0,5 cm. L'azienda offre un'ampia gamma di software e hardware di supporto. Sicuramente il migliore in commercio, richiede solo pochi secondi di calibrazione. Costo approssimativo di 20.000 €.

Figura 2.11: MyTobii P10. Tobii Technology.

Quasi tutti i dispositivi richiedono severe condizioni di luce ambientale, relativa vicinanza alla telecamera e che quest'ultima sia posizionata precisamente di fronte all'utilizzatore. Utenti affetti da strabismo e altre malattie potrebbero non essere compatibili. Solo alcuni di essi presentano versatilità nei movimenti del capo. Un riferimento speciale merita la ditta *SR Labs* [35] l'unica azienda commerciale italiana che opera in questo settore che ha collaborato alla ricerca e alla realizzazione dei dispositivi della Tobii Technology.

2.3.2 Software di supporto

Esistono in commercio soluzioni prettamente software studiate per venire incontro alle esigenze degli utenti con disabilità. Tali software sono pensati per essere integrati con dispositivi di tracciamento dello sguardo e sono esclusivamente di supporto. Verranno di seguito elencati i più comuni e utili software di supporto.

GazeTalk 5.0

GazeTalk [36] è sviluppato dall'Eye Gaze Interaction Group dell'Università di Copenhagen. È un software di supporto che si accompagna ad un dispositivo di tracciamento dello sguardo e permette l'inserimento di testo a tipologia predittiva. Basato su una tastiera con un ristretto numero di tasti, è stato pensato espressamente per l'utilizzo con sistemi di tracciamento del capo e dello sguardo. Il sistema di predizione e completamento della parola è basato sull'uso di un dizionario, caricabile in fase di installazione (disponibile anche la lingua italiana). La media di digitazione è di circa una decina di parole al minuto.

In sostituzione a questo software ne esistono altri che permettono di implementare facilmente dizionari di lingue diverse, molto simili al conosciuto T9, *Text on 9 keys*, implementato su molti cellulari, PDA e touch screen.

Un esempio di ciò è dato da *OnScreenKeys* [37], tecnologia assistenziale per persone con disabilità che consiste in una tastiera virtuale con comandi speciali, disponibile in oltre venti lingue.

Per il mio progetto è stato facilmente implementato un sistema analogo che verrà descritto nei capitoli successivi.

Dasher

Un esempio originale di software di supporto è dato da Dasher [38], sviluppato dalla Inference group della University of Cambridge. Il suo funzionamento ricorda molto i videogame classici degli anni '80, in cui bisognava al comando di una navicella distruggere gli asteroidi che si trovavano sulla propria rotta. La rotta veniva scelta attraverso la selezione di uno dei quattro tasti direzionali. Qui, invece, l'input prevede solo la scelta verticale. Sul video vengono proiettate delle lettere che compaiono secondo criteri di probabilità, come si può vedere dalla figura 2.12. Le lettere sono ordinate in ordine alfabetico ma occupano una porzione di schermo proporzionale alla loro probabilità statistica. Quelle ortograficamente scorrette sono più lontane e più piccole, mentre quelle più probabili vengono visualizzate vicine e più grandi. Inoltre l'interfaccia grafica presenta un gradevole design grafico.

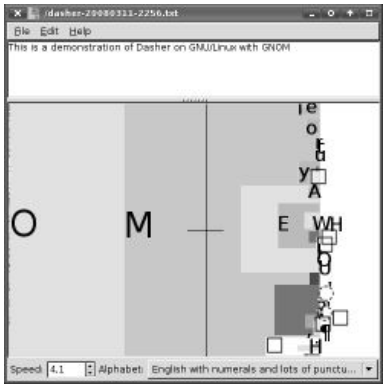


Figura 2.12: Dasher. Inference Group, University of Cambridge.

Questo software ha il grande vantaggio di non richiedere grande precisione

del sistema di tracciamento dello sguardo collegato, ma necessità solo la capacità di discriminare tra sopra e sotto. La media di digitazione è di circa una quindicina di parole al minuto, che salgono a venti con un pò d'abitudine, fino alle 30 parole al minuto proclamate dagli ideatori.

Integrated Speech

Integrated Speech and Gaze Control for Realistic Desktop Environments : questo progetto di Castellina, Corno e Pellegrino [39], si propone di realizzare un sistema multimodale, basato sull'integrazione di ingressi audio (parola) e video (sguardo) per l'interazione con un ambiente desktop reale. Questo software si integra ad un dispositivo di tracciamento dello sguardo completo. Una grammatica in tempo reale è generata per limitare il vocabolario vocale basandosi sulla zona fissata.

Questo progetto è un esempio di come applicazioni in diversi settori possono essere utilizzati per migliorare un qualsiasi dispositivo.

Interfaccia utente

Interfaccia utente basata su dispositivi per il tracciamento dello sguardo per sistemi di controllo ambientale: questa tesi di laurea magistrale di Luigi De Russis [40], svolta nella Facoltà di Ingegneria del Politecnico di Torino si propone la realizzazione di un'interfaccia utente per dispositivi di tracciamento dello sguardo.

Modelli e interfacce di comunicazione

Modelli e interfacce di comunicazione testuale per sistemi di Eye Tracking, è una tesi di laurea di Calderazzo Fausto [41]. Questa applicazione permette, interfacciandosi al sistema di tracciamento dello sguardo sviluppato precedentemente dal politecnico, di aiutare l'utente a formulare frasi di testo nella maniera più naturale e veloce possibile.

2.4 Aziende, consorzi e università che operano nel settore

Verrà mostrato l'elenco delle università e delle aziende che operano nel settore.

2.4.1 Network COGAIN

Se ci si prefigge lo scopo di ricercare aziende o università che lavorino in questo settore il primo passo da eseguire è contattare il network COGAIN. Questo consorzio fornisce tutte le informazioni necessarie a qualsiasi ricerca o lavoro svolta in questo ambito. Per questo motivo ne viene fornita di seguito una rapida descrizione.

COGAIN [42]: COmmunication by GAze INteraction è un network (associazione, consorzio) di aziende, enti e università (il Politecnico di Torino ne è entrato a far parte dal 2004) specializzate nel tracciamento dello sguardo e di tutte le possibili applicazioni relative al movimento degli occhi. Si presenta come leader europeo, e non, nella ricerca e nell'innovazione. Finanziato per anni dall'Unione Europea presenta il suo obiettivo sociale: "La rete si propone di riunire le migliori competenze europee nell'integrazione di computer con dispositivi di eye tracking, in un progetto di ricerca sulle tecnologie assistenziali per cittadini con difficoltà motorie".

Fornisce informazioni, elenchi e contatti riguardo:

- *Educational materials.* Questi documenti prendono il nome di deliverable e sono indicati dalla lettera "D" seguita da un numero. Escono, con scadenza annuale, nuovi paper con gli aggiornamenti sul settore riguardanti nuove metodologie, nuove realizzazioni e linee guida [43]. Sono documenti realizzati da COGAIN stesso; i quattro documenti più rilevanti presentano le linee guida da applicare nel progetto di un dispositivo di tracciamento dello sguardo [40]:

- "A Survey of Existing 'de facto' Standars and Systems of Environmental Control" fornisce la definizione di controllo ambientale, presentandone il rapporto con la disabilità e analizzando differenti metodi di interazione, tra cui il gaze control. Presenta, inoltre, anche alcuni prodotti commerciali e non, che possono essere considerati standard de-facto, evidenziando gli aspetti positivi e quelli negativi della loro adozione [44].
- "Draft Standars for gaze based environmental control" analizza alcune architetture di sistemi di controllo domotico basati su eye-tracking e propone una propria architettura. Infine, elenca una serie di requisiti necessari e consigliati che un'interfaccia utente basata su eye-tracking dovrebbe avere [45] .
- "User requirements report with observations of difficulties users are experiencing" espone la necessità di porre l'utente finale al centro degli obiettivi di COGAIN. Fornisce, inoltre, alcune informazioni su chi può utilizzare la tecnologia di eye-tracking. Infine, elenca qualche alternativa all'eye-tracking, indicando i vari aspetti positivi e negativi [46].
- "Report on features of the different systems and development needs" chiarisce alcune caratteristiche chiave propri dei sistemi di eye-tracking, sia per quanto riguarda la loro componente hardware che per quanto riguarda la componente software [47].

- *Eye Trackers for Assistive Technology and AAC* (Augmentative and Alternative Communication): elenco dei sistemi commerciali di gaze tracking che vengono impiegati come periferica di controllo di un computer o come ausili per la comunicazione delle persone con disabilità. (vedere anche Prodotti in commercio nella sezione 2.3.1). Un esempio viene dal EagleEyes Project del Boston College [48]: EagleEyes per-

mette alle persone di controllare il computer muovendo solo gli occhi e funziona attraverso cinque elettrodi posti sulla testa della persona. La Camera Mouse permette alle persone di controllare il computer muovendo la testa.

- *Eyetrackers for eye movement research, analysis and evaluation*: elenco di aziende, enti e università che trattano eye tracking, dove con eye tracking viene inteso come i movimenti dell'occhio e non dello sguardo (vedere differenza Eye Tracking vs Gaze Tracking 1.1). Si citano a titolo di esempio:
  - Assplied Science Laboratories [49]: Research Academic, Market, Military, Neuroscience, Usability, Virtual Environments.
  - Cambridge Research Systems [50]: applicazioni software (a pagamento) per la misura dei movimenti in tempo reale dell'occhio con strumenti oftalmici ed ottici.
  - ILAB [51]: è un insieme di funzioni Matlab®, realizzate dalla Northwestern University USA, per l'analisi della stima del movimento degli occhi.
- *Open source gaze tracking and freeware eye tracking*: questa lista contiene un elenco dei principali software low-cost, free o open source, per il gaze tracking. Contiene inoltre informazioni utili nella realizzazione di un software per il gaze tracking. Alcuni di loro sono rivolti alle persone con disabilità (sistemi di controllo oculare), mentre alcuni per eye e gaze tracking nati per la ricerca. Eccone alcuni:
  - ITU Gaze Tracker [52] : elaborato del University of Copenhagen, raccoglie una collezione di dispositivi che lavorano con hardware aggiunto o senza; ovvero una webcam integrata o una videocamera con il visore notturno e un'illuminazione ad infrarossi. Svolto

dallo stesso gruppo è il GazeTalk, di cui abbiamo già parlato nella sezione 2.3.2. Per approfondimento vedere il sito o la pubblicazione [53].

- openEyes [54]: fornisce progetti hardware e software utile per il tracciamento dei movimenti dell'occhio umano. Software open scritti in Matlab®, per eye tracking che utilizzano il metodo della dark-pupil, illuminata da infrarossi. La cvEyeTracker, lavoro già visto nella sezione sullo stato dell'arte (2.2.2.1), è una applicazione real-time di eye-tracking svolto tramite l'utilizzo dell'algoritmo Starburst, scritto in C [26].
- Opengazer [55]: software open-source per il gaze tracker svolto attraverso ordinarie webcams. Opengazer mira ad essere un software a basso costo in alternativa ai dispositivi commerciali basati su hardware costoso. Applicazione nativa per linux.
- TrackEye [31]: esempio di programma Real-Time per il tracciamento dello sguardo implementato in C++ utilizzando le librerie OpenCV. Questo lavoro viene ampiamente discusso nella sezione sullo stato dell'arte (2.2.2.3).
- myEye [56]: l'obiettivo di questo progetto è di sviluppare un software di gaze tracking per consentire alle persone con gravi disabilità motorie di utilizzare lo sguardo come dispositivo di input per interagire con un computer. Il software è gratuito (ma non Open-Source).

- *Open source and freeware eye movement analysis tools.* Elenco di software di supporto alla realizzazione di un dispositivo di gaze tracking. Ecco due importanti esempi:

  - ETU-Driver: driver universale di COGAIN. È stato sviluppato in passato da Oleg Špakov e si presenta come un livello che si pone

fra il driver vero e proprio di alcuni modelli di eye-tracker e le applicazioni di terze parti, al fine di permettere il loro utilizzo su eye-tracker di produttori differenti. In poche parole è un driver che aiuta gli sviluppatori a creare applicazioni indipendenti dal tracker e a fare delle fasi di test e simulazione off-line.

- OGAMA (OpenGazeAndMouseAnalyzer): un software open-source progettato per analizzare i movimenti oculari e il mouse in disegni di studio slideshowan.

Una particolare menzione a parte la merita l'università di Stanford che ha svolto parecchia ricerca su possibili software aggiuntivi utilizzabili tramite eye-tracking [57].

Per quanto concerne le *Università italiane*, quasi tutti i lavori e le pubblicazioni vengono svolte da tre università:

- Facoltà di Ingegneria, Politecnico di Torino, che con tutte le tesi e le pubblicazioni fatte, e l'appartenza al network COGAIN si posiziona di diritto a referente nazionale.

- DISI - Dipartimento di Ingegneria e scienze dell'informazione, Università degli Studi di Trento [58]. Molti sono i lavori svolti da questo dipartimento nell'ambito della Computer Vision. Eccone alcuni esempi:

  - Performance evaluation of an eye tracking system: questo progetto di eye-tracker si basa sulla tecnica Video-oculography. Questi metodi utilizzano una fotocamera digitale per riprendere la posizione dell'occhio e dei suoi movimenti.

  - Head Pose Estimation using OpenCV: In questo progetto è stata sviluppata un'applicazione che stima la posizione della testa umana; progetto molto utile nel contesto della Computer Vision.

- Eye tracking as an accessible assistive tool [59]: progetto di un gaze tracking utilizzando la tecnica della Chrominance [27]. È il progetto da cui si prende spunto per lavoro di questa tesi. Verrà spiegato in dettaglio più tardi.

- Department of Applied Electronics, University Roma TRE, autore del lavoro «A neural-based remote eye gaze tracker under natural head motion» [23].

Conclusioni Nel capitolo appena conclusosi sono state analizzate la maggior parte delle tecniche e dei metodi utilizzati nella creazione di dispositivi di tracciamento dello sguardo. Le tecniche sono state suddivise in categorie prima e in sottocategorie dopo, con lo scopo di facilitare la comprensione dell'argomento. È facile intuire come le realizzazioni presenti in commercio non presentino esclusivamente uno specifico metodo, ma come i migliori risultati si ottengano integrando più tecniche differenti per sopperire ai difetti che ciascun metodo presenta.

Un approfondito studio sullo stato dell'arte è fondamentale per la comprensione del metodo realizzato, descritto in questa tesi, in particolare in quale categoria esso si colloca e quali vantaggi e quali problematiche ci si aspetta di riscontrare nel progetto. Il capitolo appena descritto si lega così al capitolo successivo, dove verrà descritta minuziosamente la realizzazione del dispositivo di tracciamento dello sguardo realizzato.

Capitolo 3

Soluzione proposta

Introduzione In questo capitolo vengono descritti in dettaglio tutti i moduli del sistema realizzato. Vengono inizialmente richiamati i requisiti richiesti a tale sistema, per poi passare alla descrizione dei metodi utilizzati per la stima dello sguardo. L'ultima parte del capitolo descrive brevemente l'implementazione nel programma del dizionario predittivo.

3.1 Descrizione dei requisiti richiesti

Prima di analizzare i metodi utilizzati per la realizzazione del dispositivo di tracciamento dello sguardo verranno esposti i requisiti richiesti a tale sistema. A partire da uno stato dell'arte fatto negli anni passati (Glenstrup e Engell-Nielsen [6]) e dalle linee guida formulate nei vari report di COGAIN network [44, 45, 46, 47], sono state individuate le caratteristiche sulle quali si basa il dispositivo di tracciamento dello sguardo realizzato. Vengono di seguito elencate tali caratteristiche:

- ❑ Localizzare la direzione dello sguardo.
- ❑ Essere compatibile con tutti i soggetti.
- ❑ Non avere alcun contatto con l'utente (bassa invasività).

- ❑ Essere robusto a variazioni di luminosità.
- ❑ Non intralciare la vista dell'utente.
- ❑ Essere in possesso di una buona precisione, ovvero con una bassa percentuale di errore, nel rilevare il punto osservato. Essere quindi in grado di rilevare le minime variazioni di posizione degli occhi.
- ❑ Offrire una buona dinamica temporale e velocità di risposta (meglio se in possesso di una risposta in tempo reale).
- ❑ Essere facilmente esteso per la registrazione binoculare.
- ❑ Essere compatibile con movimenti (relativamente) ampi della testa.
- ❑ Non richiedere hardware aggiuntivo.
- ❑ Essere facilmente espandibile.

Un accenno merita anche il concetto di *usabilità*, definita dall'ISO (International Organisation for Standardisation) come l'efficacia, l'efficienza e la soddisfazione con le quali determinati utenti raggiungono determinati obiettivi in determinati contesti. In pratica essa definisce il grado di facilità e soddisfazione con cui l'interazione tra un essere umano e una macchina si compie. Un test di usabilità, come può esserlo l'esecuzione del programma a più utenti diversi, non è previsto in questo lavoro. Si è cercato tuttavia di pensare a miglioramenti e facilitazioni grafiche che accompagnino l'utilizzo del programma.

È da segnalare come dispositivi di tracciamento dello sguardo soffrano di alcuni svantaggi rispetto ad altri sistemi di puntamento, in particolare il cosiddetto "tocco di Mida": non è possibile stabilire quando l'utente sta guardando un punto intenzionalmente oppure sta semplicemente spostando lo sguardo lungo lo schermo. Non esiste cioè un modo per confermare l'intenzionalità. Per aggirare questo problema nel corso della tesi (sezione 3.3.2) verrà proposta una semplice soluzione.

3.2 Descrizione dell'approccio scelto

Questa tesi vuole migliorare il lavoro svolto da Armanini e Conci [59], dove si propone di realizzare un dispositivo di tracciamento dello sguardo basato su un classificatore a cascata, in grado di rilevare la zona osservata dall'utente in una griglia di celle 5×2. Questo lavoro si basa a suo volta sulla ricerca svolta da Kourkoutis, Panoulas e Hadjileontiadis [27].

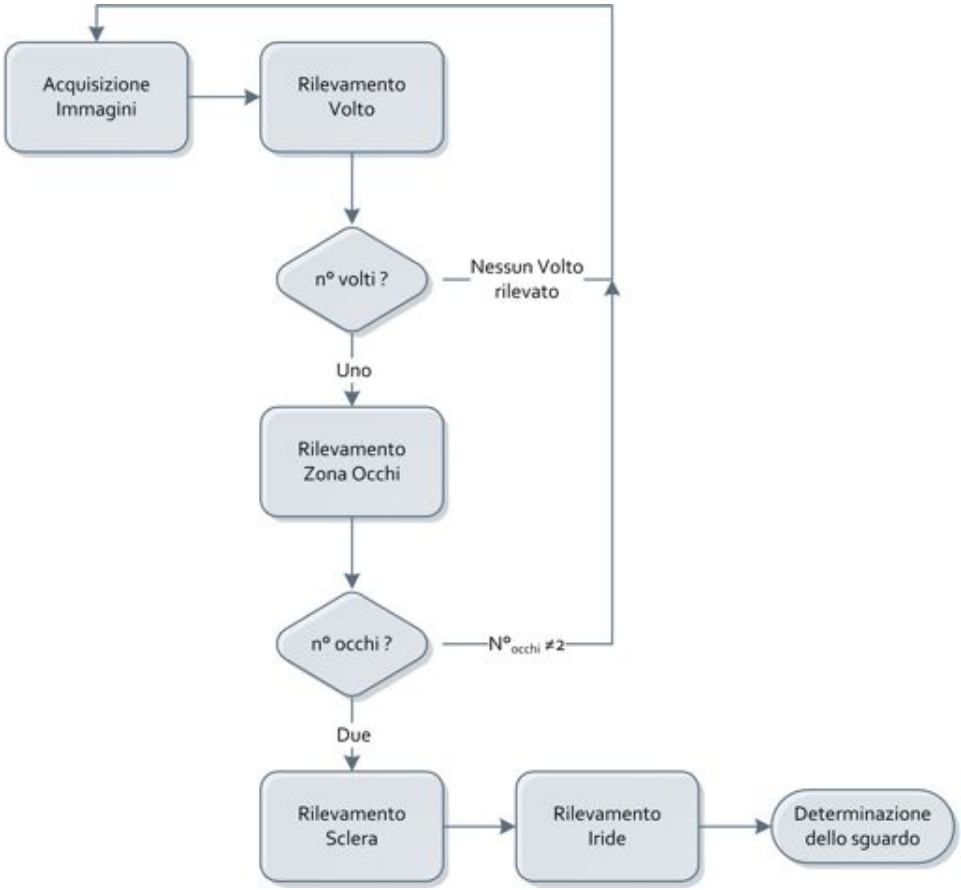


Figura 3.1: Schema a blocchi.

Proponendosi di realizzare un sistema con un costo accessibile, la soluzione utilizza una fotocamera che opera nello spettro visibile. L'algoritmo si colloca dunque nelle tecniche basate sulla luce bianca, descritte nella sezione 2.2.2. Queste soluzioni considerano il colore come fonte principale di

informazione e usano classificatori addestrati per discriminare modelli come labbra, bocca o sopracciglia. Come già accennato prima il sistema si basa sulla combinazione di diversi algoritmi a cascata. In figura 3.1 sono illustrate le diverse componenti.

Verranno ora illustrati nel dettaglio tutti i passi per la determinazione dello sguardo dell'utente che costituiscono il sistema.

3.2.1 Acquisizione dell'immagine

L'acquisizione delle immagini avviene servendosi dei metodi base di *OpenCV*[1].

Una scelta sostanziale riguarda l'utilizzo di fotocamera da cui ottenere le immagini. Le soluzioni possibili sono due: webcam integrata o webcam esterna. La realizzazione proposta da Armanini-Conci utilizza una webcam esterna da pc con risoluzione impostata a 800×600. In questa tesi si è scelto di utilizzare una webcam integrata. La scelta è ricaduta su questa tipologia per la volontà di creare un software totalmente indipendente dall'hardware e che potesse essere concettualmente trasferito su più dispositivi possibile.

Se nel caso di utilizzo di webcam integrata si dovesse realizzare l'indipendenza hardware, i vantaggi nell'utilizzo di una webcam esterna sarebbero molteplici. In prima istanza le webcam esterne presentano risoluzioni maggiori rispetto alle corrispettive integrate. Se la risoluzione della webcam integrata, disponibile sulla quasi totalità di pc, è 640×480 pixel, nelle webcam esterne si possono raggiungere facilmente e con poca spesa l'ordine dei mega pixel. Disporre di una risoluzione maggiore permette, com'è ovvio, di disporre di una maggiore quantità d'informazioni.

Altri vantaggi delle webcam esterne sono l'elaborazione automatica, tramite software interno, dell'immagine, per migliorare i livelli di luminosità e contrasto. Le webcam esterne inoltre possono vantare un'ottica più grande,

[1]Per un approfondimento sulle librerie OpenCV e sul linguaggio utilizzato nella realizzazione del sistema si rimanda all'appendice (B).

il che comporta un maggior flusso di luce entrante nella webcam, che, come vedremo in seguito, permette un aumento significativo delle prestazioni.

3.2.2 Rilevamento e tracciamento del volto

L'acquisizione dell'immagine fatta precedentemente restituisce un'immagine a colori di grandezza 640 × 480 pixel.

In figura 3.2, vengono rappresentati i passi fondamentali di questa fase.

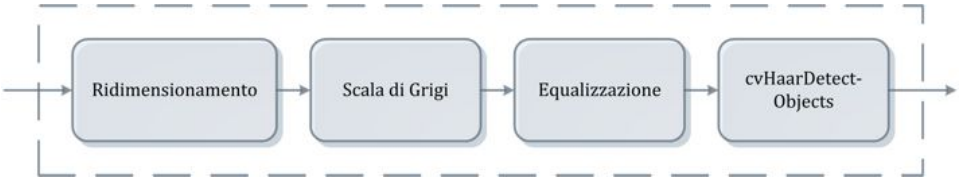


Figura 3.2: Passaggi per il rilevamento del volto.

L'immagine viene poi ridimensionata di un fattore 2, che la riduce a 320 × 240 pixel. L'operazione di ridimensionamento è applicata per ridurre il costo computazionale nella fase di rilevamento dei volti.

L'immagine, a questo punto, viene convertita in scala di grigio e successivamente viene applicata un'equalizzazione. Queste operazioni servono per preparare l'immagine alla classificazione del volto; tutte queste operazioni vengono svolte con l'utilizzo di OpenCV.

Ottenuta l'immagine ridimensionata ed equalizzata applichiamo il classificatore *AdaBoost*, addestrato con le *Haar-like Features*[2]. Il classificatore viene applicato tramite il metodo OpenCV: `cvHaarDetectObjects`. La ragione nell'utilizzo di Adaboost al posto di altri concorrenti, è dovuta principalmente alla sua idoneità per la classificazione in tempo reale, oltre al fatto che è già implementato in modo efficiente in OpenCV.

[2]Per un dettagliato approfondimento sull'algoritmo Adaboost e sulle Haar-like Features si rimanda all'appendice A.1.

```
1 CvSeq* faces = cvHaarDetectObjects( imgIn, cascade, storage,
2                       1.1, //increase search scale by 10% each
3                       3, //merge groups of three
4                       CV_HAAR_FIND_BIGGEST_OBJECT, //flags
5                       cvSize(40, 40) ); //smallest size face to detect
```

I quattro parametri significativi di questo metodo sono quelli accompagnati da commento e sono:

- *1.1*, riga 2°, è il fattore di moltiplica applicato, per ogni passo, alla scala.
- *3*, riga 3°, è il numero di rilevazioni necessarie, sulla stessa zona, per attribuire ad essa la certezza di un volto.
- *CV_HAAR_FIND_BIGGEST_OBJECT*, riga 4°, è un flag che permette di impostare il classificatore per la rilevazione di un solo volto, quello più grande. Tale impostazione permette di aumentare la velocità di rilevazione di un fattore 10. Per la rilevazione di più volti si utilizza il flag: CV_HAAR_DO_CANNY_PRUNING.
- *cvSize(40, 40)*, riga 5°, indica la dimensione minima del volto da rilevare.

L'uscita di questo algoritmo sarà il rettangolo con le coordinate del volto rilevato, (figura 3.3). Tale rettangolo, opportunamente scalato, sarà applicato all'immagine originale, quella non ridimensionata, per ottenere l'immagine contenente il solo volto. Il cascade[3] fornito al metodo è quello messo a disposizione da OpenCV: `haarcascade_frontalface_alt.xml`. Nella prima fase dello stage si è provato a costruire un cascade per la rilevazione del volto[4].

[3] Il termine *cascade* indica un file, con estensione XML, che raccoglie tutte le informazioni necessarie al rilevamento del volto. In altre parole rappresenta un file descrittivo delle caratteristiche proprie dell'oggetto da ricercare, come un volto o un occhio.

[4] I risultati sono illustrati nel capitolo relativo alla sperimentazione (3.4).

Figura 3.3: Ingresso ed uscita al rilevamento del volto.

Si può intuire come il rilevamento del volto produca ogni volta un rettangolo di dimensioni differenti anche con il volto fermo. Questo comporta una spiacevole visualizzazione a video di rettangoli sempre differenti. Per ovviare a questo problema si può pensare di far assumere al rettangolo solo valori discreti, per esempio multipli di cinque o dieci. Si lascia a sviluppi futuri questa correzione.

Inseguimento del volto

L'inseguimento, o tracciamento, di un volto umano nello spazio è uno dei tracking più difficili da realizzare e il lavoro svolto in questa ambito è notevole. OpenCV mette a disposizione diversi metodi per l'inseguimento di oggetti, utilizzando tecniche differenti.

L'algoritmo di tracciamento denominato *Camshift* utilizza le informazioni sul colore. Tramite una prima selezione della zona da inseguire vengono estratte le informazioni inerenti al colore e alla luminosità della pelle. Vengono poi cercate in ogni frame le medesime caratteristiche. È un algoritmo progettato per essere estremamente veloce e leggero, così che non richieda un alto costo computazionale.

Questo però non è l'unico tracking implementato in OpenCV. L'algoritmo *Lucas–Kanade* [60], che si occupa di registrazioni video a basso costo computazionale, viene ampiamente utilizzato nella computer vision, e permette di realizzare un tracking efficace [61]. OpenCV supporta tutti i metodi necessari

per una sua facile implementazione[5].

Per questo progetto invece si è scelta una strada differente: applicare il rilevamento del volto, frame per frame, tramite il metodo OpenCV menzionato sopra. Può sembrare una scelta poco comprensibile in quanto presenta molti svantaggi, come ad esempio un alto costo computazionale, oppure l'incapacità di tener traccia di un singolo volto in presenza di molti altri, o ancora, l'incapacità di rilevamento di volti troppo inclinati verso una spalla o volti di profilo. Si deve però tenere in conto l'utilizzo finale di questo tracciamento. L'obiettivo è tenere traccia di un solo volto, quello dell'utilizzatore, il che non crea problemi di riconoscimento nel caso di più volti, mentre il costo computazionale viene ridotto ridimensionando ed equalizzando l'immagine della webcam.

Si lascia il compito a sviluppi futuri integrare un vero e proprio tracciamento del volto in quanto in questo lavoro, anche per l'utenza a cui è destinato il programma, si considera che la testa sia tenuta ferma.

3.2.3 Rilevamento della zona degli occhi

Una volta localizzato il volto, avviene il rilevamento degli occhi; questo significa che viene rilevata la zona contenente l'occhio. I passaggi fondamentali sono illustrati in figura 3.4.

Il metodo per la detection è lo stesso usato per il rilevamento del volto: `cvHaarDetectObjects`. Questa volta sarà passato un cascade diverso, `haarcascade_eye.xml`, sempre messo a disposizione da OpenCV. Il metodo viene applicato sull'immagine contenente il solo volto. Nel codice del programma realizzato è possibile trovare due realizzazioni di questo algoritmo.

La prima realizzazione, in ordine temporale, per questo metodo prevedeva di applicare il metodo `cvHaarDetectObjects` all'intera immagine del volto (figura 3.5).

[5] Da segnalare come la realizzazione proposta da Armanini-Conci [59] utilizzi proprio questo algoritmo.

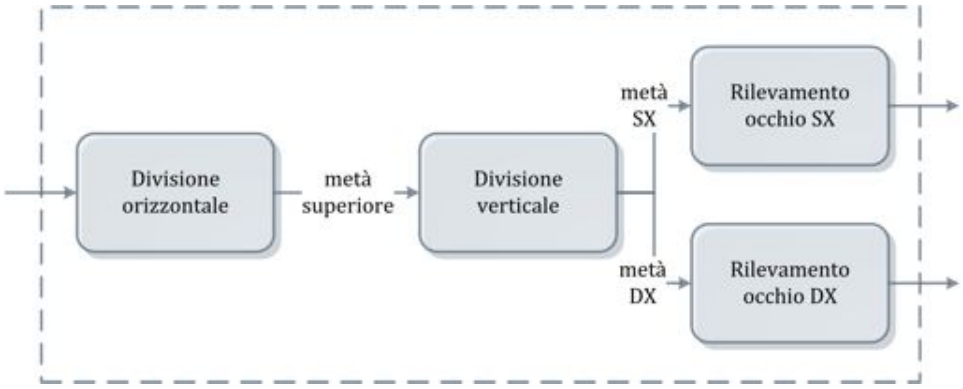


Figura 3.4: Rilevamento occhi implementato.

Impostando il flag CV_HAAR_DO_CANNY_PRUNING si riescono a rilevare più occhi dalla stessa immagine, ma poiché questo metodo si presentava meno performante[6] è stato abbandonato a favore di quello descritto di seguito.

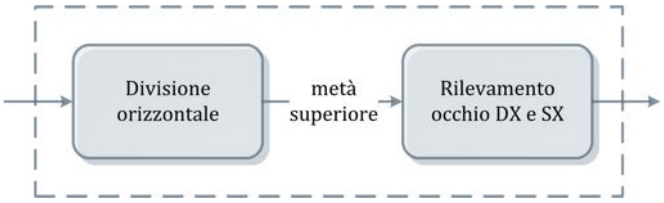


Figura 3.5: Rilevamento occhi inefficiente.

La seconda realizzazione, implementata nel codice, lavora come segue: l'immagine viene divisa verticalmente in due, la metà sinistra e la metà destra. Dalle due immagini ricavate vengono estratti, prima da una, poi dall'altra, i due occhi tramite il solito metodo di OpenCV. Il metodo restituisce poi i due rettangoli contenenti gli occhi. In questo caso la rilevazione avviene impostando il flag, CV_HAAR_FIND_BIGGEST_OBJECT, che, come già accennato, rileva solo il positivo (l'occhio) più grande nell'immagine passata. Quest'ultima implementazione permette di aumentare significativamente le prestazioni del rilevamento degli occhi.

[6]La rilevazione effettuata impostando il flag CANNY_PRUNING è meno performante rispetto alla rilevazione con flag BIGGEST_OBJECT di un fattore 10. Impostando quest'ultimo flag la rilevazione acquisisce un solo oggetto per volta, il più grande.

Si noti inoltre che come ingresso alle due realizzazioni non viene data l'immagine del volto intera, ma ne viene rimossa la parte inferiore, in quanto si suppone che gli occhi siano localizzati nella metà superiore dell'immagine. Si taglia in pratica la parte che con molta probabilità conterrà la bocca e la parte inferiore del naso, lasciando solo la parte più alta alla rilevazione. Questa operazione permette di sopperire alle limitate capacità del cascade.

Figura 3.6: Ingresso ed uscita al rilevamento della zona occhi.

Il dato in uscita a questo metodo sarà quindi composto dai due rettangoli contenenti la zona occhi (figura 3.6); tali rettangoli vengono scalati attraverso dei fattori trovati sperimentalmente. Questo permette di ottenere dei rettangoli che siano in rapporto con la dimensione del volto, fatto che agevola i passi successivi. Tali rettangoli vengono quindi utilizzati per ottenere due immagini, contenenti la sola zona occhi.

3.2.4 Rilevamento della sclera

Una volta ottenuta la zona contenente l'occhio bisogna estrarne la sola parte bianca, la sclera.

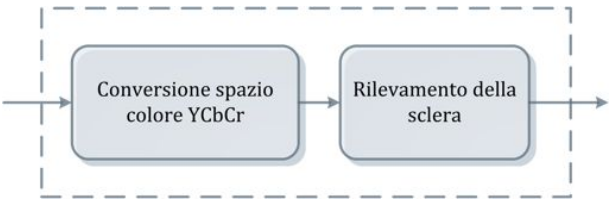


Figura 3.7: Rilevamento della sclera.

Questa procedura si propone di localizzare meglio l'area dell'occhio, e prepara l'algoritmo al riconoscimento dell'iride. In ingresso all'algoritmo per il rilevamento della sclera verrà quindi data l'immagine della zona dell'occhio catturato dal livello precedente e si avrà in uscita un rettangolo contenente la zona della sclera. I passaggi fondamentali sono illustrati in figura 3.7.

Tale procedura sfrutta le caratteristiche dell'immagine del volto nello spazio colore $YCbCr$[7]. L'algoritmo si basa sulla constatazione che la componente di crominanza Cr, differenza dal rosso, assume valori molto bassi nella zona della sclera, rispetto ai settori coperti da pelle. La sclera può essere individuata calcolando la somma dei valori di Cr su una finestra M $\times$ N, che viene spostata in ogni posizione possibile all'interno dell'immagine della zona dell'occhio. L'area con il più basso importo è indicata come la regione degli occhi.

Da notare come solo attraverso l'utilizzo dello spazio colore YCbCr sia possibile questa procedura. I pixel della pelle assumono valori elevati sia nella componente Cr dello spazio YCbCr sia nella componente R dello spazio RGB. La differenza tra i due spazi sta nella codifica del bianco. Tale colore comporta valori elevati nella componente R dello spazio RGB ma non incide minimamente sulla componente Cr dello spazio YCbCr. Questa caratteristica rende la componente Cr un'ottima candidata alla rilevazione di zone rosse, o come nel nostro caso rosa.

Per realizzare questa operazione vengono iterati più cicli nei quali un rettangolo viene spostato su tutte le zone dell'occhio; per ogni zona ne viene calcolata la somma di tutte le componenti Cr dei singoli pixel. La somma che otterrà il valore minore sarà la zona rilevata come sclera. Tale procedura è applicata per entrambi gli occhi.

Le immagini della zona occhi saranno pretrattate, prima di darle in ingresso all'algoritmo, convertendole in spazio colore YCbCr.

[7]Per approfondimenti si rimanda all'appendice (C.1).

I problemi riscontrabili in questa fase sono essenzialmente due. Il primo è la selezione della grandezza del rettangolo M × N. In particolare tale dimensione deve variare dinamicamente in base alla grandezza del volto; la grandezza del rettangolo sarà in rapporto con la grandezzza del rettangolo della zona dell'occhio trovato precedentemente (i valori del rapporto sono stati trovati per via sperimentale). Il secondo problema si presenta nel momento in cui l'utente chiude gli occhi. Con gli occhi chiusi il rilevamento di questo algoritmo seleziona zone sbagliate, la maggior parte delle volte la zona delle sopracciglia. Le scarse prestazioni del cascade fornito da OpenCV, utilizzato per il rilevamento della zona occhi, risolve intrinsecamente questo problema. Tale cascade infatti è stato allenato per rilevare solo occhi aperti e solo in casi rari vengono rilevati occhi chiusi.

Figura 3.8: Ingresso ed uscita al rilevamento della sclera.

In uscita a tale algoritmo ci saranno dunque due rettangoli, uno per occhio, posizionati sulle zone bianche dell'occhio (figura 3.8). Tali rettangoli verranno utilizzati in seguito per ottenere due immagini, una per occhio, contenenti la sola sclera.

3.2.5 Rilevamento dell'iride

Una volta localizzate le zone della sclera, l'ultimo passo sarà il rilevamento dell'iride, la parte colorata dell'occhio. Tale procedura è la più critica e quella che presenta più problematiche (la sequenza di operazioni viene mostrata in figura 3.9).

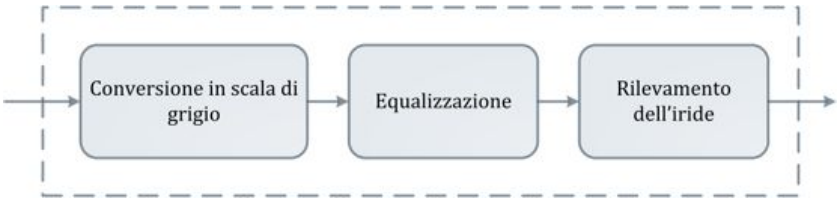


Figura 3.9: Rilevamento dell'iride.

La ricerca dell'iride ha luogo all'interno della zona dove vengono rilevate le sclere. Prima che vengano date in ingresso all'algortimo, le immagini delle sclere vengono pretrattate convertendole in scala di grigio ed equalizzandole. Al fine di trovare la posizione dell'iride, una finestra di dimensioni h $\times$ k è spostata sopra la regione delle sclere, calcolando la somma di luminanza al suo interno, nello stesso modo utilizzato per il rilevamento delle sclere. La somma che presenta valori di luminanza minori, ovvero più vicini al nero, sarà la zona dell'iride. La grandezza del rettangolo h $\times$ k sarà in rapporto con la grandezzza del rettangolo della zona della sclera trovato precedentemente. I valori del rapporto sono stati trovati sempre per via sperimentale. Ricercare nell'immagine un rettangolo al posto di un cerchio (naturale forma dell'iride) permette di ottenere un'approssimazione più che accettabile dell'occhio.

Questo algoritmo presenta un importante problema: le riflessioni. L'utilizzo del programma richiede che l'utente sia posizionato, il più vicino possibile, di fronte allo schermo. La vicinanza allo schermo ha il notevole vantaggio di illuminare uniformemente il volto, dato che facilità tutte le operazioni svolte finora. Presenta al contempo un importante svantaggio: la riflessione dello schermo sull'occhio. La riflessione, che si presenta sull'immagine come un piccolo rettangolo bianco nell'iride (figura 3.10), porta la somma dei vari pixel a valori più alti, causando nella pratica errori nella rilevazione dell'iride[8].

Si è cercato di ottenere nel sistema un buon compromesso tra vantaggi e

[8]È da segnalare che maggiore è la dimensione dello schermo più grande risulta essere la riflessione, causando di conseguenza più problemi.

Figura 3.10: Riflessione sull'iride dello schermo.

svantaggi; tuttavia i risultati sono poco apprezzabili, poiché l'illuminazione dello schermo rimane un elemento essenziale per la riuscita di tutti gli algoritmi visti precedentemente. Non potendo dunque variare la luminosità dello schermo si è cercato un metodo per eliminare, tramite elaborazione, la riflessione dello schermo sull'iride. L'utilizzo, per questo progetto, della webcam integrata ha comportato una bassa risoluzione dell'immagine e una cattiva gestione della luminosità. Applicando tutti i passaggi precedentemente illustrati fino a questo punto si ottiene un'immagine contenente la sclera non più grande di 60×30 pixel il che porta ad una rilevazione dell'iride non più grande di 20×20 pixel. Sarà facile intuire come ben poche soluzioni siano applicabili ad un immagine tanto piccola. Un qualsiasi sviluppo futuro, che vorrà migliorare le prestazioni di questo programma, non potrà escludere negli obiettivi la soluzione a tale problema.

Finora si è parlato solo di riflessioni provenienti dallo schermo, in quanto tutte le prove effettuate sono state condotte da seduto con schermo in posizione frontale e la sola illuminazione era artificiale e proveniva dal soffitto. È facile intuire come forti luci provenienti da zone laterali (come finestre e lampade da tavolo) creino a loro volta altre riflessioni ed altri problemi.

In uscita a tale algoritmo ci saranno dunque due rettangoli, uno per occhio, posizionati sulla zona scura dell'occhio (figura 3.11).

Figura 3.11: Ingresso ed uscita al rilevamento dell'iride.

3.2.6 Determinazione della direzione dello sguardo

Una volta acquisite tutte le informazioni necessarie, si passa alla stima dello sguardo. Le informazioni utilizzate nella stima sono le immagini a colori contenenti le sclere e i rettangoli contenenti le iridi estratte nelle fasi descritte precedentemente. La stima dello sguardo necessità, come verrà illustrato più tardi, di una fase di calibrazione.

La procedura di stima dello sguardo si basa sull'estrazione e sulla valutazione di sei *features* (caratteristiche). Queste features sono accorpabili in due grandi tipologie: *laterali* e *verticali*[9].

La scelta, come si vedrà, del conteggio dei valori dei pixel non poteva essere sostituita a metodi di valutazione di tipo geometrico, in quanto il rilevamento della zona della sclera è poco preciso.

3.2.6.1 Features laterali

Le features "laterali" sono due. Vengono estratte a partire dalle immagini contenenti la sola sclera, convertite in scala di grigi. Le features consistono nel calcolo della somma dei valori che i pixel assumono a lato dell'iride rilevato. Si ricorda che, essendo immagini in scala di grigi, i valori dei pixel possono assumere solo un valore intero compreso tra 0 e 255.

Le due features sono raccolte una sull'occhio sinistro e una sull'occhio destro in questo modo:

[9]Le valutazioni per stimare la qualità delle features, laterali e verticali, vengono fatte nella sezione "valutazione dell'efficacia delle features" (4.1.2).

(a) Feature laterale dell'occhio sinistro.

(b) Feature laterale dell'occhio destro.

Figura 3.12: Features laterali.

- sull'occhio sinistro viene calcolata la somma dei valori dei pixel (rettangolo rosso in figura 3.12a) a partire dal punto più a sinistra dell'immagine contenente la sola sclera, fino al punto più estremo a sinistra del rettangolo che rileva l'iride.

- Mentre sull'occhio destro viene calcolata la somma dei valori dei pixel (rettangolo rosso in figura 3.12b) a partire dal punto più estremo a destra del rettangolo che rileva l'iride, fino al punto più a destra dell'immagine contenente la sola sclera.

Ogni features non è altro che la somma così calcolata. Uno sguardo alla figura (3.12), chiarirà quanto appena scritto. Si può facilmente evincere che lo scopo delle features non è altro che la ricerca della zona bianca dell'occhio a destra o sinistra dell'iride.

Le features sono state rilevate nelle zone esterne dell'occhio, ovvero: la parte a sinistra dell'iride nell'occhio sinistro e la parte a destra dell'iride nell'occhio destro. Questa scelta deriva dalla loro maggiore grandezza rispetto alle corrispettive interne e, le zone esterne, sono anche meglio illuminate poiché meno in ombra.

Si ricordi inoltre come il rilevamento della zona della sclera consisteva nell'estrazione di un rettangolo. Facile intuire come un rettangolo mal approssimi un ovale com'è l'occhio. Rimangono dunque zone negli angoli contenenti aree di pelle del volto. Questo fatto non crea problemi nelle

estrazioni delle features, in quanto i valori dei pixel delle zone di pelle assumono valori molto bassi (intorno allo zero) rispetto ai valori molto alti (intorno a 255) nella zona della sclera.

È da notare inoltre, come il nome laterale non significhi che queste features siano utilizzate per la valutazione dello sguardo nei casi destra e sinistra, ma piuttosto perché vengono prese a lato dell'occhio.

3.2.6.2 Features verticali

Le features "verticali" sono in totale quattro: due per occhio. Vengono estratte a partire dalle immagini, a colori, contenenti la sola sclera. Le features consistono nel calcolo della somma dei valori che i pixel assumono al di sotto dell'iride rilevato. I valori che i pixel possono assumere sono tre numeri interi compresi tra 0 e 255, uno per canale (RGB).

(a) Occhio SX. **(b)** Occhio DX.

Figura 3.13: Feature verticale.

Per ogni occhio vengono estratte due features, attraverso la modalità di seguito riportata. Viene calcolata la somma dei valori di tutti i pixel presi a partire dal lato in basso del rettangolo contenente l'iride fino alla fine dell'immagine della sclera. Vengono calcolate due somme, ognuna corrispondente ai canali G e B. La figura 3.13, mostra la zona dove vengono calcolate le somme. Lo scopo delle features è, principalmente, di ricercare la zona bianca dell'occhio sotto l'iride, ma non solo. Un alto valore dei pixel dei canali B e G, nello spazio colore RGB, si ha non esclusivamente per valori di bianco ma anche per valori rispettivamente blu e verdi. Si ha

quindi una ricerca, non solo della zona bianca, peraltro molto ridotta, ma anche della parte colorata dell'occhio, l'iride. Questa strategia infatti è stata adottata perché compensa gli errori che potrebbero verificarsi nella stima della posizione dell'iride.

La scelta di escludere dal calcolo i pixel del canale R viene dalla necessità di non tener conto, nel calcolo della somma, dei pixel che contengono zone di pelle.

Il ragionamento può essere facilmente esteso anche ai pixel che si trovano al di sopra del rettangolo rilevato come iride. Questo apporterebbe un notevole vantaggio in termini di robustezza. Si lascia a sviluppi futuri una possibile integrazione.

La scelta di concentrarsi sulla zona al di sotto dell'iride trova un notevole riscontro, con relativo aumento delle prestazioni, nelle implementazioni con webcam esterna, posizionata al di sotto dello schermo.

3.2.6.3 Stima dello sguardo

Una volta trovate tutte e sei le features, non resta che stimare lo sguardo. Le sei features vengono estrapolate da ogni frame e ne viene mediato il numero con gli ultimi quindici valori prelevati. La scelta di mediare risponde alla necessità di ridurre gli errori nell'estrazione. I valori mediati delle features vengono poi confrontati con le soglie ricavate in fase di calibrazione.

Fase di calibrazione

La fase di calibrazione è la prima fase del programma. In questa procedura si invita l'utente a guardare ogni casella impressa sullo schermo. Ogni casella viene fissata per dieci secondi.

Nel frattempo vengono estratte, frame per frame, le sei features. La popolazione di misure che si viene a creare viene approssimata, per ogni

feature, ad una distribuzione normale[10] e ne viene calcolato il baricentro. Si ottengono quindi i valori dei baricentri delle sei features per ognuna delle dieci caselle. Tali valori saranno le soglie: i riferimenti con i quali effettuare i confronti nella fase di stima dello sguardo.

L'ultima fase di calibrazione effettuata viene salvata e può essere caricata all'inizio di ogni utilizzo. Una nuova calibrazione deve essere effettuata ad ogni cambiamento di condizioni ambientali, come ad esempio luce o distanza dalla webcam.

Fase di utilizzo

Nella fase denominata "di utilizzo", ovvero nella fase in cui c'è la stima vera e propria dello sguardo, avvengono i confronti con i valori presi nella fase di calibrazione.

All'interno del progetto di tesi, questi confronti sono stati fatti in un primo momento creando uno spazio a sei dimensioni, una per ogni feature. All'interno di questo spazio sono stati individuati dieci punti, che sono i baricentri ottenuti in fase di calibrazione. Ciascun punto corrisponde ad una casella. In fase di programma, le features estratte vengono mediate negli ultimi quindici valori. Le medie così ottenute rappresentano un punto nello spazio a sei dimensioni. Tale punto è stato quindi confrontato con tutti e dieci i punti ottenuti in fase di calibrazione, secondo questa formula:

$$dist = \sqrt{(F_1 - f_1)^2 + (F_2 - f_2)^2 + \ldots + (F_6 - f_6)^2}$$

dove $F_1, F_2, \ldots F_6$ rappresentano le sei features del punto n-esimo ottenuto nella fase di calibrazione, e dove $f_1, f_2, \ldots f_6$ rappresentano le sei features ottenute mediando gli ultimi quindici valori estratti dal programma. Il

[10] La valutazione della bontà di questa approssimazione verrà trattata nel capitolo dei risultati sperimentali (4.1.2).

confronto, rappresentativo della distanza tra due punti, con valore minore è stato utile per individuare la casella visualizzata in tale momento.

Questo tipo di implementazione ha rappresentato la scelta concettualmente migliore. Tuttavia è sorto un problema riguardo i valori delle features. Essendo infatti due tipologie differenti di features, i valori non sono dello stesso ordine di grandezza. L'impossibilità di ottenere una normalizzazione efficace ha provocato l'inefficienza del metodo di confronto.

Si è scelto quindi un metodo di confronto differente: valutare le features separatamente per tipologia. In questo modo vengono dapprima analizzate le features verticali. Le quattro features, due per occhio, vengono adibite alla determinazione della riga visualizzata. Vengono quindi utilizzate per discriminare i casi sopra e sotto. Si passa dunque ad uno spazio a quattro dimensioni, uno per feature, con solo due punti, che rappresentano sopra o sotto. Viene poi determinata la distanza del punto attuale da tali punti, e quella minore rappresenta la riga osservata dall'utente.

Una volta scelta la riga si passa a discriminare le colonne. Per questo compito si usano le features laterali utilizzate per discriminare i casi destra e sinistra. In fase di calibrazione vengono estratte cinque coppie di valori, uno per ogni colonna: ogni coppia è formata dalle due features laterali. Le features laterali formano quindi le coordinate di un punto in uno spazio a due dimensioni. Ogni punto è identificativo di ogni colonna. In fase di programma viene poi estratta la coppia di features, mediata negli ultimi quindici valori, che identifica il punto attuale. Tale punto viene confrontato con quelli acquisiti in fase di calibrazione. Il confronto che offre la distanza inferiore rappresenta la colonna attualmente guardata.

La casella è quindi scelta come combinazione della scelta delle righe e delle colonne.

La scelta di determinare prima le righe è essenziale, in quanto ad ogni riga corrispondono, per le colonne, valori diversi di features laterali. In

particolare le estensioni delle campane dei valori sono più larghe nella prima riga rispetto alla seconda, come descritto nella sezione 4.1.2. Esistono quindi, per le features laterali, due gruppi di valori, per ciascuna riga.

Pare subito evidente la poca eleganza del metodo. Si lascia tuttavia a sviluppi futuri riprendere la strada dello spazio a sei dimensioni, prevedendo un buon metodo di normalizzazione dei valori.

3.3 Dizionario

Nonostante la stima dello sguardo fosse l'obiettivo primario di questo progetto, si è scelto di spendere una considerevole quantità di tempo per realizzare un'applicazione che desse la capacità di scrittura utilizzando lo sguardo. L'obiettivo è stato quello di realizzare un dizionario di tipo predittivo. Si noti come l'applicazione che implementa il dizionario lavori in parallelo alla stima dello sguardo senza, dunque, modificarne le prestazioni.

Si elencano ora le caratteristiche del dizionario implementato:

- dizionario di lingua italiana, con possibilità di espansione per altre lingue.
- Dizionario predittivo, che consenta di prevedere la parola voluta dall'utente in base alle lettere già acquisite.
- Possibilità di scegliere un'altra parola dal dizionario, diversa da quella suggerita.
- Possibilità di cancellare l'ultima lettera inserita.
- Possibilità di cancellare l'ultima parola inserita.

Si è anche utilizzato uno spazio per implementare un'opzione che permette di uscire dal programma. Quest'opzione è stata introdotta perché ci si è resi conto, utilizzando il programma, che la tipologia di utente a cui è rivolto il

progetto difficilmente sarebbe stata in grado di premere un qualsiasi pulsante della tastiera.

Il dizionario è stato preso dal progetto di OpenOffice.org [62] e presenta un totale di 95191 parole. È possibile inoltre estendere il dizionario ad altre lingue attraverso i passaggi di seguito sintetizzati. Dopo aver scaricato dal sito il dizionario della lingua voluta, si devono riordinare tutte le parole in ordine crescente di lunghezza: prima quelle di una lettera (es. a,b,c), poi quelle di due lettere (es. ab,ad,ah), e così via. Tutte le parole con la stessa lunghezza devono essere disposte in ordine alfabetico. La scelta di ordinare in questo modo le parole è dovuta alla necessità di suggerire una parola, in fase di programma, che abbia un numero di lettere il più possibile vicino al numero di lettere già inserite. Questa affermazione sarà chiara dopo questo semplice esempio: nel caso che l'utente selezioni le lettere *p*, *e* e *r*, è indispensabile che il programma mi suggerisca *per* e non *perché*.

Si segnalano altri due possibili sviluppi futuri per quanto riguarda il dizionario. Il primo è sulle lettere accentate. Quest'ultime, essendo al di fuori dalla codifica ASCII originale, non vengono visualizzate a video correttamente per due motivazioni. La prima è la codifica imposta dal sistema operativo e la seconda è il metodo di OpenCV per illustrare a video dei caratteri. Mentre il secondo possibile sviluppo è l'inserimento di parole lettera per lettera, in quanto spesso il dizionario, nonostante la grande vastità di parole, non contiene parole di uso molto comune come ad esempio la parola casa.

3.3.1 Predittività

Poiché la predittività è un concetto chiave all'interno del progetto, si ritiene utile darne una breve esplicazione.

Tutte le operazioni vengono svolte su due liste: la prima contiene il dizionario italiano, opportunamente caricato nelle fasi iniziali, mentre la

seconda contiene tutte le possibilità in base alle lettere scelte, che chiameremo lista delle parole.

L'utente seleziona, tramite lo sguardo, una casella. Ad ogni casella (dieci in totale) corrispondono tre o quattro lettere. A questo punto viene aggiornata la lista delle parole, che contiene tutte le possibili parole ottenute combinando le parole già esistenti con le lettere della nuova selezione. Si veda per chiarezza il seguente esempio.

Si ipotizzi di selezionare in progressione le caselle 2, 4, 5. A queste caselle corrispondono rispettivamente le lettere: *[r,t,y]* per la casella 2, le lettere *[a,s,p,à]* per la casella 4, le lettere *[d,g,f]* per la casella 5. A questo punto la lista delle parole è aggiornata come illustrato in tabella 3.14.

Casella	2	4	5
lista delle parole	r		rad
		ra	rag
			raf
			rsd
		rs	rsg
			rsf
			rpd
		rp	rpg
			rpf
			ràd
		rà	ràg
			ràf
	t	ta	...
		ts	...
		tp	...
		tà	...
	y	...	...

Figura 3.14: Situazione della lista delle parole dopo tre inserimenti.

Si noti come la disposizione delle lettere in ogni casella non sia in ordine alfabetico, ma in ordine di probabilità d'uscita. Un esempio: la casella numero 1 presenta le lettere *[q,w,e]*, oltre le accentate. Pare evidente che la lettera che con più probabilità d'uscita sia la lettera *e*, seguita dalla *q*,

seguita dalla *w*. L'inserimento nella lista delle parole sarà dunque per indice di probabilità, nell'esempio *[e,q,w]*. Tutti gli indici di probabilità sono stati realizzati scansionando e conteggiando lettera per lettera un noto romanzo italiano[11], in modo da ottenere un indice di ricorrenza per ogni lettera.

La lista delle parole contiene però tutte le possibili parole, quindi oltre a quella voluta anche quelle che non trovano riscontro in italiano, come nell'esempio *rsg*. Per scoprire quale parola si voleva inserire bisogna aggiornare il dizionario. Per ogni casella selezionata il dizionario viene aggiornato eliminando tutte le parole non possibili in base alle selezioni fatte. Per far ciò si confrontano il dizionario e la lista delle parole. Tutte le parole del dizionario che non esistono all'interno della lista delle parole vengono eliminate.

La ricerca termina nel momento in cui si verifica una delle seguenti possibilità:

- quando viene selezionata la casella "Spazio" (avviene quando l'utente decide che la parola proposta a video è la parola voluta).
- Quando nel dizionario rimane solo una parola, che per esclusione viene eletta a parola voluta.

Ad ogni casella inserita viene visualizzata a video la parola che con più probabilità è la parola voluta dall'utente e cioè la prima nel dizionario (notare come il dizionario debba essere riordinato dinamicamente in base all'ordine imposto dalla lista delle parole). Ogni volta invece che la ricerca finisce decretando la parola voluta il dizionario viene ricaricato, mentre la lista delle parole viene svuotata, pronta per accogliere nuove combinazioni.

3.3.2 Eliminare l'ultima lettera

Per eliminare l'ultima lettera inserita, si è proceduto come segue. Una volta dato l'apposito comando vengono fatte due cose: la prima è aggior-

[11] "Il nome della rosa" scritto da Umberto Eco nel 1980.

nare la lista delle parole riportandola alla situazione precedente all'ultimo inserimento, la seconda è ripristinare il vecchio dizionario. Per quest'ultima operazione si è scelto di tenere in memoria tutti i vecchi dizionari, ovvero un dizionario per lettera dell'ultima parola. Si è scelta questa via che, anche se non ottimizza l'utilizzo di memoria, evita di ripetere tutti i confronti da capo causando un notevole rallentamento nelle prestazioni. Questo opzione ha risolto anche il problema del "tocco di Mida", precedentemente analizzato, ovvero l'impossibilità di verificare l'intenzionalità della selezione.

Conclusioni In questo capitolo è stata affrontata in maniera completa ed esaustiva la descrizione dei metodi utilizzati nella realizzazione del progetto di stage. In particolare sono stati descritti con cura, e nella maniera più chiara possibile, tutti i livelli che compongono l'algoritmo a cascata realizzato.

Per valutare le prestazioni del programma viene nel capitolo successivo lasciato spazio ai risultati di una fase di sperimentazione creata su misura. Verranno redatti i risultati di una fase di test svolti su un cascade generato per il riconoscimento del volto e verrà data una valutazione dell'efficacia delle features utilizzate nella stima dello sguardo. Un'ultima importante parte verrà lasciata all'analisi della raggiungibilità dei requisiti fissati all'inizio del progetto e del loro eventuale grado di scostamento.

Capitolo 4

Sperimentazione e Risultati

Introduzione In questo capitolo vengono mostrati i risultati della fase di sperimentazione del programma, realizzato attraverso i metodi ampiamente descritti nel capitolo precedente.

Nella prima fase di sperimentazione si è testato il cascade[1] generato per la fase di riconoscimento del volto. Nella seconda invece si è andato ad analizzare l'efficacia delle features utilizzate per la stima dello sguardo.

Nell'ultima parte vengono richiamati gli obiettivi stabiliti all'inizio del progetto e ne viene analizzata la raggiungibilità e il grado di scostamento.

4.1 Descrizione tecnica e commento critico dei risultati prodotti

La stima dell'efficacia dei risultati prodotti passa attraverso la valutazione di due fattori:

- l'efficacia del file XML generato per il rilevamento del volto;
- l'efficacia delle features.

[1] Il termine *cascade* indica un file, con estensione XML, che raccoglie tutte le informazioni necessarie al rilevamento del volto. In altre parole rappresenta un file descrittivo delle caratteristiche proprie dell'oggetto da ricercare, come un volto o un occhio.

Verranno affrontati entrambi gli aspetti nelle sottosezioni che seguono.

4.1.1 Generazione del file *XML*

Verranno ora riassunti tutti i comandi utilizzati per la generazione del file di estensione XML. Tale file è un parametro necessario da passare al metodo di rilevamento del volto, discusso nella sezione 3.2.2[2].

```
1  //Mettersi nella cartella:
2  cd C:/Users/Michele/Desktop/Database/Definitivo
3
4  //Dare il comando per generare il positive.dat e il negative.dat:
5  find ./positivi -name '*.ppm' -exec echo {} 1 0 0 100 100 > positive.dat
6  find ./negativi -name '*.jpg' > negative.dat
7
8  //Dare il comando per generare il positive.vec:
9  /cygdrive/C/OpenCV2.0/bin/opencvcreatesamples.exe -info positive.dat -vec positive.
       vec -w 20 -h 20 -num 5974
10
11 //Dare il comando haartraining
12 /cygdrive/C/OpenCV2.0/bin/opencvhaartraining.exe -data ./haarcascade -vec positive.
       vec -bg negative.dat -nstages 20 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5
        -npos 5974 -nneg 4398 -w 20 -h 20 -nonsym -mem 512 -mode ALL
```

Per velocizzare le operazioni è possibile utilizzare le librerie *IPP*. Le IPP, *Intel Integrated Performance Primitives*, sono librerie per l'utilizzo del multi-core nella elaborazione grafica di file multimediali. Essendo proprietarie è possibile sostituirle con le librerie *OpenMP*. *OpenMP* (Open Multi-Processing) è un *API* (Application Program Interface) per la programmazione parallela in ambienti a memoria condivisa. *OpenMP* è una combinazione di direttive per il compilatore, funzioni di libreria e variabili d'ambiente che permettono di esprimere il parallelismo in linguaggi come Fortran, C e C++[3]. Per poter utilizzare il multi processore occorre ricompilare il sorgente del `haartraining` abilitando l'opzione di compilazione OpenMP.

[2]Per la descrizione completa del processo si rimanda all'appendice (A.2).
[3]Per ulteriori informazioni visitare i siti: [63] e [64].

Attraverso l'utilizzo delle OpenMP è stato possibile ridurre notevolmente i tempi di calcolo della fase di allenamento: da più di una settimana a single-core si è passati a poco più di un giorno con il multi-core. In questa fase è stato utilizzato un Mac Pro 4.1 Quad-core Intel Xenon 2.66 GHz con Memory 3GB.

Per il *Training* sono state utilizzate 5974 immagini positive (volti), prese e ritagliate dal *FERET Database*[4], e 4398 immagini negative contenenti paesaggi o altro prese dal web.

Tutto il processo è durato 28 ore. Verranno adesso mostrati i risultati ottenuti.

4.1.1.1 Descrizione dei risultati ottenuti

Dopo aver generato il cascade (il file XML) è stata eseguita la fase di test. Il cascade generato è stato testato tramite un programma creato su misura, con il quale è stato possibile impostare tutti i parametri a piacere. Il programma aveva il compito di caricare un'immagine e di rilevarne i volti all'interno; venivano poi conteggiati i volti rilevati e i falsi rilevamenti.

(a) Immagine *test01.png* [65].

(b) Immagine *test02.png* [66].

(c) Immagine *test03.png* [67].

Figura 4.1: Immagini del test.

Le prove sono state svolte utilizzando tre diverse immagini trovate sul web. Queste immagini, mostrate figura (4.1), sono state scelte per l'elevata presenza di persone (quindi di volti) da rilevare, rispettivamente in numero pari a: 204 (4.1a), 61 (4.1b) e 59 (4.1c).

[4]Per approfondimenti si rimanda all'appendice (E).

Verranno ora presentati i risultati dei test, eseguiti su due cascade differenti:

- il cascade fornito da OpenCV di nome `haarcascade_frontalface_alt.xml`;
- il cascade generato con le operazioni descritte nella sezione precedente.

La tabella 4.2 contiene i risultati dei test effettuati sulle tre immagini utilizzando il migliore **cascade** di **OpenCV**. Altri cascade sono messi a disposizione nelle librerie, `haarcascade_frontalface_default.xml` e `haarcascade_frontalface_alt_tree.xml`, ma entrambi offrono risultati nel complesso peggiori rispetto a quello illustrato.

Nome	*Numbers detection*	*Numero Volti Reali*	*% Volti Rilevati*	*% False Rilevazioni*
test01.jpg	2	204	86,27	7,85
	3	204	82,35	3,45
test02.jpg	2	61	93,44	3,39
	3	61	88,52	0,00
test03.jpg	2	59	94,92	6,67
	3	59	89,83	1,85

Figura 4.2: Risultati dei test sul cascade OpenCV.

Quello che si evince dalla tabella è la bontà dei risultati ottenuti; la percentuale di volti rilevati è alta e si assesta mediamente attorno al 85-90% del totale dei volti presenti, mentre poche sono le false rilevazioni, intorno al 5% dei volti rilevati.

La colonna *Numbers detection* rappresenta il numero di rilevazioni necessarie, sulla stessa zona, per attribuire ad essa la presenza di un volto. Facile intuire come, impostando tale parametro prima con il valore due e poi con

il valore tre, il numero complessivo di rilevamenti diminuisca, richiedendo più rilevazioni per l'attribuzione della zona a volto, comportando una diminuzione sia in termini di volti trovati che di false rilevazioni. In fase di sviluppo questo parametro è stato impostato col valore tre[5]; questa scelta permette di ottenere un numero di false rilevazioni minore, obiettivo primario per il programma in quanto non sono previsti controlli in tal senso. Non prevedendo controlli per riscontrare falsi rilevamenti, la percentuale di falsi rilevamenti incide maggiormente rispetto alla percentuale di volti rilevati, in quanto nel caso non venga rilevato un volto si può ripetere facilmente il rilevamento.

Nome	*Numbers detection*	*Numero Volti Reali*	*% Volti Rilevati*	*% False Rilevazioni*
test01.jpg	2	204	60,29	15,17
	3	204	45,10	8,91
test02.jpg	2	61	78,69	9,43
	3	61	63,93	0,00
test03.jpg	2	59	71,19	6,67
	3	59	66,10	2,50

Figura 4.3: Risultati del test sul cascade realizzato.

In tabella 4.3, si possono vedere i risultati dei test effettuati con il **cascade generato**. Viene spontaneo intuire come i risultati siano peggiori del cascade testato in precedenza, sia in termini di percentuale di volti rilevati, intorno al 65-70%, che in termini di false rilevazioni, con percentuali intorno al 10%.

Le spiegazioni possono essere molteplici. La realizzazione del cascade non rientrava negli obiettivi prefissi in questa tesi, quindi poco tempo è stato speso a riguardo. La limitata quantità di tempo a disposizione per realizzare

[5]Come si può notare nella sezione 3.2.2.

il cascade non era paragonabile alla difficoltà di ottenere risultati migliori rispetto a quelli ottenuti ed offerti dagli sviluppatori delle librerie.

In entrambe le tabelle si può notare come le immagini *test02.jpg* e *test03.jpg* abbiamo risultati simili, differenti dai risultati sull'immagine *test01.jpg*. La motivazione va ricercata nella differenza di posa che i soggetti assumono nella prima immagine, dove la fotografia è stata scattata dall'alto e non frontalmente.

Avendo ottenuto risultati così poco incoraggianti si è scelto di non procedere a realizzare un cascade per gli occhi. Si è scelto dunque di utilizzare il cascade offerto dalle librerie di OpenCV, sia per il rilevamento del volto sia per il rilevamento della zona degli occhi. Si lascia a sviluppi futuri realizzare un cascade migliore, sia per il volto, ma soprattutto per gli occhi.

4.1.2 Valutazione dell'efficacia delle features

Come esposto precedentemente le features sono sei. In questa sezione si è cercato di fornirne una valutazione il più possibile completa, per stabilirne quanto più possibile l'efficacia.

Per farlo sono stati monitorati i valori di ognuna delle sei features mentre l'utente guardava ogni casella. La cattura dei dati è durata venti secondi per ognuna delle dieci caselle[6]. Le prove sono state effettuate in una stanza senza finestre con illuminazione d'ufficio media. Tutti i valori sono stati salvati ed esposti in grafici attraverso l'utilizzo del software Matlab® con i comandi illustrati di seguito.

```
1 textread('.\valoriSalvati.txt','','delimiter','\n');
2 hist(colonnaValori,400)
3 set(gcf,'NumberTitle','off')
4 set(gcf,'Name','NomeGrafico')
```

[6]La procedura eseguita è simile alla fase di calibrazione (sezione 3.2.6.3).

Tutti gli istogrammi sono stati realizzati con intervalli discreti (bins o classi), riga 2, di grandezza pari a 400.

I grafici, realizzati come istogrammi, appartengono a due categorie in base alla tipologia di webcam da cui sono stati estratti: esterna o integrata. Ad ogni istogramma si accompagna un ulteriore grafico dove i valori estrapolati vengono approssimati da una distribuzione normale.

Verranno prima esposte le curve che rappresentano le features ricavate della webcam integrata.

Webcam integrata

Il primo istogramma (figura 4.4) rappresenta la *feature laterale*, estratta dall'occhio sinistro e corrispondente alla prima riga.

La prima cosa che si nota è la presenza di cinque gruppi di valori approssimabili a distribuzioni normali, o curve a campana, come si può osservare in figura 4.5. Le cinque campane corrispondono alle cinque caselle della prima riga. Si può subito notare come, anche se le campane si toccano, esse siano ben separate l'una dall'altra. In particolare il baricentro di ogni campana è ben distinguibile rispetto ad un altro. Si noti come i valori utilizzati per le soglie siano i punti medi tra baricentri, che ben approssimano i valori di intersezioni delle gaussiane associate. L'altra feature laterale, quella dell'occhio destro, ha un andamento molto simile a questa. Stesso discorso vale per le features della seconda riga.

Il secondo istogramma (figura 4.6) rappresenta invece la *feature verticale*, del pixel G, sempre dell'occhio sinistro. In figura 4.7, vengono illustrate le distribuzioni estratte.

La feature verticale è stata utilizzata per discriminare le righe, ovvero i casi sopra e sotto. I due colori stanno quindi ad indicare le due righe. Quello che emerge subito è che in questo caso la precisione vista precedentemente non è stata la stessa.

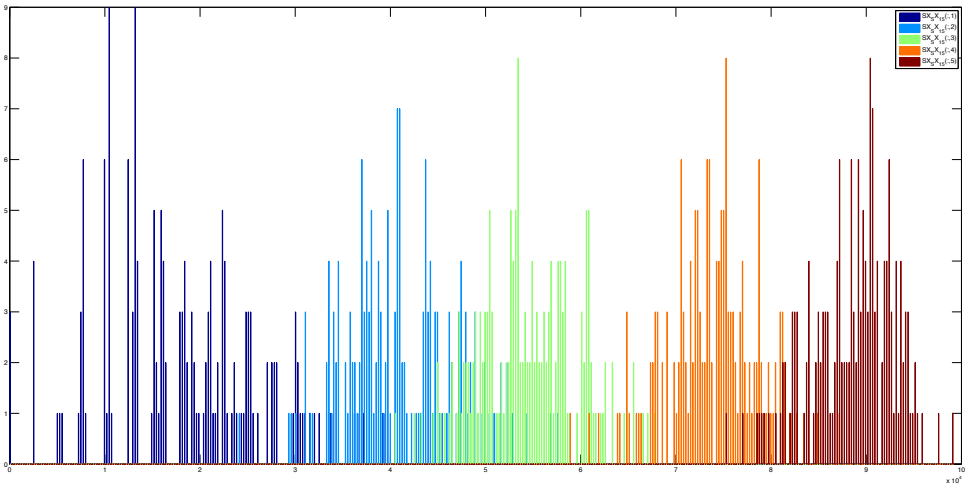


Figura 4.4: Valutazione della feature laterale sulla prima riga dell'occhio sinistro. Webcam integrata.

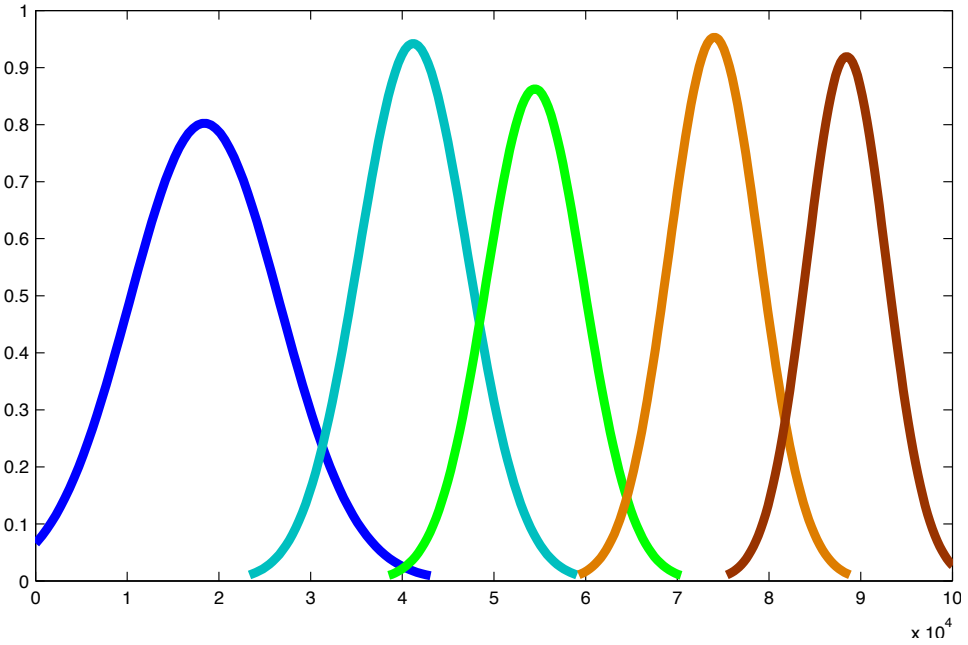


Figura 4.5: Distribuzione della feature laterale sulla prima riga dell'occhio sinistro. Webcam integrata.

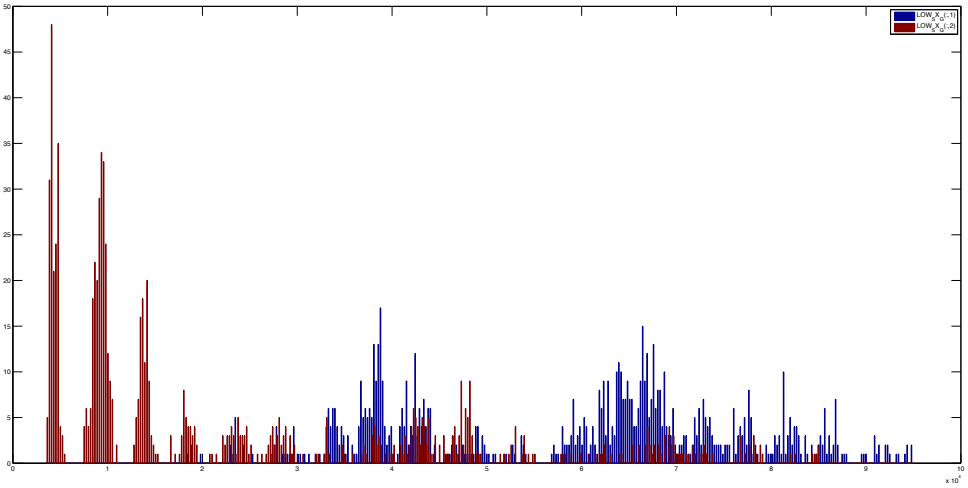


Figura 4.6: Valutazione della feature verticale, del pixel G sull'occhio sinistro. Webcam integrata.

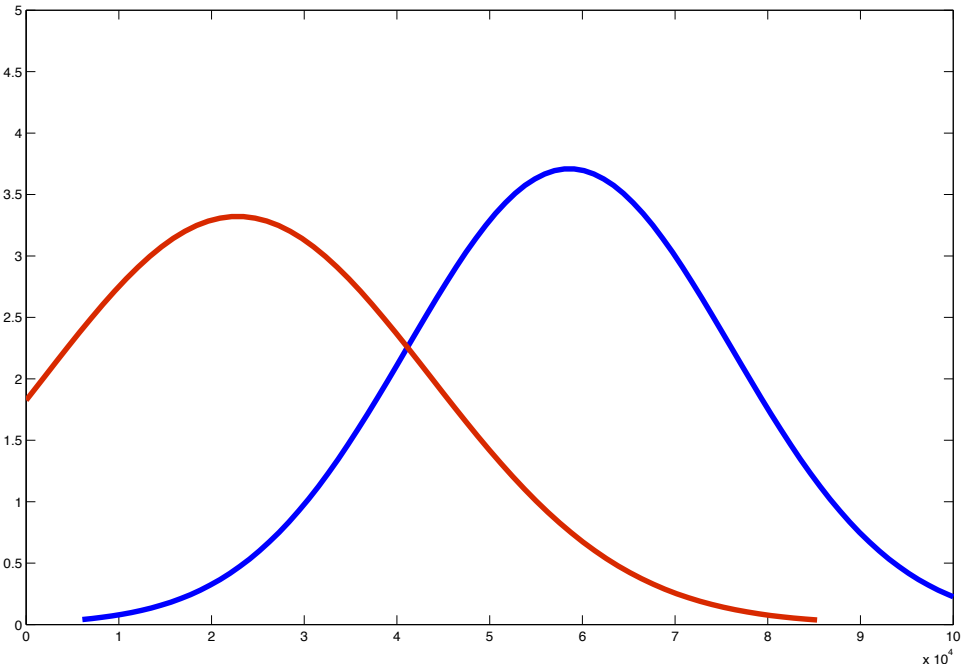


Figura 4.7: Distribuzione della feature verticale, del pixel G sull'occhio sinistro. Webcam integrata.

Si nota subito che un'approssimazione a distribuzione normale è alquanto forzata. Inoltre i due gruppi di valori si intersecano parecchio. Nonostante questo, i valori medi delle due popolazioni sono ancora parecchio distanti, il che rende accettabile la tecnica. Tale istogramma si ripresenta nella stessa forma anche per le altre tre features verticali.

In figura 4.6, è stata illustrata la feature verticale su entrambe le righe, contraddistinte da due colori. Per capire come le caselle di ogni riga si distribuiscono, si è scelto di rappresentare gli istogrammi delle features laterali, dei pixel G, della prima e della seconda riga, sempre dell'occhio sinistro.

L'istrogramma della prima riga (figura 4.8) indica come la feature verticale, del pixel G, si distribuisce nelle cinque caselle della prima riga. In figura 4.9, vengono illustrate le distribuzioni estratte corrispondenti. Subito si nota come nonostante siano individuabili delle campane, esse sono sovrapposte e quindi poco distinguibili l'una dall'altra.

L'istogramma della seconda riga (figura 4.10), con relative distribuzioni (figura 4.11), non presenta risultati migliori. In questo caso non è possibile individuare le campane; tutti i valori invece si posizionano intorno a valori più bassi. Stesso discorso vale per la feature che utilizza l'altro pixel e lo stesso vale per le due features dell'occhio destro.

Osservando questi grafici si intuiscono le motivazioni che hanno portato alla scelta di discriminare solo dieci caselle. È stato provato nella fase di programmazione l'inserimento di un'ulteriore riga, per un totale di tre. Si è ottenuto però un risultato piuttosto scadente dovuto principalmente alla bassa efficacia delle features verticali utilizzate. Con l'utilizzo della webcam esterna non si sono rilevati significativi miglioramenti. Si lascia a sviluppi futuri l'introduzione di una terza riga, a patto di introdurre features più efficaci.

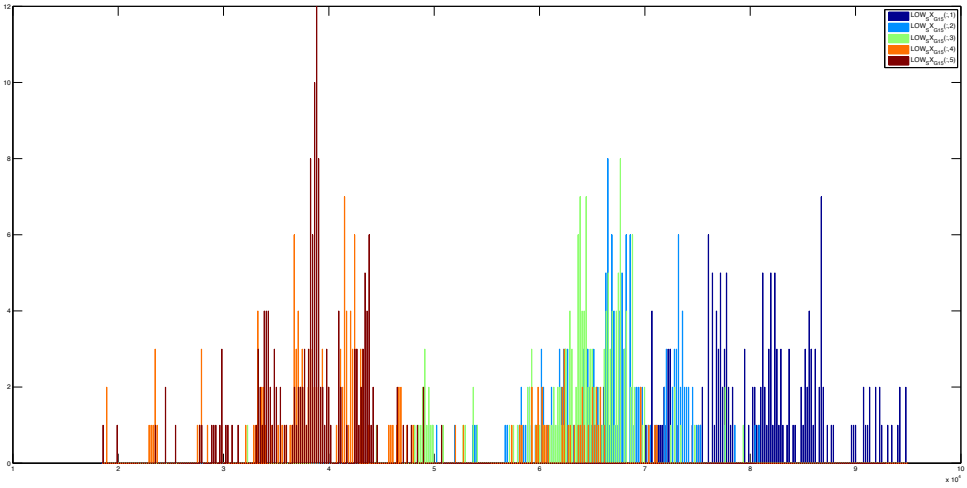


Figura 4.8: Valutazione della feature verticale sulla prima riga, del pixel G sull'occhio sinistro. Webcam integrata.

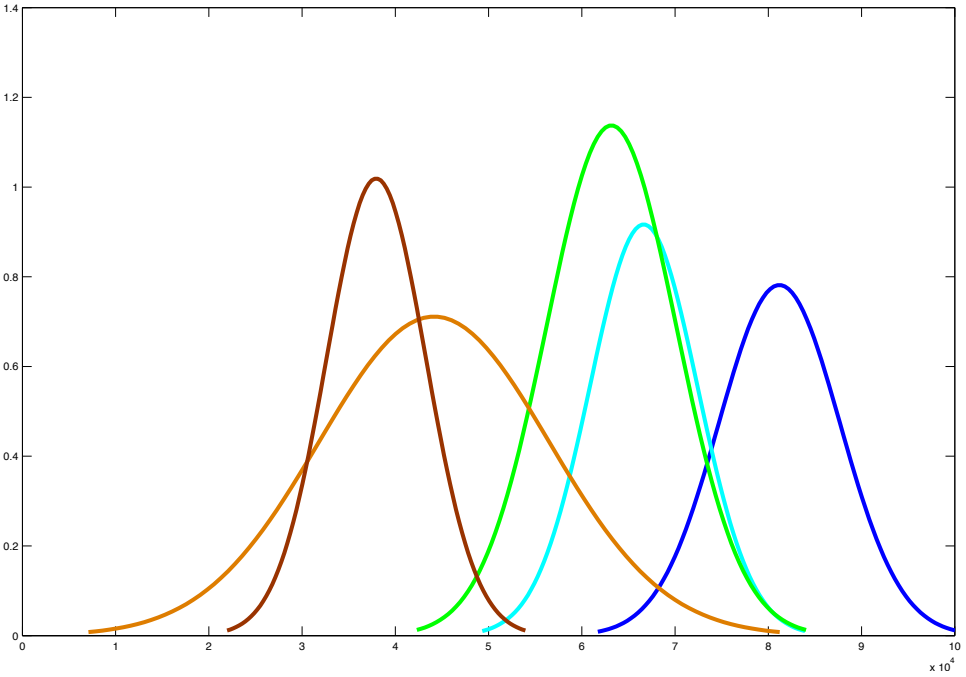


Figura 4.9: Distribuzione della feature verticale sulla prima riga, del pixel G sull'occhio sinistro. Webcam integrata.

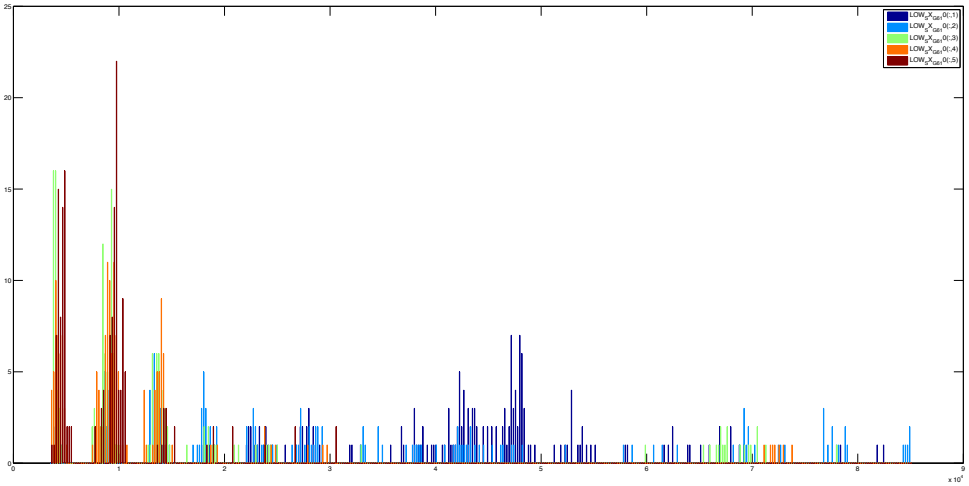


Figura 4.10: Valutazione della feature verticale sulla seconda riga, del pixel G sull'occhio sinistro. Webcam integrata.

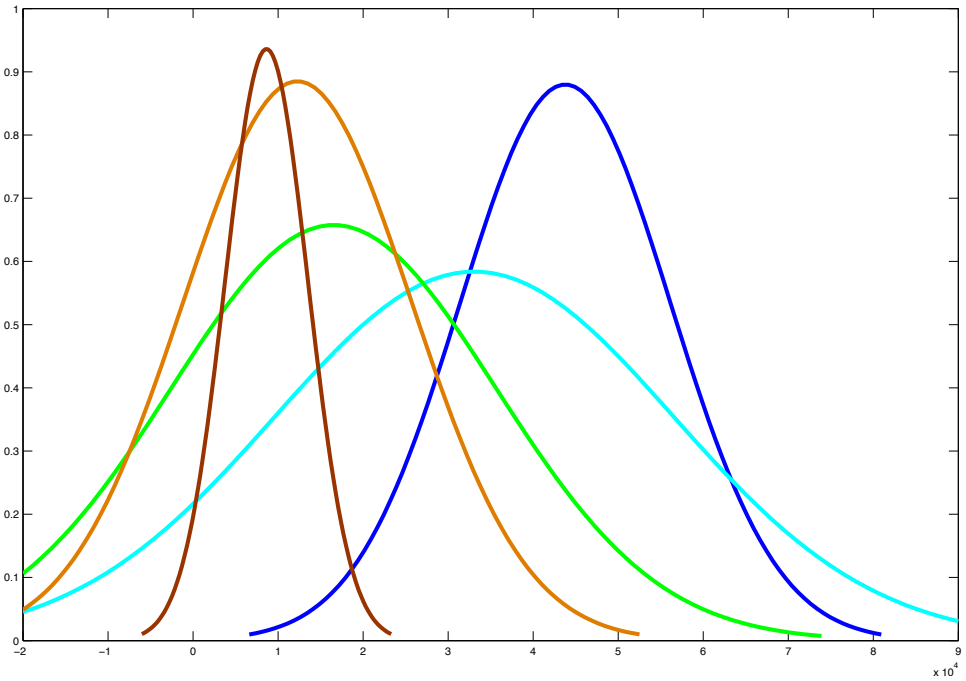


Figura 4.11: Distribuzione della feature verticale sulla seconda riga, del pixel G sull'occhio sinistro. Webcam integrata.

Webcam esterna

A puro scopo informativo sono stati realizzati i grafici utilizzando la webcam esterna. L'istogramma, mostrato in figura 4.12, rappresenta la *feature laterale* dell'occhio sinistro. In figura 4.13, vengono illustrate le distribuzioni estratte corrispondenti.

Come si può notare, anche in relazione all'istogramma della webcam integrata di figura 4.4, le campane sono molto spaziate l'una dall'altra e meglio riconoscibili.

Un miglioramento lo si riscontra anche nell'istogramma di figura 4.14, che mostra la *feature verticale* dell'occhio destro. In figura 4.15, vengono illustrate le distribuzioni estratte corrispondenti

Il parziale miglioramento lo si può notare nell'accentuata distanza tra i valori medi, rispetto alla figura 4.6 presa dalla webcam integrata. Lo stesso discorso vale anche per le altre features.

A fronte di questi grafici pare evidente che una futura implementazione, che preveda l'utilizzo di webcam esterna, non può che essere favorevole, considerando anche i ragionamenti fatti nella sezione dell'acquisizione delle immagini (3.2.1).

Tutte le misurazioni sono state fatte con il sottoscritto come soggetto. Non v'è alcuna prova in merito alla valutazione dell'efficacia delle features con altri utenti.

4.2 Analisi dell'aderenza agli obiettivi stabiliti

In questa sezione vengono richiamati gli obiettivi del progetto e vengono analizzati gli eventuali scostamenti e le relative cause.

Di seguito (tabella 4.16) vengono elencati gli obiettivi e ne viene specificata una sorta di *grado di priorità* con il quale s'è cercato di raggiungerli; la scala

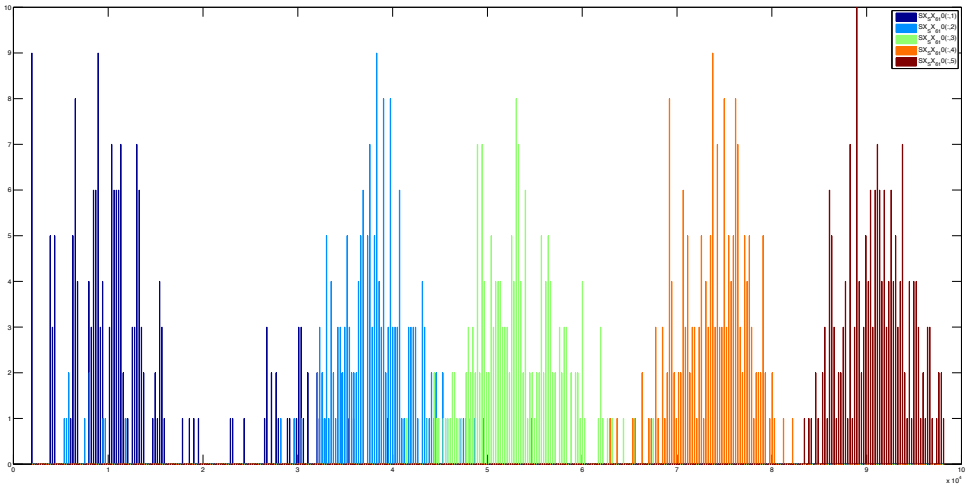


Figura 4.12: Valutazione della feature laterale sulla seconda riga dell'occhio sinistro. Webcam esterna.

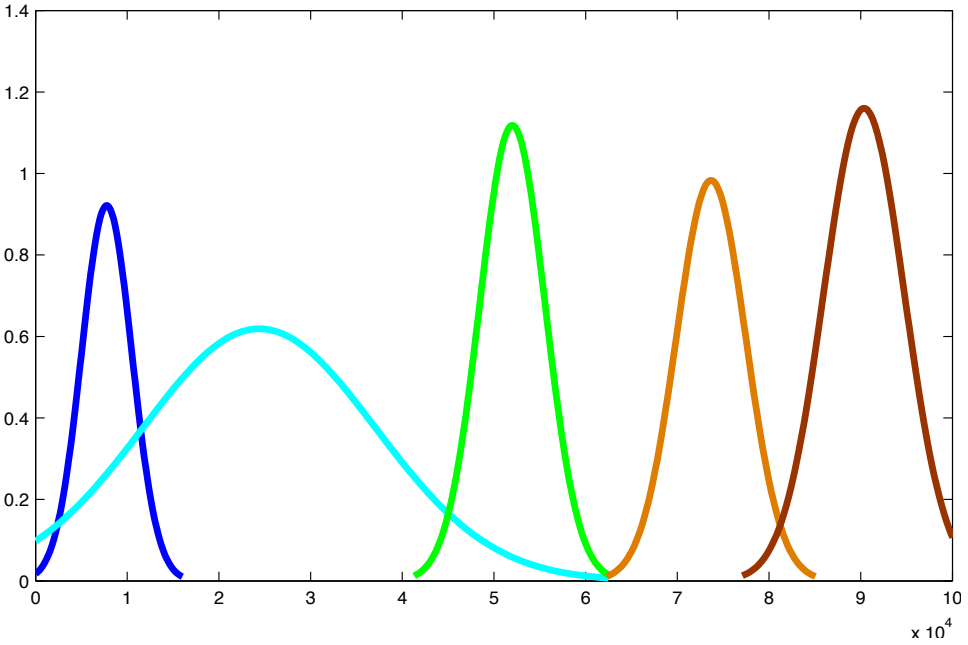


Figura 4.13: Distribuzione della feature laterale sulla seconda riga dell'occhio sinistro.Webcam esterna.

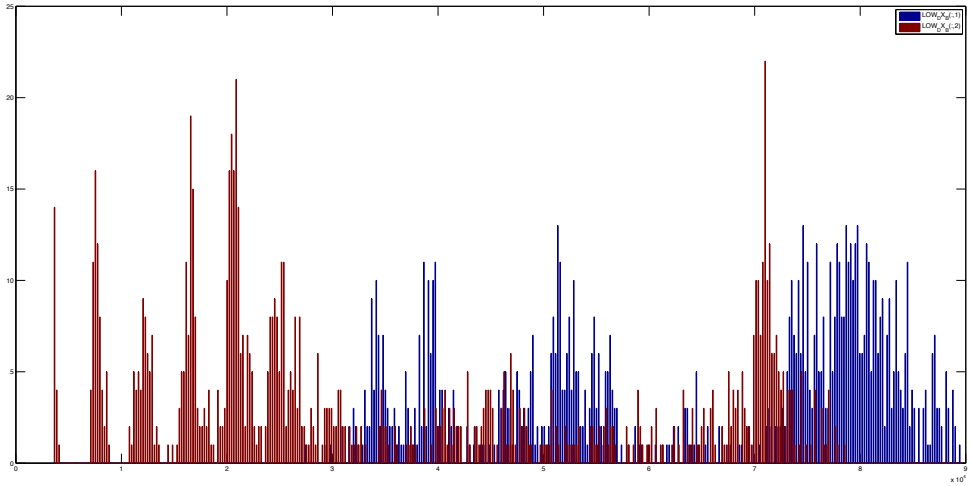


Figura 4.14: Valutazione della feature verticale, del pixel G sull'occhio destro. Webcam esterna.

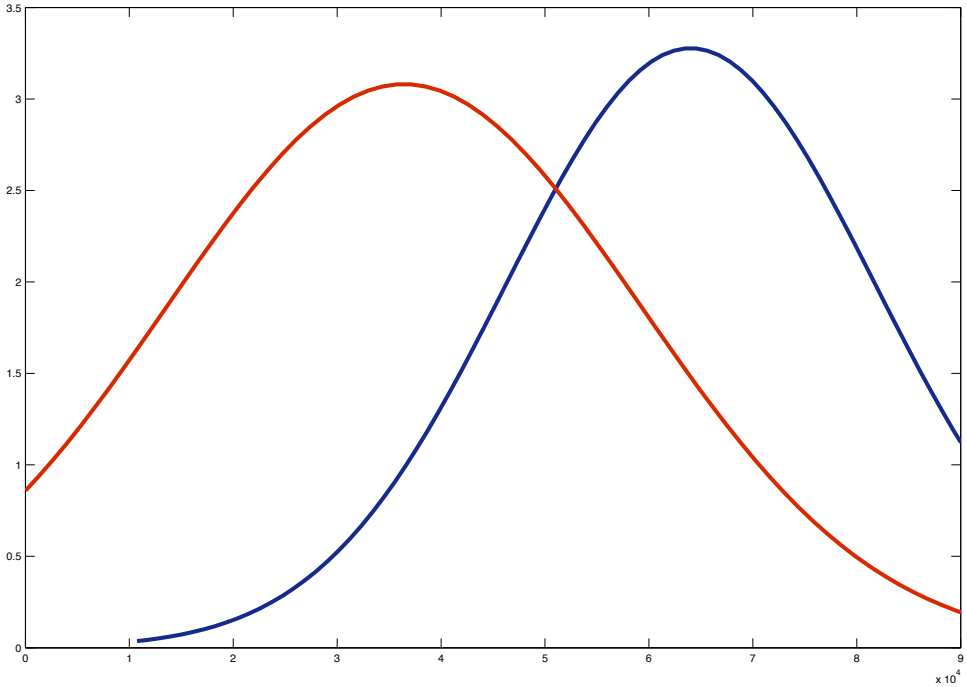


Figura 4.15: Distribuzione della feature verticale, del pixel G sull'occhio destro. Webcam esterna.

ha valori interi compresi da 1 a 3, dove 3 rappresenta il valore a maggior priorità. Viene inoltre indicato se tale obiettivo è stato raggiunto o meno.

Spendiamo ora due parole sulla *luminosità*. Come spiegato nella sezione 3.2.5, la parte che crea più problemi è il rilevamento dell'iride. In particolare l'elemento che degrada maggiormente le prestazioni sono le riflessioni sull'iride. Tale fenomeno si manifesta con evidenti zone bianche sull'iride che causano un innalzamento dei valore dei pixel in quell'area. Siccome il metodo identifica la zona a minor somma dei valori dei pixel come iride, le riflessioni causano un degrado nell'efficacia del metodo. In particolare una qualsiasi, anche minima, dissimmetria nell'illuminazione determina il degrado delle prestazioni.

Non è stato possibile, in questo sistema, fornire una soluzione efficace a tale problema. Si ribadisce come un qualsiasi sviluppo futuro, che vorrà migliorare le prestazioni di questo programma, non potrà escludere negli obiettivi una soluzione a tale problema.

Prestazioni del sistema

Come ultima analisi del lavoro svolto si valutano le prestazioni del programma come processo, all'interno del sistema operativo.

Una valutazione sulle prestazioni in termini di percentuale di processore utilizzato, di memoria allocata o di frame al secondo non è però possibile. La motivazione va ricercata ancora una volta nell'ambiente in cui si adopera il programma. In particolare il limite delle prestazioni è posto dalla velocità di acquisizione delle immagini da parte della webcam, dipendente quest'ultima dalla quantità di luce che invade l'obiettivo della webcam. In poche parole più luce è presente nell'ambiente, più ne viene catturata dalla webcam e più veloce va il programma (frame rate elevato).

Descrizione	*Priorità*	*Rispetto*
Localizzare la direzione dello sguardo.	3	**Sì**. Il programma offre un'ottima localizzazione nei dieci intervalli discreti.
Essere compatibile con tutti i soggetti.	3	**Parzialmente rispettata**. Non sono stati effettuati abbastanza test per fugare ogni dubbio. Da segnalare l'incompatibilità con le persone con gli occhiali o affette da strabismo.
Non avere alcun contatto con l'utente (bassa invasività).	3	**Sì**, totalmente rispettata. Invasività pressochè nulla.
Essere robusto a variazioni di luminosità.	3	**No**.
Non intralciare la vista dell'utente.	2	**Sì**, totalmente rispettata.
Essere in possesso di una buona precisione, ovvero con una bassa percentuale di errore, nel rilevare il punto osservato. Essere quindi in grado di rilevare le minime variazioni di posizione degli occhi.	2	**Parzialmente rispettata**. La scelta (forzata) di discriminare solo valori discreti limita gli effetti negativi di errori e disturbi, ma riduce il numero di punti osservabili.
Offrire una buona dinamica temporale e velocità di risposta. (meglio se in possesso di una risposta in tempo reale).	2	**Parzialmente rispettata**. La velocità dipende unicamente dalla quantità di luce nella stanza. Si veda la sezione sulle prestazioni (4.2).
Essere facilmente esteso per la registrazione binoculare.	2	**Sì**. Il programma necessità di entrambi gli occhi.
Essere compatibile con movimenti (relativamente) ampi della testa.	2	**No**. Per problemi di tempo non si è potuto implementare un tracciamento del volto.
Non richiedere hardware aggiuntivo.	1	**Sì**, totalmente rispettata.
Essere facilmente espandibile.	1	**Sì**. La modularità con cui è stato realizzato il programma garantisce la massima espandibilità possibile.

Figura 4.16: Elenco e valutazione degli obiettivi.

In condizioni di luminosità medie da ufficio, si hanno 10-15 frame al secondo. In una stanza con luce naturale proveniente da una finestra non è difficile ottenere 20-25 frame al secondo.

Il livello di prestazioni viene anche influenzato dal costo computazionale della fase di elaborazione e manipolazione delle immagini, anche se in minor parte rispetto a quanto detto.

Da notare come il cambio di illuminazione cambi le prestazioni delle varie parti di codice descritte in (3.2). In particolare nel rilevamento del volto si hanno prestazioni generalmente migliori, in termini sia di velocità che di efficacia, in condizioni di molta luce. Al contrario nel rilevamento della zona degli occhi si hanno prestazioni migliori con una quantità di luce minore.

Conclusioni In questo capitolo sono stati mostrati i risultati di una significativa fase di test. Sono state illustrate le sperimentazioni relative al cascade generato per la fase di riconoscimento del volto e quelle condotte la fine di valutare l'efficacia delle features utilizzate per la stima dello sguardo. Nell'ultima parte è stata condotta un'importante analisi sulla raggiungibilità degli obiettivi fissati all'inizio del progetto e del loro eventuale grado di scostamento.

Considerando le qualità delle prestazioni analizzate, nel prossimo ed ultimo capitolo, verrà stilato un utile elenco di possibili sviluppi da approntare al programma che lascerà spazio, in un successivo capitolo a se stante, alle conclusioni e ad un giudizio conclusivo sul lavoro svolto.

Capitolo 5

Sviluppi futuri

Introduzione In quest'ultimo capitolo vengono proposti dei possibili sviluppi futuri da apportare al programma. Vengono mostrati per ogni problematica diverse espansioni realizzabili che potrebbero contribuire ad un aumento delle prestazioni del programma.

Viene poi redatta, in un capitolo a se stante, una sintesi critica ed un giudizio conclusivo sul lavoro svolto.

5.1 Sviluppi futuri e proposte di miglioramento

Il programma realizzato in questo lavoro consiste in una serie di algoritmi a cascata. La scelta di implementare il programma con un elevata modularità garantisce la massima espandibilità possibile; ogni modulo è quindi migliorabile indipendentemente dagli altri. Nonostante alcune parti di programma siano migliori più efficaci di altre, un miglioramento in termini di prestazioni può essere apportato in qualsiasi modulo realizzato.

Verranno ora elencati i possibili sviluppi futuri, già discussi durante la descrizione dell'approccio utilizzato per questo progetto.

Eliminazione delle riflessioni

Il principale miglioramento realizzabile è l'eliminazione, o almeno l'attenuazione, delle riflessioni sull'iride. Non si eccede quando si afferma che la quasi totalità degli errori del programma proviene da questa fonte, o più in generale dalla bassa efficacia del metodo utilizzato per il rilevamento dell'iride. Si segnala una possibile soluzione proveniente dal lavoro della University of Sheffield, UK[24], dove le immagini degli occhi venivano pre-processate per eliminare le riflessioni tramite un semplice metodo di sottrazione. Si lascia ai posteri verificarne la fattibilità e l'efficacia.

Tracciamento del volto

Il secondo grande miglioramento può essere l'integrazione di un vero e proprio tracciamento del volto, col fine di diminuire il costo computazionale e di ottenere maggiore libertà di movimento all'utente. Una possibile realizzazione prevede l'utilizzo dell'algoritmo Lucas–Kanade.

Espansione delle features

Il terzo miglioramento consiste nell'espansione delle features atte alla stima dello sguardo. Una possibile realizzazione può essere estendere le features verticali anche ai pixel che si trovano al di sopra del rettangolo rilevato come iride, e non solo al di sotto.

Stima dello sguardo

Nella sezione dove viene descritta la stima dello sguardo è stato presentato il processo di manipolazione delle features raccolte. Una prima realizzazione prevedeva che le sei features formassero uno spazio a sei dimensioni, e che ogni casella rappresentasse un punto in questo spazio. Tale realizzazione era stata scartata per problematiche relative alla normalizzazione dei valori delle features. Un importante sviluppo può essere riprendere l'implementazione

originale risolvendo questo problema, prevedendo quindi un buon metodo di normalizzazione dei valori delle features.

Training

Il rilevamento degli occhi prevedeva l'utilizzo del cascade messo a disposizione da OpenCV. Tale cascade può essere allenato differenziando il rilevamento di occhi sinistri e destri, cosa non prevista in quello di OpenCV. Realizzare due cascade, uno per occhio, apporterebbe notevoli vantaggi in termini affidabilità del metodo.

Al contrario, a fronte dei risultati ottenuti non vi è la necessità di realizzare un cascade per il volto che sia migliore di quello offerto da OpenCV.

Webcam esterna

Per quanto riguarda l'utilizzo di una webcam esterna, è evidente che una futura implementazione che preveda l'utilizzo di questa tipologia di fotocamere non può che essere favorevole, e fortemente consigliata.

Dizionario

Si rileva un ampio margine di miglioramento all'interno del dizionario.

Vengono di seguito descritti due possibili miglioramenti. Il primo è la visualizzazione delle lettere accentate. Il secondo è l'inserimento di nuove parole, lettera per lettera.

Per quanto riguarda il dizionario non si può non accennare al fatto che sono presenti in rete molte soluzioni affidabili, ben sviluppate ma soprattutto già pronte all'uso.

Conclusioni

L'intelligenza è la capacità di adattarsi al cambiamento.

Stephen Hawking

In questo documento viene proposta una possibile realizzazione di un dispositivo di tracciamento dello sguardo.

Si è ritenuto opportuno dedicare all'interno della tesi un approfondimento riguardo all'utilizzo di questi dispositivi nell'ambito dell'assistenza ai disabili. Si stima che nel nostro paese vivano all'incirca tre milioni di persone affette da disabilità, in forme più o meno gravi. V'è dunque una forte esigenza di tecnologie che facilitino la vita e le interazioni delle persone.

In ambito commerciale esistono molti prodotti in vendita ma, come analizzato, a un costo difficilmente sostenibile dagli utilizzatori o, nel migliore dei casi, alquanto gravoso per la sanità nazionale. Scovare soluzioni a basso costo nella rete è impresa ardua e spesso ci si imbatte in software mal funzionanti o con grosse lacune. Non esiste in sostanza un vero riferimento con licenza open-source.

È con lo spirito di voler offrire un'opportunità, che si è deciso di spendere il lavoro di stage a servizio di questo progetto.

In conformità con l'esigenza di ottenere un dispositivo a basso costo si è optato per una realizzazione che non richiedesse alcun hardware aggiunto,

oltre ovviamente ad un moderno personal computer con webcam integrata. Lavorando al solo software è venuta di conseguenza la scelta realizzativa da adottare: il lavoro si inserisce nelle tecniche che utilizzano la luce bianca. In particolare si è cercato di realizzare un algoritmo in grado di rilevare e tenere traccia del limbo[1].

Il programma realizzato è composto da una serie di algoritmi a cascata. L'alta modularità con cui il programma è stato scritto garantisce la struttura necessaria per essere espanso il più possibile. Un ulteriore punto di forza del programma è l'utilizzo delle librerie OpenCV e del linguaggio di programmazione C. Questo linguaggio è uno dei più utilizzati e basilari che esistano, mentre la compilazione in linguaggio C++ estende le funzioni disponibili in fase di programmazione. Le librerie OpenCV sono inoltre una inesorabile fonte di metodi per elaborare e manipolare immagini. L'aggiunta infine di un dizionario predittivo non fa che accrescere il valore del progetto.

Il progetto presenta tuttavia alcune criticità. La più considerevole è rappresentata dalle riflessioni. Questi fenomeni causano un'errata valutazione delle posizione dell'iride e quindi della valutazione dello sguardo. Si è tuttavia piuttosto sicuri nell'affermare che una soluzione non sia un obiettivo irraggiungibile.

Dopo aver analizzato gli aspetti positivi, quelli negativi, le criticità e i possibili sviluppi sorge spontaneo porsi le seguenti domande sulle finalità del sistema di tracciamento dello sguardo realizzato:

- ha le potenzialità per diventare una buona piattaforma di sviluppo per applicazioni dedicate all'interazione uomo-macchina?

- L'algoritmo permette di ottenere risultati paragonabili ai sistemi commerciali?

[1] Il limbo è il confine tra la sclera (la parte bianca dell'occhio) e l'iride (membrana vascolare dell'occhio di colore variabile).

Nonostante gli obiettivi posti all'inizio del progetto siano stati raggiunti nella quasi totalità, l'illuminazione, con le annesse riflessioni, resta la problematica più grande. Questo problema incide in maniera rilevante sulla valutazione qualitativa del progetto. L'incapacità di trovare una soluzione adatta non permette di rispondere alla seconda domanda in maniera affermativa. La qualità di realizzazione e l'affidabilità richiesta a sistemi di tipo commerciale è ad un livello difficilmente comparabile con quello riscontrato nel sistema realizzato.

Si può tuttavia paragonare il sistema ad una versione *beta* di un dispositivo commerciale. Si parla dunque di un software non definitivo e nonostante non si possano offrire garanzie sull'affidabilità e sull'assenza di errori del software è possibile con poco sforzo migliorare sensibilmente le prestazioni, fino a raggiungere un livello più che accettabile.

L'utilizzo ripetuto del programma lascia inoltre trapelare un cauto ottimismo sulla potenzialità del progetto. Il programma realizzato permette, nonostante tutto, di localizzare lo sguardo con un accettabile grado di precisione. Quando la fase di calibrazione si compie correttamente, limitando il più possibile i movimenti del capo e tenendo gli occhi il più possibile aperti, i risultati in fase di utilizzo del programma sono più che apprezzabili, riuscendo a comporre facilmente diverse parole. Inoltre le funzioni di supporto inserite nel dizionario rendono l'utilizzo del sistema sufficientemente piacevole.

Si spera dunque che tale lavoro svolto possa gettare le basi per ulteriori studi e approfondimenti.

Appendice A

Adaboost e generazione del file *XML*: `haartraining`

In questo capitolo verranno spiegati nella prima sezione i principi teorici alla base dell'algoritmo *AdaBoost* e dell'utilizzo delle *Haar-like Features*, mentre nella seconda sezione verranno analizzati i passaggi fondamentali per la realizzazione di un file *XML* attraverso una fase di *Training*, allenamento, chiamata `haartraining`. Tale file viene utilizzato come *cascade* per il rilevamento del volto.

A.1 Adaboost

AdaBoost (Adaptive Boosting) è un algoritmo di apprendimento macchina. L'algoritmo permette di creare un classificatore forte partendo da classificatori deboli. L'algoritmo Adaboost viene implementato, in OpenCV, tramite il metodo `cvHaarDetectObjects` che permette di rilevare determinati oggetti da un'immagine. Il metodo è il seguente [68] :

```
1 CvSeq * cvHaarDetectObjects (const CvArr* image, CvHaarClassifierCascade* cascade,
      CvMemStorage* storage, double scalefactor=1.1, int minneighbors=3, int flags=0,
      CvSize minsize=cvSize(0, 0));
```

Prima di poter utilizzare il metodo bisogna svolgere una fase di *Training*, allenamento, per istruire l'algoritmo. La fase di Training è quella procedura che permette al metodo di ottenere le capacità di discriminare se una determinata zona di una immagine ha le stesse caratteristiche dell'oggetto da riconoscere, come un volto o un occhio, oppure no. Per svolgere l'operazione di Training, OpenCV mette a disposizione una serie di eseguibili, con codici sorgente annessi; il risultato finale di questa operazione sarà la generazione di un file con estensione *XML*. Questo file sarà passato come argomento al metodo `cvHaarDetectObjects` nel codice del proprio programma[1].

A.1.1 Haar-like features

Il metodo per il rilevamento del volto utilizzato dalle librerie OpenCV, e anche in questa tesi, si basa sul famoso metodo proposto da Viola-Jones nel 2001 [69] e migliorato da Rainer Lienhart [70] l'anno succcessivo introducendo una serie di nuove features.

Le *Haar-like* sono features usate nella *object recognition*, vale a dire nel riconoscimento di oggetti. Il nome è ispirato alle *Harr Wavelets*, sequenze di onde quadre, che possono assumere, se scalate, solo i valori uno e zero.

L'algoritmo di Viola e Jones proposto nel 2001 introduce tre elementi di novità rispetto agli algoritmi di ricerca di volti precedentemente sviluppati[2]:

- Il primo consiste nell'utilizzo delle *Haar-like features* sulle regioni che compongono un'immagine in combinazione con una nuova rappresentazione in forma matriciale dell'immagine stessa, detta immagine integrale. Le features utilizzate, grazie alla loro semplicità, presentano un basso costo computazionale e l'impiego della nuova struttura dati permette

[1] Per la descrizione delle operazioni necessarie alla creazione del file con estensione XML si rimanda all'appendice (A.2).

[2] Per affrontare l'argomentazione in questa sezione farò prevalentemente riferimento ai contenuti del sito: [71].

di effettuare l'analisi in un tempo costante, indipendentemente dalla dimensione delle regioni analizzate.

- In secondo luogo viene presentato un metodo per la costruzione di classificatori basato sulla selezione di un basso numero di features di Haar attraverso l'algoritmo *AdaBoost* di Freud e Shapire del 1995 [72]. Questa strategia permette di eliminare la maggior parte delle features di scarsa capacità discriminante e selezionare solo quelle più efficaci per la costruzione del classificatore.

- Il terzo maggiore contributo consiste nell'introdurre una nuova strategia di analisi dell'immagine basata su una *struttura a cascata* dove ogni livello della cascata è un classificatore creato con procedura AdaBoost. La complessità dei livelli cresce man mano che si procede verso la fine della cascata; in questo modo le regioni di facile analisi vengono scartate velocemente ai primi stadi, mentre quelle di più difficile interpretazione sono sottoposte a più livelli di verifica. Qualora una sotto regione completi la cascata, ovvero superi tutti gli stadi di classificazione con successo, viene etichettata come regione contenente una faccia.

Come anticipato, il metodo ideato da Viola e Jones classifica le regioni di una immagine basandosi su valori calcolati attraverso semplici operatori chiamati Haar-like features. In particolare ve ne sono di tre tipologie, visualizzate in figura A.2a:

- features composte da una coppia di rettangoli affiancati verticalmente o orizzontalmente (A e B);

- features formate da tre rettangoli affiancati orizzontalmente (C) ;

- features composte da quattro rettangoli disposti a scacchiera (D).

Le features, oltre che per tipo, si differenziano tra loro anche per dimensione e posizione all'interno della detection window (la finestrella di ricerca che

scansiona l'immagine in analisi). Il valore di una feature, una volta applicata su una regione dell'immagine, (figura A.1), è dato dalla differenza (normalizzata) tra i livelli di grigio dei pixel appartenenti ai rettangoli bianchi rispetto a quelli neri.

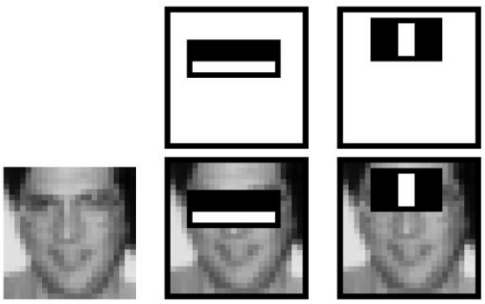


Figura A.1: Haar-like features applicate ad una immagine.

Le motivazioni a favore dell'utilizzo di questo tipo di features sono molteplici. In primo luogo, considerando che i volti hanno una struttura regolare (ad esempio si noti che l'asse degli occhi e la posizione centrale del naso sono caratteristiche comune a tutte le facce), è intuibile come tali caratteristiche possano contribuire a discriminare i positivi, l'oggetto che voglio identificare, dai negativi, tutto il resto, in maniera netta.

In seconda battuta, grazie all'utilizzo dell' immagine integrale, l'operazione di accesso ai pixel dell'immagine risulta computazionalmente molto efficiente (il costo è costante indipendentemente dall'estensione in pixel del rettangolo). L'immagine integrale, che ha le stesse dimensioni dell'immagine originale, nella posizione di indice (x,y) contiene la somma dei livelli di grigio di tutti i pixel superiori e a sinistra del punto (x,y) incluso. Utilizzando l'immagine integrale, la somma dei livelli di grigio di una qualsiasi area rettangolare in un'immagine può essere calcolata attraverso pochi riferimenti alla matrice.

La parola "cascade" significa che il classificatore risultante si presenta robusto e veloce ed è definito come forte. Esso è costituito da diversi classificatori più semplici, definiti deboli, che vengono applicati in sequenza

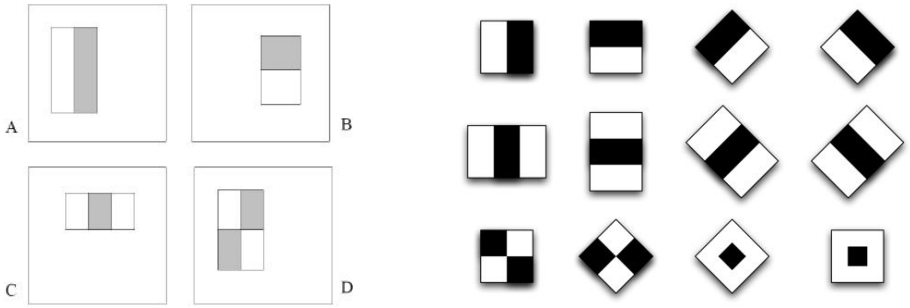
ad una regione di interesse. L'operazione finisce quando il candidato viene respinto o quando tutte le fasi sono passate e l'immagine è classificata come positiva.

Ogni classificatore debole è predisposto per correggere gli errori del classificatore precedente e l'analisi è effettuata calcolando la cosiddetta immagine integrale, che è calcolata come:

$$P(x,y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

dove $P(x,y)$ è l'immagine integrale e $I(x,y)$ è l'immagine originale.

Attualmente vengono supportati Adaboost discreti, Real Adaboost, Gentle Adaboost e Logitboost. Come abbiamo detto, il metodo `cvHaarDetectObjects` utilizza le seguenti Haar-like features (figura A.2b).



(a) Set base di Haar-like features. **(b)** Set esteso di Haar-like features.

Figura A.2: Haar-like features.

La ragione principale dell'utilizzo di Adaboost con l'utilizzo di Haar-like features al posto di altri concorrenti, è principalmente per la sua idoneità nella classificazione in tempo reale.

A.2 Generazione del file *XML*: `haartraining`

Di seguito verrà spiegato come creare il file *XML* attraverso una fase di *Training*, allenamento, chiamata `haartraining`. Questa sezione prende spunto da un'ottima guida [73].

A.2.1 Preparazione dei dati

La prima fase consiste nella preparazione dei dati. Questa fase richiede non poco lavoro, soprattutto per la ricerca e l'adattamento delle immagini allo scopo. Le fasi da seguire sono:

- scelta del database. Il database scelto per questo progetto è il FERET Database, lo stesso (probabilmente) usato da OpenCV[3].
- Fare la collezione delle immagini positive, ovvero del solo oggetto di cui ci interessa fare la rilevazione (faccia, occhio, ecc.). Le immagini devono essere ritagliate in modo da contenere il solo l'oggetto desiderato. Le immagini dovranno inoltre essere tutte delle stesse dimensioni.
- Fare la collezione delle immagini negative, ovvero immagini che non contengano l'oggetto che ci interessi rilevare (paesaggi, altri oggetti, ecc.).
- Creare i campioni (*samples*) da fornire al `haartraining`. I campioni sono dei vettori (estensione .vec) che contengono tutte le immagini. Bisognerà farne uno per i positivi e uno per i negativi. Esiste l'utility `createsamples` che serve a tale scopo. Viene descritta qui nel seguito.

A.2.2 Fase di creazione dei *samples*

L'utility `createsamples` è la funzione che serve alla preparazione dei dati del database prima della fase di addestramento. È così strutturata:

[3]Per un approfondimento su FERET Database si rimanda all'appendice (E).

```
Usage: ./createsamples
       [-info <descriptionfilename>]
       [-img <imagefilename>]
       [-vec <vecfilename>]
       [-bg <backgroundfilename>]
       [-num <numberofsamples = 1000>]
       [-bgcolor <backgroundcolor = 0>]
       [-inv] [-randinv]
       [-bgthresh <backgroundcolorthreshold = 80>]
       [-maxidev <maxintensitydeviation = 40>]
       [-maxxangle <maxxrotationangle = 1.100000>]
       [-maxyangle <maxyrotationangle = 1.100000>]
       [-maxzangle <maxzrotationangle = 0.500000>]
       [-show [<scale = 4.000000>]]
       [-w <samplewidth = 24>]
       [-h <sampleheight = 24>]
```

Questa è la lista di tutte le opzioni, ma ci sono principalmente quattro funzioni per produrre i campioni:

1. *Create training samples from one*: si usa per creare campioni per il training partendo da una sola immagine applicando poi alcune distorsioni. Si usa questa opzione nel caso non si disponga di un numero adeguato di immagini, ne sia possibile recuperarle.

2. *Create training samples from some*: si usa per creare campioni per il training partendo da molte immagini. Viene descritta nel seguito.

3. *Create test samples*: si usa per creare campioni con il relativo sfondo da un'immagine singola applicando distorsioni. Si utilizzano nella fase di test.

4. *Show images*: si utilizza per mostrare immagini all'interno di un file .vec.

A.2.2.1 Create training samples from some

La seconda funzione è quella da utilizzare per creare i campioni da molte immagini senza applicare distorsioni. Questa funzione (si trova nel sorgente `cvsamples.cpp`, metodo: `cvCreateTrainingSamplesFromInfo`) viene lanciata quando le opzioni, `-info` e `-vec` sono state specificate, ed indicano:

- -info <description_file_of_samples>
- -vec <name_of_the_output_file_containing_the_generated_samples>

Un esempio di codice completo per creare i campioni è il seguente:

```
createsamples -info samples.dat -vec samples.vec -w 20 -h 20
```

Si può pensare questa funzione come una funzione di conversione del formato del file. Il formato del <description_file_of_samples> è il seguente:

```
[filename] [# of objects] [[x y width height] [... 2nd object] ...]
[filename] [# of objects] [[x y width height] [... 2nd object] ...]
[filename] [# of objects] [[x y width height] [... 2nd object] ...]
...
```

dove (x, y) è l'angolo superiore sinistro dell'oggetto da rilevare, nel nostro caso (0,0) visto che le immagini erano ritagliate in modo da contenere il solo oggetto.

Questo formato è chiamato *description file format*. Questa funzione specifica e ridimensiona le immagini e le converte in formato vettore (.vec). È possibile utilizzare questa seconda funzione solo quando si ha un numero sufficiente di immagini positive e negative. L'ordine di grandezza consigliato è intorno a 5000/7000 immagini.

A.2.2.2 Come creare un *description file*

Si elencano i passi per creare un file di descrizione quando sono già disponibili i file delle immagini ritagliate. È possibile utilizzare il comando

`find` e `identify` per creare il file di descrizione [74]. Nel caso tutte le immagini abbiano la stessa dimensione (come nel nostro), diventa più semplice e veloce nel seguente modo:

```
find <dir> -name '*.<ext>' -exec echo {} 1 0 0 <width> <height> \; > <
    descriptionfile>
```

A.2.3 Fase di Training

Viene ora la vera e propria fase di *Training* mediante l'utilizzando dell'utility `opencv_haartraining.exe`. Si trova all'indirizzo `C:\OpenCV2.0\bin\`:

```
Usage: ./haartraining
       -data <dirname>
       -vec <vecfilename>
       -bg <backgroundfilename>
       [-npos <numberofpositivesamples = 2000>]
       [-nneg <numberofnegativesamples = 2000>]
       [-nstages <numberofstages = 14>]
       [-nsplits <numberofsplits = 1>]
       [-mem <memoryinMB = 200>]
       [-sym (default)] [-nonsym]
       [-minhitrate <minhitrate = 0.995000>]
       [-maxfalsealarm <maxfalsealarmrate = 0.500000>]
       [-weighttrimming <weighttrimming = 0.950000>]
       [-eqw] [-mode <BASIC (default) | CORE | ALL>]
       [-w <samplewidth = 24>]
       [-h <sampleheight = 24>]
       [-bt <DAB | RAB | LB | GAB (default)>]
       [-err <misclass (default) | gini | entropy>]
       [-maxtreesplits <maxnumberofsplitsintreecascade = 0>]
       [-minpos <minnumberofpositivesamplespercluster = 500>]
```

Si noti dal codice sorgente come le immagini vengano convertite, se a colori, in scala di grigi. Nello specifico vengono settati i seguenti parametri:

- $nstages = 20$;
- $nsplits = 2$;

- *minhitrate* = 0.9999 (default: 0.995);
- *maxfalselarm* = 0.5 (default: 0.5);
- *weighttrimming* = 0.95 (default: 0.95);
- *20x20* of sample size;
- *nonsym*, è usato quando le classe oggetto non ha simmetria verticale. (Se la classe oggetto ha simmetria verticale, come facce frontali, deve essere utilizzato *-sim* (default). In questo caso sarà più veloce.
- *mem* = 512, è la memoria disponibile, in MB, per il solo precalcolo.
- *mode* = ALL, utilizza il set Extended delle Haar-like Features. Il BASIC è di default.

Il codice che si ottiene è quindi:

```
1 haartraining -data haarcascade -vec samples.vec -bg negatives.dat -nstages 20 -
      nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 7000 -nneg 3019 -w 20 -h
      20 -nonsym -mem 512 -mode ALL
```

A.2.3.1 Generare il file XML

Il processo di `haartraining` genera il file *XML* quando il processo è completamente finito. Se si vuole convertire un intermedio haartraining in un file *XML*, basta compilare il software che c'è nella directory `C:\OpenCV2.0-\samples\c\convert_cascade.c` (ovvero nella cartella di installazione). L'ingresso da inserire è:

```
1 convertcascade --size="<samplewidth>x<sampeheight>" <haartrainingouputdir> <
      ouputfile>
```

Il file *XML* generato sarà dato in pasto al face detection, ovvero al metodo `cvHaarDetectObjects`.

OpenCV mette a disposizione parecchie utility per il testing e per valutazione delle performance. Nel mio caso queste valutazioni sono state fatte

personalmente. Si rimanda alla sezione sperimentazione e risultati (4.1.1), per la descrizione dei risultati ottenuti.

Appendice B

Linguaggio e librerie utilizzate

Per questo progetto ci si è avvalsi dell'utilizzo di OpenCV.

OpenCV è una libreria orientata alla Computer Vision, scienza che si occupa di acquisire, elaborare e riprodurre immagini allo scopo di ricreare modelli reali [75].

Da "libreria proprietaria", sviluppata da *Intel*, nei primi anni di questo secolo è diventata "licenza open source *BSD*" (Berkeley Software Distribution). Questo permette una libera ridistribuzione del codice sia in forma sorgente che binaria, anche all'interno di prodotti commerciali, a condizione di mantenere le note di copyright e di non utilizzare il nome Intel. La prima versione fu rilasciata nel 2000, seguita da altre cinque release negli anni successivi.

L'utilizzo primario delle librerie è quello collegato alla visione artificiale, il cui problema principale, come già visto, è quello di estrarre dalle immagini dati significativi e trattarli in modo automatico. A questo scopo si aggiungono anche "toolkit", ovvero librerie di primitive per la creazione di oggetti grafici, librerie di rendering e multimedia, come DirectX e OpenGL e librerie di gestione hardware grafico. È una libreria multipiattaforma ed è quindi

compilabile con molti sistemi operativi (Windows, Mac OS X, Linux, PSP, VCRT) e sono integrabili con programmi come Visual Studio, Dev-C++ e Xcode.

Il linguaggio di programmazione utilizzato per sviluppare questa libreria fu originariamente il C, ma venne esteso anche per il linguaggio C++ a partire dalla versione 2.0. Le motivazioni di questa scelta furono la riduzione del numero di righe di codice, la riduzione degli errori di programmazione, come ad esempio la perdita di memoria, fatto molto comune nell'elaborazione delle immagini. Attualmente la maggior parte dei nuovi sviluppi ed algoritmi sono scritti in linguaggio C++, ma può essere utilizzata con altri linguaggi come il C e il Python.

La libreria è divisa in alcuni binari distinti (.dll), divisi per settori, in modo che non si debba fare il linking completo. I vari settori sono:

- CxCore: è l'oggetto principale nonché l'unico strettamente indispensabile. Include infatti tutte le funzioni di inizializzazione delle strutture utilizzate, l'algebra lineare e le altre funzioni di base.

- Cv e CvAux: includono praticamente tutte le funzioni di trattamento ed analisi, quindi il cuore delle *OpenCV*.

- HighGui: include alcune comode funzioni GUI, come ad esempio la tipica finestra di display, oltre alle funzioni di salvataggio e caricamento immagini da file.

- CvCam: include funzioni di acquisizione video e di gestione delle telecamere.

Le principali aree d'applicazione di *OpenCV* sono:

- Facial recognition

- Gesture e Motion recognition

- ❑ Human–computer interaction (HCI)
- ❑ Mobile robotics
- ❑ Object Identification
- ❑ Motion tracking

All'interno del sito [75] è possibile trovare tutto il materiale, le documentazioni, gli esempi e il supporto necessario a qualsiasi progetto[1].

Ritornando allo specifico di questa tesi, il codice scelto è il linguaggio di programmazione C. La scelta è ricaduta su questo linguaggio in quanto la versione delle librerie OpenCV utilizzata è la 2.0. Inoltre, come è possibile notare anche dall'estensione dei file sorgente, il progetto prevede vari classi proprie del linguaggio di programmazione C++, come ad esempio la classe `string` o la classe `list`.

[1] Si segnala *Seeing With OpenCV* [76], ottimo tutorial per chi volesse iniziare ad utilizzare *OpenCV*. Un utile aiuto può venire dal libro *Learning OpenCV: Computer Vision with the OpenCV Library* [77], che presenta molti esempi e spiegazioni anche se la risorsa maggiore resta il web; a tal scopo si segnala l'ottimo e molto usato forum per le discussioni su Open Source Computer Vision Library [78].

Appendice C

Spazio colore

Uno *spazio colore* è la combinazione di un modello matematico e di una funzione di mappatura di quest'ultimo. Il modello matematico non è nient'altro che una rappresentazione numerica dei colori, tipicamente attraverso tre o quattro valori. La combinazione di questi valori da luogo ad uno spazio all'interno del quale i colori assumono determinati valori, approssimabili a dei punti.

Lo spazio colore più conosciuto è lo spazio *RGB*. Sviluppato nel 1931 dalla CIE (Commission internationale de l'éclairage), si basa sui tre colori rosso (Red), verde (Green) e blu (Blue), da cui appunto il nome RGB. Un'immagine può infatti essere scomposta, attraverso filtri o altre tecniche, in questi colori base che miscelati tra loro danno quasi tutto lo spettro dei colori visibili. La scelta di questi colori è legata alla fisiologia dell'occhio umano.

C.1 Spazio YCbCr

Uno spazio altrettanto noto nella trasmissione dei segnali è lo spazio *YCbCr*, o *Y'CbCr*. Si tratta nella realtà di una famiglia di spazi colore e vengono utilizzati principalmente per la trasmissione di segnali televisivi. YCbCr non è uno spazio colore assoluto ma un modo di codificare l'informazione RGB.

Le immagini televisive vengono trasmesse con questa rappresentazione. I televisori a colori utilizzano tutte e tre le componenti, convertite poi nello spazio RGB. I televisore in bianco e nero, o meglio in scala di grigi, utilizzano solo la componente di luminanza, ovvero il canale Y. L'utilizzo di questo sistema è più efficiente in termini di registrazione e trasmissione, rispetto ad un segnale nello spazio RGB.

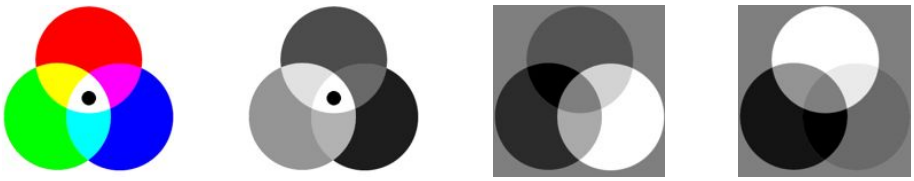
I segnali Y'CbCr (il simbolo primo ' significa che l'intensità della luce è non-lineare e viene codificato utilizzando una correzione di gamma) prima di essere processati per ottenere un segnale in forma digitale, sono chiamati *YPbPr*, e vengono creati dai corrispondenti primari RGB corretti in gamma usando due costanti: Kb e Kr. Il valore luma (Y) risultante avrà un valore nominale da 0 a 1, e i valori di croma o crominanza (Cb e Cr) da -0.5 a +0.5. Di solito viene aggiunto un valore di offset o vengono codificati in scale di range [0..255].

Le componenti principali dello spazio sono tre:

- il valore di *luminanza* (**Y**) rappresenta l'intensita di luce complessiva dell'immagine (cioe la somma dei tre colori primari). Si ottiene attraverso la formula: $Y = Blu + Rosso + Verde$.
- Il valore di *croma rosso* (**Cr**), differenza dal colore rosso, si ottiene tramite la formula: $C_r = Rosso - Y$.
- Il valore di *croma blu* (**Cb**), differenza dal colore blu, si ottiene tramite la formula: $C_b = Blu - Y$.

In figura (C.1) sono illustrate tutte le componenti dello spazio YCbCr.

Le varie immagini sono fatte al seguente modo: all'immagine originale in RGB sono state estratte le sole componenti specificate. L'immagine è quindi composta da un solo canale, con valori compresi tra [0..255]. La visualizzazione in scala di grigi è dovuta all'interpretazione del calcolatore, il quale riconoscendo un solo canale lo interpreta in questo modo.



(a) RGB Image. **(b)** Comp Y. **(c)** Comp Cb. **(d)** Comp Cr.

Figura C.1: Immagini in spazio colore YCbCr.

Le varie componenti sono:

- la componente **Y**, figura (C.1b), mostra la rappresentazione in scala di grigi dell'immagine originale.
- La componente **Cb**, figura (C.1c), mostra valori alti (zona bianca, valore 255) nelle zone blu e valori bassi (zona nera, valore 0) nelle zone opposte ad esso, quindi il giallo, essendo quest'ultimo formato nello spazio RGB dalle sole componenti R e G (rosso e verde).
- La componente **Cr**, figura (C.1d), mostra valori alti (zona bianca, valore 255) nelle zone rosse e valori bassi (zona nera, valore 0) nelle zone opposte ad esso, quindi l'azzurro, essendo quest'ultimo formato nello spazio RGB dalle sole componenti B e G (blu e verde).

La conversione da uno spazio all'altro si ottiene tramite le formule illustrate in figura (C.2).

$$\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256} \begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix} \bullet \begin{bmatrix} R'_{255} \\ G'_{255} \\ B'_{255} \end{bmatrix}$$

(a) Conversione da RGB a YCbCr.

$$\begin{bmatrix} R'_{255} \\ G'_{255} \\ B'_{255} \end{bmatrix} = \frac{1}{256} \begin{bmatrix} 298.082 & 0. & 408.583 \\ 298.082 & -100.291 & -208.120 \\ 298.082 & 516.411 & 0. \end{bmatrix} \bullet \left(\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

(b) Conversione da YCbCr a RGB.

Figura C.2: Conversioni RGB/YCbCr.

La conversione avviene tra canali a 8 bit, con valori di range [0..255]. La conversione YPbPr o Y'CbCr e RGB viene definita nella specifica ITU-R BT.601 (L'ITU Radiocommunication Sector è uno dei tre settori (divisioni o unità) dell'International Telecommunication Union (ITU) ed è responsabile per le comunicazioni radio).

Appendice D

Anatomia dell'occhio umano

Uno studio completo e approfondito dell'anatomia dell'occhio non è qui previsto, e ne si lascia il compito ad altri lavori. Le uniche informazioni utili alla comprensione del progetto sono quelle di seguito descritte.

D.1 Movimenti dell'occhio

Lo sguardo generalmente non compie spostamenti continui, ma si muove a intervalli discreti con una serie di salti improvvisi. Le tipologie di movimento che l'occhio umano compie nel suo normale utilizzo, come nella lettura di un testo, sono due: le saccadi e le fissazioni.

Le *saccadi* consistono in rapidi movimenti degli occhi eseguiti per portare la direzione dello sguardo a coincidere con la zona di interesse. Sono movimenti ad alta velocità che compaiono durante i cambiamenti di fissazione visiva da un oggetto all'altro. Vengono eseguite in media 3-4 saccadi al secondo.

Spesso a una saccade è seguita una *fissazione*. Una fissazione è un periodo di relativa stabilità nel movimento durante il quale un oggetto viene "guardato". In tale periodo lo sguardo rimane fisso sull'oggetto interessato.

Durante la fissazione l'occhio non rimane mai completamente fermo, ma compie dei piccoli movimenti intorno alla zona d'interesse.

Pare evidente come il programma realizzato, rilevando lo sguardo entro valori discreti, permetta solo di riconoscere movimenti saccadi. I piccoli movimenti tipici delle fissazioni non vengono rilevati.

D.2 Effetto occhi rossi

In fotografia l'effetto occhi rossi è il fenomeno per cui, nelle foto scattate con l'uso del flash, spesso accade che gli occhi dei soggetti acquisiscano un colore rosso piuttosto marcato. Il fenomeno è dovuto all'incapacità dell'iride di chiudere la pupilla a fronte del lampo troppo veloce del flash. Il lampo va dunque a colpire la retina, ricca di sangue, che causa l'effetto di colore rosso. Lo stesso principio è usato intenzionalmente dall'oftalmoscopio, strumento utilizzato per esaminare la retina.

Nel caso in cui la luce che incide sull'occhio sia a lunghezza d'onda infrarosso, con una fotocamera in grado di rilevare tale lunghezza d'onda si vedrebbe la pupilla illuminarsi e diventare molto chiara. Il contrasto che si viene così a creare tra pupilla ed iride viene di frequente utilizzato per rilevare le relative posizioni.

D.3 Immagini di Purkinje-Sanson

Le immagini di *Purkinje* o di *Purkinje-Sanson* sono riflessioni delle superfici dei mezzi oculari [79].

La riflessione della luce è quel fenomeno fisico per cui quando la luce incontra una discontinuità nel mezzo viene generalmente divisa in due fasci, uno trasmesso e uno riflesso.

Le immagini di Purkinje visibili su un'occhio sono quattro, e ne si può vedere una schematizzazione in figura (D.1).

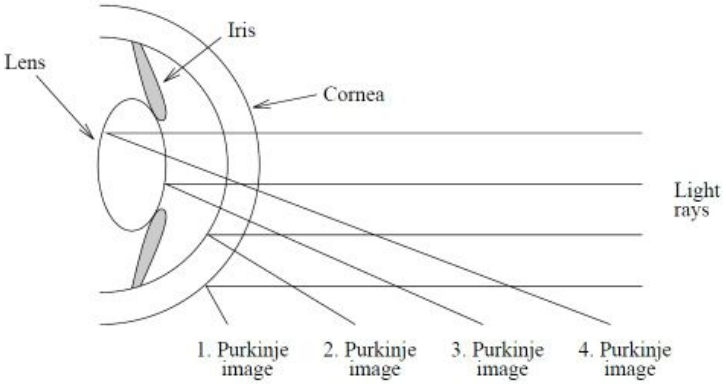


Figura D.1: Le quattro immagini di Purkinje sono riflessi di luce in entrata ai confini della lente e della cornea.

La prima immagine di Purkinje (1) è il riflesso della superficie esterna della cornea. La seconda immagine di Purkinje (2) è il riflesso della superficie interna della cornea. La terza immagine di Purkinje (3) è il riflesso della superficie esterna (anteriore) della lente. La quarta immagine di Purkinje (4) è il riflesso della superficie interna (posteriore) della lente. A differenza degli altri, la quarta immagine è un'immagine invertita.

Questi fenomeni di riflessione prendono il nome dall'anatomista ceco Jan Evangelista Purkyně (1787-1869) e dal contributo del medico francese Louis Joseph Sanson (1790-1841). Le prima e la quarta immagine di Purkinje vengono spesso utilizzate da dispositi di tracciamento dello sguardo.

Appendice E

FERET Database

Il programma denominato *FERET* è durato dal 1993 al 1997. Sponsorizzato dal Dipartimento della Difesa americana attraverso il Defense Advanced Research Products Agency (DARPA), la sua missione primaria era sviluppare un sistema di riconoscimento automatico dei volti [80]. Tale capacità era utilizzata per assistere la sicurezza nazionale, l'intelligence e le forze dell'ordine nell'esercizio delle loro funzioni.

Il database FERET è stato raccolto in 15 sedute tra agosto 1993 e luglio 1996. Il database contiene 1.564 set di immagini per un totale di 14.126 immagini: esse comprendono 1.199 individui diversi e 365 set di immagini duplicate. Un duplicato è una successiva serie di immagini di una persona già nel database, catturate in sessioni temporali diverse. Per alcuni individui sono trascorsi oltre due anni fra la prima seduta e l'ultima. Questo intervallo di tempo è stato fondamentale poichè ha permesso ai ricercatori di studiare i piccoli cambiamenti nella fisionomia di un soggetto nell'arco degli anni.

Per ottenere l'accesso al database bisogna richiedere al sito [81] l'abilitazione. Il database è diviso in due DVD che contengono:

- *binaries*: alcuni eseguibili, con sorgenti, che servono come utilities per scompattare o elaborare le immagini.

- *Immagini* in tre formati diversi. La maggior parte a colori, il che rende il FERET Database molto utilizzato a livello internazionale.

- File con estensione *.txt* e *.html*, una coppia per immagine, che contengono una descrizione dell'immagine corrispondente. Ci sono informazioni riguardo: sesso della persona, razza, posizione del volto, coordinate di punti strategici (come occhi, bocca e naso) e altra informazioni riguardanti la fotografia (luminosità, ambiente, ecc.).

Bibliografia

[1] Wikipedia, The Free Encyclopedia, "Amyotrophic lateral sclerosis," 2011. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Amyotrophic_lateral_sclerosis&oldid=426379424

[2] A. I. S. L. Amiotrofica. [Online]. Available: www.aisla.it

[3] Wikipedia, The Free Encyclopedia, "Cerebral palsy," 2011. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Cerebral_palsy&oldid=426272992

[4] [Online]. Available: www.paralisi-cerebrale.org

[5] A. Calvo, A. Chiò, E. Castellina, F. Corno, L. Farinetti, P. Ghiglione, V. Pasian, and A. Vignola, "Eye tracking impact on quality-of-life of als patients," in *Proceedings of the 11th international conference on Computers Helping People with Special Needs*, ser. ICCHP '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 70–77. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70540-6_9

[6] A. J. Glenstrup and T. Engell-Nielsen, "Eye controlled media: Present and future state," vol. 1, 1995. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.6067

[7] C. H. Morimoto and M. R. M. Mimica, "Eye gaze tracking techniques for interactive applications," *Comput. Vis. Image*

Underst., vol. 98, pp. 4–24, April 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1061935.1649095

[8] S. Rosa and prof. Fulvio Corno, "Nuovi metodi di interazione con ambienti virtuali mediante gaze tracking." Politecnico di Torino, IT: Facoltà di Ingegneria - Corso di laurea in Ingegneria Informatica, Gennaio 2008.

[9] [Online]. Available: http://www.oculist.net/downaton502/prof/ebook/duanes/pages/v3/v3c005.html

[10] Wikipedia, The Free Encyclopedia, "Electrooculography," 2011. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Electrooculography&oldid=431859136

[11] M. Brown, M. Marmor, Vaegan, E. Zrenner, M. Brigell, and M. Bach, "Iscev standard for clinical electro-oculography (eog) 2006," *Documenta Ophthalmologica*, vol. 113, pp. 205–212, 2006, 10.1007/s10633-006-9030-0. [Online]. Available: http://dx.doi.org/10.1007/s10633-006-9030-0

[12] A. Bulling, J. A. Ward, H. Gellersen, and G. Troster, "Eye movement analysis for activity recognition using electrooculography," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 741–753, 2011.

[13] [Online]. Available: http://www.tobii.com/

[14] C. H. Morimoto, D. Koons, A. Amir, and M. Flickner, "Pupil detection and tracking using multiple light sources," *Image and Vision Computing*, vol. 18, no. 4, Mar. 2000.

[15] Y. Ebisawa, "Improved video-based eye-gaze detection method," in *IEEE Transactions on Instrumentation and Measurement , Vol. 47, N. 4, August 1998.* Member, IEEE.

[16] Q. Ji and Z. Zhu, "Eye and gaze tracking for interactive graphic display," in *Proceedings of the 2nd international symposium on Smart graphics*, ser. SMARTGRAPH '02. New York, NY, USA: ACM, 2002, pp. 79–85. [Online]. Available: http://doi.acm.org/10.1145/569005.569017

[17] M. K. Stanford, "Reducing the cost of eye tracking systems," Gates Building, Stanford, CA 94303-9035,USA, 2006.

[18] A.Pérez, M.L.Córdoba, A.García, R.Méndez, M.L.Muñoz, J.L.Pedraza, and F.Sánchez, "A precise eye-gaze detection and tracking system," Spain, 288660 Boadilla del Monte, Madrid, Tech. Rep., 2003.

[19] D. W. Hansen and R. I. Hammoud, "An improved likelihood model for eye tracking," *Comput. Vis. Image Underst.*, vol. 106, pp. 220–230, May 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1242862.1243344

[20] Li, Dongheng and Winfield, D. and Parkhurst, D. J., "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," vol. 3, 2005, p. 79. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.531

[21] M. R. Clark, "A two-dimensional purkinje eye tracker," Stanford Research Institute, Menlo Park, California 94025, Tech. Rep.

[22] L. qun Xu, D. Machin, P. Sheppard, M. Heath, and I. I. Re, "A novel approach to real-time non-intrusive gaze finding," 1998.

[23] D. Torricelli, S. Conforto, M. Schmid, and T. D'Alessio, "A neural-based remote eye gaze tracker under natural head motion," vol. 92. New York, NY, USA: Elsevier North-Holland, Inc., October 2008, pp. 66–78. [Online]. Available: http://portal.acm.org/citation.cfm?id=1410474.1410691

[24] J. P. Ivins and J. Porrill, "A deformable model of the human iris for measuring small three-dimensional eye movements," vol. 11. Secaucus, NJ, USA: Springer-Verlag New York, Inc., June 1998, pp. 42–51. [Online]. Available: http://portal.acm.org/citation.cfm?id=293913.293920

[25] M. Betke, W. J. Mullally, and J. J. Magee, "Active detection of eye scleras in real time," in *In IEEE CVPR Workshop on Human Modeling, Analysis and Synthesis*, 1999.

[26] D. Li and D. Parkhurst, "Open-source software for real-time visible-spectrum eye tracking," in *The 2nd Conference on Communication by Gaze Interaction – COGAIN 2006, Turin, Italy.* Iowa State University, USA: Human Computer Interaction Program.

[27] L. G. Kourkoutis, K. I. Panoulas, and L. J. Hadjileontiadis, "Automated iris and gaze detection using chrominance: Application to human-computer interaction using a low resolution webcam," in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence - Volume 01*, ser. ICTAI '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 536–539. [Online]. Available: http://dx.doi.org/10.1109/ICTAI.2007.60

[28] I. Signorile and prof. Fulvio Corno, "Studio e realizzazione di un sistema di puntamento oculare per l'accesso al computer da parte di disabili motori gravi." Politecnico di Torino, IT: Facoltà di Ingegneria - Corso di laurea in Ingegneria Informatica, 2002.

[29] V. Kruger, A. Happe, and G. Sommer, "Affine real-time face tracking using gabor wavelet networks," in *Proceedings of the International Conference on Pattern Recognition - Volume 1.* Computer Science Institute, Christian-Albrechts University Kiel, Germany.

[30] Wikipedia, The Free Encyclopedia, "Wavelet," 2011. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Wavelet&oldid=432323380

[31] [Online]. Available: http://www.codeproject.com/KB/cpp/TrackEye.aspx

[32] [Online]. Available: http://www.eyetechds.com/

[33] [Online]. Available: http://www.dynavoxtech.com/

[34] [Online]. Available: http://www.eyegaze.com/

[35] [Online]. Available: http://www.srlabs.it/index.html

[36] [Online]. Available: http://www.gazegroup.org/research/15

[37] [Online]. Available: http://www.tomweber-software.com/products/onscreenkeys/1040/main.htm

[38] [Online]. Available: http://www.inference.phy.cam.ac.uk/dasher/

[39] E. Castellina, F. Corno, and P. Pellegrino, "Integrated speech and gaze control for realistic desktop environments," in *Proceedings of the 2008 symposium on Eye tracking research & applications*, ser. ETRA '08. New York, NY, USA: ACM, 2008, pp. 79–82. [Online]. Available: http://doi.acm.org/10.1145/1344471.1344492

[40] L. D. Russis, prof. Fulvio Corno, and dott. Emiliano Castellina, "Interfaccia utente basata su eye-tracking per sistemi di controllo ambientale." Politecnico di Torino, IT: Facoltà di Ingegneria - Corso di laurea in Ingegneria Informatica, Febbraio 2010.

[41] C. Fausto and prof. Fulvio Corno, "Modelli e interfacce di comunicazione testuale per sistemi di eye tracking." Politecnico di Torino, IT: Facoltà di Ingegneria - Corso di laurea in Ingegneria Informatica, Gennaio 2007.

[42] [Online]. Available: http://www.cogain.org/

[43] [Online]. Available: http://www.cogain.org/wiki/COGAIN_Reports

[44] R. Bates, G. Daunys, A. Villanueva, E. Castellina, G. Hong, H. Istance, A. Gale, V. Lauruska, O. Spakov, and P. Majaranta, "D2.4 a survey of existing 'de-facto' standards and systems of environmental control." Communication by Gaze Interaction (COGAIN): IST-2003-511598: Deliverable 2.4, 2006.

[45] F. Corno, E. Castellina, R. Bates, P. Majaranta, H. Istance, and M. Donegan, "D2.5 draft standards for gaze based environmental control." Communication by Gaze Interaction (COGAIN): IST-2003-511598: Deliverable 2.5, 2007.

[46] M. Donegan, L. Oosthuizen, R. Bates, G. Daunys, J. Hansen, M. Joos, P. Majaranta, and I. Signorile, "D3.1 user requirements report with observations of difficulties users are experiencing." Communication by Gaze Interaction (COGAIN): IST-2003-511598: Deliverable 3.1, 2005.

[47] Donegan et al., "D3.2 report on features of the different systems and development needs." Communication by Gaze Interaction (COGAIN): IST-2003-511598: Deliverable 3.2, 2006.

[48] [Online]. Available: http://www.bc.edu/schools/csom/eagleeyes/

[49] [Online]. Available: http://asleyetracking.com/Site/

[50] [Online]. Available: http://www.crsltd.com/index.html

[51] [Online]. Available: http://www.brain.northwestern.edu/ilab/

[52] [Online]. Available: http://www.gazegroup.org/home

[53] D. W. Hansen and A. E. C. Pece, "Eye tracking in the wild," *Comput. Vis. Image Underst.*, vol. 98, pp. 155–181, April 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1061935.1649098

[54] [Online]. Available: http://thirtysixthspan.com/openEyes/

[55] [Online]. Available: http://www.inference.phy.cam.ac.uk/opengazer/

[56] [Online]. Available: http://myeye.jimdo.com/

[57] [Online]. Available: http://hci.stanford.edu/research/GUIDe/index.html

[58] [Online]. Available: http://mmlab.disi.unitn.it/

[59] A. Armanini and N. Conci, "Eye tracking as an accessible assistive tool." Multimedia Signal Processing and Understanding Lab., DISI - University of Trento , Via Sommarive, 14 , 38123 Trento - Italy: 11th International Workshop on Image Analysis for Multimedia Interactive Services, Piascataway, 2010.

[60] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, April 1981, pp. 674–679.

[61] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," 2000. [Online]. Available: http://robots.stanford.edu/cs223b04/algo_tracking.pdf

[62] [Online]. Available: http://wiki.services.openoffice.org/wiki/Main_Page

[63] [Online]. Available: http://openmp.org/

[64] Wikipedia, The Free Encyclopedia, "Openmp," 2011. [Online]. Available: http://en.wikipedia.org/w/index.php?title=OpenMP&oldid=422276000

[65] [Online]. Available: http://www.graphytech.it/

[66] [Online]. Available: http://www.parrocchiamssdesolata.it/

[67] [Online]. Available: http://www.ircvenezia.it/

[68] [Online]. Available: http://opencv.willowgarage.com/documentation/object_detection.html

[69] P. A. Viola and M. J. Jones, "Robust real-time face detection," in *ICCV*, Vancouver, Canada, 2001, p. 747.

[70] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *IEEE ICIP 2002*, Intel Labs, Intel Corporation, Santa Clara, CA 95052, USA, 2002, pp. 900–903.

[71] [Online]. Available: http://sites.google.com/site/ferrariimageprocessing/

[72] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, August 1997.

[73] [Online]. Available: http://note.sonots.com/SciSoftware/haartraining.html

[74] [Online]. Available: http://www.imagemagick.org/script/identify.php

[75] Wikipedia, The Free Encyclopedia, "Opencv," 2011. [Online]. Available: http://en.wikipedia.org/w/index.php?title=OpenCV&oldid=425549724

[76] [Online]. Available: http://www.cognotics.com/opencv/servo_2007_series/index.html

[77] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library.* Cambridge, MA: O'Reilly, 2008.

[78] [Online]. Available: http://tech.groups.yahoo.com/group/OpenCV/

[79] Wikipedia, The Free Encyclopedia, "Purkinje images," 2011. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Purkinje_images&oldid=412183918

[80] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The feret evaluation methodology for face recognition algorithms," 2000.

[81] [Online]. Available: http://face.nist.gov/colorferet/

Elenco delle figure

Ringraziamenti

Il primo ringraziamento spetta di diritto al prof. Sergio Benini e al prof. Nicola Adami per la disponibilità e la cortesia dimostratami; particolarmente preziosi sono gli insegnamenti e la visione ingegneristica con cui mi hanno spinto ad affrontare il periodo di stage, dimostrandomi quanto e dove posso ancora migliorare.

Un sentito grazie a Marco che con la sua sagacia e l'interminabile riserva di buon umore ha reso il periodo di stage davvero piacevole e proficuo. Nonostante non vengano citati ricordo affettuosamente tutti i compagni d'università che hanno percorso con me quest'avventura.

Un particolare ringraziamento ai miei genitori, che mi hanno premurosamente accudito e supportato sia materialmente, ma soprattutto moralmente nell'arco della mia carriera universitaria affrontando con me i momenti più difficili. Una dedica speciale spetta a Silvia, la persona che più di tutte mi è stata vicina in questi anni e che sempre mi ha sostenuto e consigliato.

Un pensiero particolare lo dedico agli amici di lunga data che mai hanno smesso di starmi accanto nonostante i miei ripetuti errori e di avermi concesso in questi anni gli spazi di cui necessitavo.