Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali

Corso di Laurea Magistrale in Informatica

RICONOSCIMENTO DI GESTI: estensione a movimenti fini su spazi limitati. Teoria e implementazione

Tesi di Laurea in Sistemi e Applicazioni Multimediali

Relatore:
Chiar.mo Prof. Marco Roccetti

Presentata da:
Andrea Marcomini

Co-relatore:
Dott. Gustavo Marfia

Sessione I

Anno Accademico 2011/2012

Alla mia famiglia, Tutta.

Indice

Introduzione

Riconoscere un gesto, tracciarlo ed identificarlo è una operazione complessa ed articolata. Negli ultimi anni, con l'avvento massivo di interfacce interattive sempre più sofisticate, si sono ampliati gli approcci nell'interazione tra uomo e macchina. L'obiettivo comune, è quello di avere una comunicazione "trasparente" tra l'utente e il computer, il quale, deve interpretare gesti umani tramite algoritmi matematici.

Il riconoscimento di gesti è un modo per iniziare a comprendere il linguaggio del corpo umano da parte della macchina. Questa disciplina, studia nuovi modi di interazione tra questi due elementi e si compone di due macro obiettivi :

- *tracciare i movimenti di un particolare arto;*
- *riconoscere tale tracciato come un gesto identificativo.*

Ognuno di questi due punti, racchiude in sé moltissimi ambiti di ricerca perchè moltissimi sono gli approcci proposti negli anni. Non si tratta di semplice cattura dell'immagine, è necessario creare un supporto, a volte molto articolato, nel quale i dati grezzi provenienti dalla fotocamera, necessitano di filtraggi avanzati e trattamenti algoritmici, in modo tale da trasformare informazioni grezze, in dati utilizzabili ed affidabili. La tecnologia riguardo la *gesture recognition* è rilevante come l'introduzione delle interfacce tattili sui telefoni intelligenti. L'industria oggi ha iniziato a produrre dispositivi in grado di offrire una nuova esperienza, la più naturale possibile, agli utenti. Dal videogioco, all'esperienza televisiva gestita con dei piccoli gesti, all'ambito biomedicale, si sta introducendo una nuova generazione di dispositivi i cui impieghi sono innumerevoli e, per ogni ambito applicativo, è necessario studiare al meglio le peculiarità, in modo tale da produrre un qualcosa di nuovo ed efficace.

Questo lavoro di tesi ha l'obiettivo di apportare un contributo a questa disciplina. Ad oggi, moltissime applicazioni e dispositivi associati, si pongono l'obiettivo di catturare movimenti ampi: il gesto viene eseguito con la maggior parte del corpo e occupa una posizione spaziale rilevante. Questa tesi vuole proporre invece un approccio, nel quale i movimenti da seguire e riconoscere sono fatti "nel piccolo". Si avrà a che fare con gesti classificati *fini*, dove i movimenti delle mani sono compiuti davanti al corpo, nella zona del torace, ad esempio. Gli ambiti applicativi sono molti, in questo lavoro si è scelto ed adottato l'ambito artigianale.

L'artigianato italiano è frutto di quel connubio tra cultura umanistica e cultura tecnologica che è conosciuto in tutto il mondo come "Made in Italy". Come nell'idea del modello Bauhaus, il sapere artigiano, è frutto dell'esperienza e della tecnica apprese, nel corso del tempo, attraverso un percorso tradizionale e consolidato nel progettare e realizzare un particolare prodotto. Al giorno d'oggi il modello di insegnamento "maestro-apprendista" non è più comune come una volta per cui non è facile insegnare e tramandare le abilità e le conoscenze apprese da artigiani specializzati, i quali sono capaci di creare e sviluppare sofisticati prodotti artigianali unici, in un impeccabile bilanciamento tra tradizione e modernità.

Lo scopo di questo lavoro, è proporre un approccio di codifica della conoscenza manuale e artigianale non invasivo, il quale, con specifici algoritmi, riesce a catturare e riconoscere i *gesti fini* compiuti, senza l'uso di indumenti e strumenti particolari, ma solamente con l'ausilio di una semplice webcam. Verranno presentati i risultati ottenuti posizionando la webcam in posizione frontale rispetto all'artigiano all'opera, attraverso quattro esempi di gesti tipici, atti alla produzione di una scarpa di alta moda.

E' chiaro che all'interno di un'azienda artigianale è di notevole importanza l'archivio, il quale, è una collezione di saperi materiali e immateriali. Questo quindi è una direzione in cui la tecnologia può fare la differenza, codificando e preservando l'esperienza in supporti digitali. Attraverso l'archivio è possibile inoltre valutare e discutere i processi di design e re-design di un particolare prodotto.

Il nostro lavoro vuole dimostrare che, utilizzando le tecnologie non invasive, è possibile codificare e mantenere la conoscenza dei gesti artigianali attraverso un

sistema di riconoscimento di gesti fini. Partendo dall'idea in cui vi deve essere una forma "pura" di interazione con il sistema, senza l'ausilio di hardware dedicato e specializzato [1] e [38], si è voluto realizzare un sistema che con una sola webcam e particolari algoritmi, riesce a codificare i gesti artigianali. Sono stati scelti quattro gesti significativi, indentificati come casi d'uso, nella lavorazione di scarpe di alta moda.

Tecnicamente, a livello algoritmico, la webcam cattura un frame e lo sottrae al frame precedente, dopo aver applicato ad entrambi i frame un filtro gaussiano per eliminare eventuale rumore e successivamente ne viene calcolata la differenza. Dopo queste azioni preliminari, vengono rilevate le macro aree dove vi è stato rilevato un cambiamento. Le macro aree sono ottenute tramite la suddivisione dell'area inquadrata dalla webcam in una griglia di aree sensibili al movimento. Le macro aree attivate, vengono poi filtrate per eliminare quelle troppo piccole e isolate, e che quindi non rappresentano un movimento significativo. Il nostro approccio attua una clusterizzazione delle aree ottenute con algoritmo Kmeans (K=2); in questo modo si ipotizza, a priori, il movimento sulla scena di due mani. Ovviamente questo non è sempre vero, infatti ci possono essere azioni che richiedono il movimento di una singola mano e altre invece che richiedono il movimento di entrambe. Per discriminare se il movimento è compiuto da una singola mano o da tutte e due, viene considerata la distanza tra i due centri di cluster. Più la distanza supera un fissato valore di soglia, più è probabile che il movimento sia compiuto da due mani; se invece la distanza è inferiore al valore di soglia, molto probabilmente il movimento è effettuato da una singola mano. Determinato il numero di mani in movimento sulla scena, viene calcolata la posizione della mano (o delle mani). Per fare ciò, si considera che, se la mano in movimento è una sola, allora è scelta come posizione della mano la macro area corrispondente al centro di cluster posizionato più in alto nella griglia; se invece ci sono due mani in movimento, vengono considerate le due macro aree corrispondenti alla posizione dei due centri di cluster, calcolati precedentemente e ogni centro di cluster rappresenta la posizione di una mano. Con questa tecnica emergono delle discontinuità nel tracciare i movimenti tra frame successivi; per ovviare a questo disturbo il sistema incorpora un filtro di Kalman che ha il compito di tracciare una traiettoria coerente e fluida lungo la quale è effettuato un particolare

movimento.

Nel primo capitolo verrà introdotto l'ambito di ricerca in cui questo lavoro si sviluppa, cioè quello del riconoscimento di gesti; nel secondo vedremo una carrellata delle varie metodologie e dei vari approcci utilizzati fino ad ora; nel terzo capitolo verrà discusso il funzionamento dell'algoritmo, illustrando un esempio di applicazione pratica; nel quarto viene descritta la struttura concreta, a basso livello, dell'algoritmo, introducendo inoltre le tecniche adottate per la sua realizzazione.I risultati ottenuti con questo approccio verranno discussi nel capitolo conclusivo di questa tesi.

Capitolo 1

Il riconoscimento di gesti e le interfacce naturali

L'interazione tra uomo e computer è una disciplina volta alla valutazione, progettazione e realizzazione di sistemi informatici interattivi che instaurano una comunicazione tra una o più persone, con una o più macchine computazionali. Questo campo di ricerca, si occupa di interpretare i compiti impartiti da esseri umani alle macchine, enfatizzando anche la struttura comunicativa che sussiste tra loro. Di conseguenza, è quindi necessario creare interfacce che siano le più naturali possibili dal punto di vista dell'uomo. Dal lato macchina sono rilevanti tecniche di computer grafica, sistemi operativi, linguaggi di programmazione e ambienti di sviluppo. Dal lato umano entrano in gioco la teoria della comunicazione, le discipline del design industriale, della grafica, della linguistica, delle scienze sociali e della psicologia cognitiva. In questa evoluzione le interfacce si sono continuamente spostate da un punto di vista centrato attorno al computer ad un punto di vista centrato sull' uomo.

La computer grafica è nata molto presto nella storia dei computer, dall'uso di dispositivi CRT e puntatori hardware. Ciò ha portato allo sviluppo di varie tecniche di interazione. Con il passare del tempo, si è continuato a sviluppare algoritmi ed utilizzare hardware, che consentissero una visualizzazione e manipolazione di oggetti, sempre più realistici (come la rappresentazione di componenti CAD o immagini mediche). I lavori riguardanti i sistemi operativi, nel frattempo,

si sono concentrati anche sullo sviluppo di tecniche per l'interfaccia di dispositivi input-output, per il tempo di risposta del sistema all'interazione dell'utente, per il multiprocessing, e per il sostenimento degli ambienti di animazione. I fattori umani, come disciplina, hanno un ruolo importante in quanto devono essere considerati aspetti cognitivi delle persone e la loro comunicazione, il tutto per fornire una interfaccia utente naturale[1].

Per decenni, le persone hanno operato con i computer e altri dispositivi, quasi esclusivamente toccando i tasti e facendo "clic" sul mouse. Vi è, ad oggi, un grande cambiamento in atto, a seguito di una ondata di nuove tecnologie che rispondono alle azioni più naturali, come il movimento delle mani o dell'intero corpo. Prodotti come l'iPhone, la console Wii di Nintendo o la Xbox di Microsoft, hanno già scosso i mercati sostituendo le tecniche di interazione standard, con approcci di tipo "touch and motion sensing"[2]. Il passo successivo è l'introduzione di **interfacce utente naturali**. Con questo termine, utilizzato anche da designer e sviluppatori di interfacce, ci si riferisce a interfacce completamente invisibili[3]. Lo scopo, è realizzare un componente in grado di controllare un'applicazione, o manipolare dei contenuti nel computer, tramite l'utente, il quale interagisce con il sistema solamente con gesti e azioni del corpo; un aspetto importante è la velocità di apprendimento dell'utente nell'utilizzo di tali interfacce.

1.1 Parametri della computer vision su Interfacce

Lo sviluppo di interfacce deve tenere in considerazione una serie di parametri aventi un ruolo importante:

- *robustezza*: le condizioni variano molto da ambiente ad ambiente, risulta quindi necessario sviluppare interfacce che tengano conto di queste situazioni e che siano in grado di adattarsi alle diverse variazioni di luce, di ambiente e di sfondo;

[1] Natural User Interface. Con il termine naturale viene inteso il fatto che i computer utilizzano dispositivi di controllo che richiedono un apprendimento.

[2] Dispositivi sensibili al tocco e al movimento.

[3] Oppure lo diventano in un secondo momento dopo successive interazioni da parte degli utenti.

- *velocità*: essendo la comunicazione in tempo reale, è necessario offrire uno streaming video sufficientemente alto, in modo da utilizzare questa tipologia di input per tutta l'elaborazione software che deve essere fatta;

- *costo*: con l'abbattimento dei prezzi dei dispositivi hardware si è potuto diffondere nuove tipologie di interazione. Normalmente i sistemi odierni incorporano, al loro interno, processori dedicati e ottimizzazione di codice per interfacce di compressione ed elaborazione immagine;

- *efficacia del gesto*: la comunicazione gestuale deve essere veloce e concisa, per minimizzare le ambiguità tra gesti simili e offrire un tempo di codifica reattivo;

- *comfort*: deve essere confortevole interagire con il sistema; tipologie di interfacce che richiedono indumenti con sensori, posso offrire un'esperienza utente complicata o scomoda;

- *conoscenza a priori del gesto*: il sistema deve sapere a priori cosa deve riconoscere. Sia che si tratti di pose o di movimenti, è necessario inserire della conoscenza in modo tale da permettere alla macchina di basare le sue decisioni su un insieme di azioni ammissibili;

- *discriminare il rumore*: è necessario isolare il movimento principale da quelli secondari, o dovuti a rumore del dispostivo di cattura;

- *discriminare i passi di un gesto*: i gesti generalmente si compongono di più passi e di solito il cambiamento tra uno e l'altro è continuo. Per codificare efficacemente un gesto globale, il computer deve essere in grado di scomporre la sequenza continua ed interpretare tutte queste sotto-azioni, indipendentemente tra loro.

L'utilizzo di comandi gestuali per interagire con i sistemi, rende l'approccio, da parte dell'utente, più naturale. Sistemi che riescono a catturare i movimenti di particolari arti del corpo sono espressivamente più efficaci. Da un punto di vista cognitivo, l'utente diventa, in prima persona, il controller di input, non dipendendo così da dispositivi intermedi.

1.2 Comprensione della gestualità

I gesti fanno parte del linguaggio del corpo e veicolano una comunicazione non verbale. Vi sono due tipologie di gestualità, *comunicativa* e *movimenti involontari*. La prima tipologia si riferisce a tutti i movimenti intesi in ambito sociale che hanno lo scopo di veicolare un contenuto semantico; la seconda tipologia rappresenta quei movimenti che sono presenti all'interno del dialogo, ma non riguardano la comunicazione volontaria[4]. Questa distinzione da un lato introduce margini di ambiguità, dall'altro però fornisce un modello rappresentativo motorio e gestuale.

A livello computazionale, si categorizzano i gesti sulla base della loro funzione durante una interazione. Vi sono così, altre due tipologie di gesti: *manipolativi* e *comunicativi*. I primi sono effettuati per agire su oggetti dell'ambiente, come spostamenti e rotazioni, i secondi, sono collegati alla comunicazione verbale e possono rappresentare un simbolo[5] o un atto.

I gesti si distinguano in due tipologie:

- *statici*: dove viene considerata una determinata posizione della mano e delle dita, in un particolare istante di tempo;
- *dinamici*: dove viene considerata una sequenza temporale di gesti statici nel tempo;

Partendo dalle specifiche di dispositivo, si devono prendere in considerazione solamente dei set di gesti interpretabili dalla specifica macchina. Ovviamente questi gesti devono appartenere a specifiche convenzioni etniche-sociali.

Un modello generico e standard di interazione, che esula da specifiche tecniche, si basa sul modello descritto in [48], dove viene schematizzata l'interazione di un sistema attraverso i gesti. Viene considerata una zona attiva rappresentante un'area di ripresa della camera. I gesti sono catturati solamente quando la mano si trova all'interno dell'inquadratura. Ogni sequenza di gesti è rappresentata da posizioni iniziali, che hanno lo scopo di far partire un comando, seguito poi da una sequenza di posizioni assunte dall'arto sino ad arrivare ad una posizione finale che determina la fine della sequenza di comandi.

[4]Esprimono sensazioni come rabbia, felicità, curiosità, ecc...
[5]I simboli hanno un corrispettivo linguistico codificato.

L'inserimento di un gesto nel dialogo tende anche a semplificare l'identificazione, quando l'identificazione dell'oggetto è difficile. Si supponga per esempio di voler identificare un preciso quadrato, scelto dall'utente, che si trova in una deterimata posizione, vicino ad altri quadrati posti in altre posizioni. Non avendo proprietà che ne porti ad una identificazione diretta, si deve ricorrere alla posizione spaziale. Il tutto sarebbe più immediato se si indicasse con un dito l'obiettivo desiderato.

1.3 Ambiti di applicazione

Le tecnologie basate su interfacce naturali hanno seguito un lungo percorso di inserimento nel mercato di massa, ed ora si trovano in numerosi ambiti e contesti di applicazione. A metà degli anni '80, si stava già lavorato su tavolette "multitouch" per svolgere compiti su uno schermo, utilizzando più di un dito. La tecnologia finalmente compì un grande passo in avanti con la distibuzione dell'iPhone (nell'ambito della telefonia), dotato unicamente di un grande schermo non ricoperto di pulsanti fisici, in grado di eseguire zoom avanti e indietro su fotografie e mappe, solamente con gesti quali il pizzicare due dita assieme.

In un contesto videoludico, Nintendo, nel frattempo, ha voluto dare ai videogiocatori operazioni di gioco più semplici rispetto all'esecuzione di complesse sequenze di pulsanti, necessari nei controller tradizionali. La Nintendo Wii ha compiuto un'innovazione circa il concetto di controller, offrendo agli utenti la sensazioni di giocare a giochi come il tennis e il tiro con l'arco imitando i movimenti proprio di quegli sport. Successivamente Sony corporation, ha rilasciato la propria versione di **motion-sensing controller** per la console PlayStation 3.

Figura 1.1: Controller con accelerometro: (sinistra) Nintendo Wii, (destra) Playstation Move

Secondo John Seely Brown, ex direttore di Xerox PARC[6], questi dispositivi sono più naturali perché "condividono lo spazio con noi", attraverso la rilevazione di un'azione compiuta con il controller dotato di accelerometri. Sempre nel medesimo ambito, Microsoft e Asus hanno compiuto uno step successivo, eliminando la necessità di un controller fisico e basandosi interamente su una telecamera di profondità, avente lo scopo di rilevare i movimenti dei giocatori. In altre parole, l'utente diventa lui stesso il controller.

Figura 1.2: (sinistra) Asus Xtion Pro, (destra) Microsoft Kinect

Sicuramente l'ambito dei videogiochi è stato il punto di partenza, ma queste nuove tipologie di interfacce possono essere usate in altri contesti, anche molto diversi tra loro. Un famoso modo di dire afferma che una foto vale più di mille parole: un'esperienza basata sull'esecuzione di un gesto, probabilmente, vale più di mille immagini. Per un contesto educativo, nel quale sia richiesta l'esecuzione di mosse specifiche (es. sport, sicurezza, fasi di training, ecc...), il gesto è la tecnologia più conveniente ed efficiente da utilizzare: nessun hardware specifico da indossare, solamente una camera di acquisizione video posta a distanza per monitorare le azioni delle persone.

Ambito Artistico

Nel campo dell'arte, numerosi artisti e performer utilizzano il riconoscimento di gesti per offrire performance ed installazioni più coinvolgenti ed emozionanti. Numerosi sono i casi in cui vi sono superfici che comunicano con le persone attraverso l'interpretazione dei gesti. Con piccoli movimenti del corpo, si interagisce con immagini, suoni e contenuti digitali. L'opera dell'artista viene così usata e vissuta dal pubblico in un modo più diretto. E per recepire messaggi che riman-

[6]Il laboratorio di ricerca di Xerox che ha introdotto per la prima volta le tecnologie di interfaccia utente.

gono impressi più a lungo nella memoria. Tutto questo crea installazioni versatili che rendono più semplice, oltre che più emozionante, la visita a musei, fiere ed esposizioni.

Figura 1.3: Esempi di installazioni interattive con muri e tappeti sensibili al movimento

Nell'ambiente dell'entertainment e nell'industria cinematografica, ma anche nello spettacolo e nelle simulazioni militari, viene utilizzata la tecnica chiamata **motion capture**. Questa tecnica consiste nella registrazione dei movimenti di oggetti e persone. Viene registrata l'azione degli attori e le informazioni di movimento così catturate, vengono utilizzate per animare modelli di personaggi digitali nell'animazione computerizzata, sia 2D che 3D. Nelle sessioni di motion capture, i movimenti di uno o più attori sono campionati molte volte al secondo, con l'utilizzo di diverse telecamere e sensori indossabili. Questi dati vengono poi associati a un modello 3D che segue le medesime azioni dell'attore. Un computer elabora i dati e visualizza i movimenti dell'attore, fornendo una panoramica di posizione e posa degli oggetti nella scena.

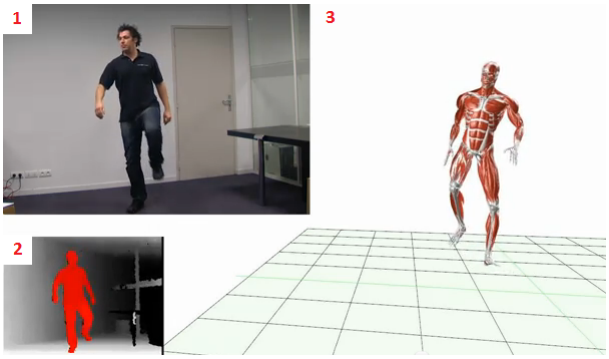


Figura 1.4: Esempio di motion capture: (1) movimento eseguito dall'attore, (2) specifico algoritmo di tracking, (3) modellazione grafica 3D basata sulla posa rilevata dal software

Ambito Medico

Nelle sale operatorie di oggi, le immagini virtuali possono essere difficili da recuperare e manipolare: un chirurgo non può utilizzare un mouse o una tastiera, non sono oggetti sterili e presentano un rischio di infezione. Sicuramente un chirurgo può dare istruzioni a qualcun altro, che poi utilizzerà un mouse o una tastiera, ma il chirurgo deve interrompere la sua concentrazione per "pilotare" l'assistente. Per semplificare questo processo si sono creati dei dispositivi che usano il natural user interface. Esso consente ai medici di utilizzare semplici gesti delle mani per cambiare, spostare o ingrandire scansioni di immagini mediche. Questi dispositivi sono stati testati in diversi ospedali[7], ad esempio in interventi cardio vascolari. I chirurghi vascolari spesso tentano di evitare di eseguire interventi a cuore aperto, tuttavia, laddove questo si rende necessario, deve esserci una grandissima precisione circa il punto corretto in cui dove intervenire. L'interazione con immagini digitali molto accurate, visionate senza l'uso di interfacciamento con dispositivi hardware, può essere di grande aiuto. Utilizzando dei sensori, in combinazione con un monitor di grandi dimensioni, un chirurgo può facilmente modificare la visualizzazione dell'immagine della zona del torace di un paziente, guardare una nuova immagine o ingrandire sui dettagli, il tutto senza toccare uno schermo, una tastiera, o un mouse. Le immagini sono in 3D e vengono acquisite prima dell'intervento con una tomografia computerizzata.

Figura 1.5: Esempio di gesture recognition in campo biomedicale

[7]Uno di questi è riferito al progetto Touchless Interaction in Medical Imaging, sviluppato da Microsoft Research.

Ambito dell'interpretazione dei gesti

Esistono differenti tipi di gesti, però gli insiemi più strutturati appartengono al linguaggio dei segni. Nel linguaggio dei segni, ogni gesto ha associato un significato. Forti regole di contesto e grammatica possono essere applicate per rendere il riconoscimento trattabile. American Sign Language (ASL) è la lingua scelta per la maggior parte di persone non udenti negli Stati Uniti. ASL utilizza circa 6000 gesti per parole comuni e uno spelling a livello di dita per comunicare parole particolari o nomi propri. In [18] vengono descritti due sistemi estensibili che utilizzano una telecamera a colori per tenere traccia della posa delle mani, in tempo reale, con lo scopo di interpretare il linguaggio dei segni americano. La fase di analisi del sistema non tenta una descrizione accurata della forma di una mano, ma si concentra sull'evoluzione del gesto nel tempo. Si ha un sistema di visione basato sul metodo di riconoscimento di una frase, codificata da ASL, che permette anche di indossare una camera in grado di analizzare e classificare le varie pose della mano.

Figura 1.6: Pose della mano riferite al linguaggio dei segni americano

Ambito di interazione su televisori interattivi

Diversi modelli di smart tv di fascia alta possono essere comandati con la voce, con i gesti e con il viso. Per essere notati, la mano deve essere portata a livello della testa e agitata con una leggera velocità. Una volta attirato l'interesse del dispositivo, compare a destra e a sinistra un indicatore con i parametri relativi a

volume e canali. Salutare uno schermo può sembrare demenziale all'inizio ma ci si abitua prestissimo. Tenendo chiuso il pugno poi si scorrono velocemente canali e volume. L'interfaccia gestuale funziona senza troppe difficoltà e anche la risposta ai comandi è pressoché immediata. Il movimento è rilevato da una webcam posta sulla cornice in alto del televisore. Più complicato è il lavoro per il software, che deve elaborare le informazioni catturate. Vi è anche il sistema di riconoscimento del volto incorporato. Il sistema riconosce la persona e, previa impostazione, usa le password che ha inserito per permettere in automatico l'accesso a Skype, Facebook e altri social network. La webcam effettua solamente il riconoscimento facciale. L'aspetto paradossale di queste nuove interfacce è che vengono definite naturali, ma richiedono più energie di un controller fisico. L'integrazione però tra voce, telecomando e gestualità, funziona bene.

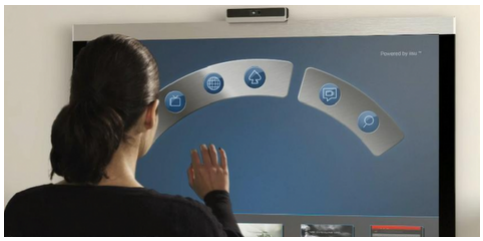


Figura 1.7: Interazione con il televisore senza l'uso di telecomando

Approccio del lavoro di tesi

Il riconoscimento dei gesti, come sopra esemplificato, si sta diffondendo in numerose aree ed ambiti, molto distinti tra loro. La maggior parte dei rilevamenti però, viene fatta su posizioni o movimenti macro, in uno spazio spesso ampio. L'approccio seguito in questo lavoro è, invece, focalizzato sul riconoscere e seguire piccoli movimenti della mano eseguiti su spazi ristretti. Attraverso l'utilizzo di aree sensibili al movimento, si vuole riuscire ad individuare la posizione delle mani nello spazio e a tenere traccia del movimento eseguito. Questo approccio non richiede l'utilizzo di indumenti particolari o hardware specializzato. Solamente con una webcam (o camera di acquisizione video generica), posizionata di fronte all'esecutore, è possibile tenere traccia dei movimenti delle mani. L'ambito da cui

questo lavoro trae spunto è quello della codifica di gesti e movimenti, classificati fini, di maestri artigiani durante il loro lavoro di creazione.

Capitolo 2

Contesto scientifico e tecnologico analizzato

Il riconoscimento di gesti (*gesture recognition*) si pone l'obiettivo di interpretare e discriminare diverse tipologie di gesti umani attraverso algoritmi. Ci sono innumerevoli gesti che, presi singolarmente o combinati tra loro, possono essere generati da qualsiasi parte del corpo. Interpretare questi movimenti è di fondamentale importanza nel progettare un'interfaccia efficace ed efficiente di *interazione uomo-macchina*, creando così una comunicazione più naturale, senza l'ausilio di particolare hardware d'acquisizione. Attraverso tecniche di visione artificiale e di elaborazione dell'immagine, è possibile creare algoritmi in grado di codificare la natura ed il significato del gesto compiuto, in modo tale da impiegare queste applicazioni in numerosi ambiti, come ad esempio, il riconoscimento del linguaggio dei segni [18], la robotica applicata alla riabilitazione di pazienti affetti da varie patologie, le tecniche di puntamento con marcatori o con l'ausilio di indumenti particolari [19], l'analisi di gesti facciali [20], le tecnologie di gioco immersive [1] e i controller virtuali.

Un gesto può essere visto anche come una tecnica di compressione delle informazioni che devono essere inviate ad un ricevitore, per poi essere elaborate. Dal momento che i movimenti sono fatti da esseri umani, questi possono rappresentare, a parità di esecuzione, significati differenti a seconda della posizione geografica, della cultura e delle usanze locali. Questo introduce delle ambiguità semantiche

e contribuisce ad incrementare il grado di difficoltà nell'interpretazione da parte di dispositivi elettronici.

Sono stati introdotti diversi approcci e metodologie per elaborare tutte queste informazioni: a seconda del movimento che si intende studiare, vengono presi in considerazione algoritmi con determinate peculiarità, in grado di catturare quelle proprietà distintive del movimento da riconoscere.

La tipologia di gesti può essere *statica*, dove si prende in considerazione una particolare posizione, oppure *dinamica,* dove il gesto è composto da una serie di posizioni, rappresentate da fotogrammi in successione e scandite dal tempo. Alcuni movimenti, per essere riconosciuti, richiedono un fotogramma di partenza e un fotogramma di arrivo, che sanciscono, rispettivamente, l'inizio e la fine del movimento.

Tipicamente, quando si parla di gesti, possiamo individuare tre "parti" del corpo e i relativi aspetti da tenere in considerazione in fase di riconoscimento:

- **mani e braccia**: stima della posizione e postura di questi arti, utilizzati ad esempio per il riconoscimento del linguaggio dei segni;
- **testa e viso**: stima della direzione dello sguardo, movimenti della muscolatura facciale, dove un uso applicativo è quello dell'interpretazione di stati emozionali;
- **tutto il corpo**: individuazione di persone in movimento, calcolo e stima della postura del corpo di una persona, utilizzati ad esempio per analisi in campo medico e riabilitativo

Il riconoscimento di gesti, come argomentato da [2], è un campo di ricerca multidisciplinare: partendo dagli scopi dell'applicazione che si vogliono raggiungere, sono presenti in letteratura numerosi approcci molto differenti tra loro che spaziano tra modellizzazione statistica, visione artificiale, pattern recognition e image processing. Nel seguito verranno esposti i lavori ed approcci più significativi, presenti in letteratura, per risolvere particolari problemi di riconoscimento. C'è da premettere, sin da ora, che ogni tipologia di gesto ha delle proprie caratteristiche e questo comporta il fatto che, di volta in volta, in base a quello che ci si prefigge di interpretare artificialmente, si ricorre a questo o quel metodo differementemente.

Vi sono algoritmi che enfatizzano, ad esempio, il percorso effettuato (*algoritmi di tracking*), oppure la postura (*pose estimation*), con tecniche di elaborazione di post-produzione a causa del notevole costo computazionale [16], o ancora la forma (*Hough transformation*), l'analisi di istogrammi (*color detection*), il movimento (*motion detection*), i movimenti facciali (*face recognition*).

2.1 Analisi sul colore: skin detection come parametro di tracciamento

Per il tracciamento di elementi aventi determinati colori, si utilizza questo approccio che consente l'individuazione di aree, nei singoli frame, che rappresentano la posizione di un particolare colore cercato. Un approccio di ricerca è quello riguardante il riconoscimento del colore della pelle, atto ad individuare parti del corpo all'interno di ogni frame. Questo processo è tipicamente usato, come passo di pre-processing, per trovare le regioni che potenzialmente hanno volti umani e arti, all'interno di immagini. Un rilevatore del colore della pelle tipicamente trasforma un dato pixel in uno spazio di colore appropriato e, successivamente, attua una classificazione per determinare, tramite un valore di soglia di un determinato colore [4], se in quel pixel vi è la presenza di pelle. Una variante di tale procedimento, si compone di due fasi: *fase di training* e *fase di rilevamento*. L'apprendimento (o fase di training) di uno **skin detector** si compone di tre passi:

- formazione di un database, di esempi di pelle, con immagini provenienti da differenti sorgenti; tale raccolta contiene esempi catturati con differenti esposizioni alla luce;
- scelta di uno spazio di colore adatto [5](RGB[1], HSV[2], HSI[3]); questo influisce anche sulle prestazioni dell'algoritmo;
- apprendimento dei parametri, tramite albero decisionale.

[1]Classico modello di colori additivo basato sui colori rosso, verde e blu.
[2]Hue Saturation Value - tinta, sfumatura e tono. Il sistema di coordiante è conico.
[3]Hue Saturation Intensity - basato su tonalità, saturazione ed intensità

Una volta definito il classificatore, per individuare i pixel della pelle in un dato frame si procede come segue:

- conversione dell'immagine in uno spazio di colore prestabilito, che è lo stesso di quello utilizzato durante la fase di training del classificatore;
- classificazione di ogni singolo pixel del frame;
- fase di post-produzione che utilizza la morfologia per determinare una omogeneità sulle regioni rilevate

Per riconoscere, ad esempio, una mano, frame per frame, viene tracciato un percorso caratterizzato dalle posizioni delle varie aree rilevate nel tempo. Vi sono presenti diverse soluzioni, come illustrato in [3], tuttavia però i fattori che influenzano la stima di un determinato colore da ricercare sono molteplici: il colore della pelle, per esempio, varia da persona a persona, specialmente quando si è in presenza di etnie diverse. Fattore molto importante, inoltre, è l'illuminazione: una sorgente di luce che varia continuamente modifica il calcolo della stima di colore. Ogni ambiente ha una propria illuminazione e le condizioni di luce tra interno ed esterno sono molto differenti. In aggiunta, la componente hardware di rilevamento dell'immagine varia da modello a modello, in termini di sensori adottati e sensibilità di adattamento luminoso. Questo porta ad avere diversi livelli qualitativi di immagine, a parità di uguali condizioni di luce. Lo sfondo (o background) gioca un ruolo non indifferente nella complessità della skin detection; il livello più semplice consiste nel dover individuare le porzioni di pelle su sfondi uniformi con colori contrastanti, o molto diversi, da quello da ricercare. Purtroppo però, questa è una condizione ideale che difficilmente si riesce ad avere nella realtà: molti oggetti che ci circondano, infatti, hanno colori abbastanza simili a quello della pelle; in particolar modo il legno o gli indumenti di colore rosso, che possono essere molto difficili da distinguere. Trovare un algoritmo capace di superare tutti questi problemi è impossibile; solitamente vengono introdotte sin da subito delle precondizioni di lavoro per semplificare il problema e si progetta, successivamente, un algoritmo che si adatti, il meglio possibile, ad ogni specifica applicazione. La sfida è quella di essere più generali possibile. Una considerazione che emerge, è che molti dei problemi potrebbero essere superati concentrandosi

non solo sullo spettro del visibile, ma su tecniche che sfruttano tutto lo spettro. Lo svantaggio è che viene richiesta una componentistica hardware costosa e quindi non è possibile adattarli alle numerose applicazioni.

Un altro approccio per determinare la skin detection è quello probabilistico: dato un pixel avente un colore C, la probabilità che questo pixel rappresenti la pelle è data da:

$$P\left(skin|C\right) \tag{2.1}$$

Una volta calcolata questa probabilità, il pixel viene etichettato come "pixel-pelle" se tale probabilità è maggiore di una certa soglia. Ovviamente non si può calcolare tale probabilità per ogni colore possibile (ad esempio, in 24 bit RGB, ci sono 256^3 colori). Utilizzando però la **regola di Bayes**, questa può essere riscritta come:

$$P\left(skin|C\right) = \frac{P\left(C|skin\right)P\left(skin\right)}{P\left(C|skin\right)P\left(skin\right) + P\left(C|\neg skin\right)P\left(\neg skin\right)} \tag{2.2}$$

In questo caso, la probabilità $P\left(skin|C\right)$ di avere un "pixel-pelle" dato il colore C è data dalla combinazione della probabilità $P\left(C|skin\right)$ di osservare quel colore, dato un determinato tipo di pelle, e la probabilità $P\left(skin\right)$ che quel pixel rappresenti effettivamente la pelle. Il denominatore nella regola di Bayes è la probabilità totale di osservare il colore C, un fattore che non pregiudica la decisione se un pixel debba essere etichettato come pelle o no. Data la regola di Bayes, la classificazione dei "pixel-pelle" si riduce a calcolare $P\left(C|skin\right)$. Con l'ausilio di un database, contenente esempi di pixel del colore cercato, è possibile stimare la funzione di densità di probabilità $P\left(C|skin\right)$. Diversi approcci sono stati introdotti per calcolare i classificatori, compreso l'uso di istogrammi [4], l'uso di un unico modello gaussiano, o un insieme di modelli gaussiani [6]. O ancora, dalla regola di Bayes si calcola il rapporto di probabilità di osservare una data classe di colore pelle contro una classe di colore non pelle, in altri termini, $\frac{P(C|skin)}{P(C|\neg skin)}$. Tale rapporto può essere usato come filtro per decidere se un pixel è pelle o no, utilizzando dei modelli basati su istogrammi 3D, nello spazio RGB, sulla base di un ampio database di immagini.

In un ambiente di fondo controllato, un rilevamento della pelle può essere sufficiente ad individuare la posizione della mano, come illustrato in figura 2.1.

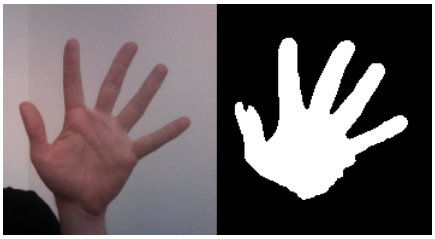


Figura 2.1: Applicazione di skin detection su ambiente controllato

Tenendo traccia delle regioni in cui viene stimata posizione della mano, è possibile ottenere un tracciato, attraverso i vari frame, del gesto eseguito.

2.2 Stimatore di moto: Optical Flow

Spesso si vuole valutare e stimare il movimento compiuto tra due fotogrammi (o una sequenza di fotogrammi), senza alcuna conoscenza a priori circa altri elementi contenuti nei frame. In generale, il moto indica che qualcosa di interessante sta accadendo sulla scena (figura 2.2). Per rilevare tale movimento, si può procedere in questo modo: in un certo frame, si individuano delle caratteristiche (come, ad esempio, bordi o angoli) in corrispondenza di determinati pixel. Una volta ottenuto il frame successivo, vengono ricercate le caratteristiche presenti nel frame precedente. Associando quindi una velocità a ciascun pixel nel frame, il movimento viene determinato dal rapporto tra la distanza in pixel dello spostamento della caratteristica in questione e il tempo intercorso tra un frame e l'altro. Tale costrutto, viene solitamente indicato come *dense optical flow* (flusso ottico denso), perchè associa una velocità a tutti i pixel dell'immagine. Una prima definizione di flusso ottico, prevede il moto apparente dei valori di luminosità in una sequenza di immagini. Formalmente, è il campo bidimensionale di velocità generato da una sequenza temporale di immagini, dovuto al moto degli oggetti presenti nella scena. La stima del movimento richiede il calcolo degli intervalli di tempo e questo rende necessari un incremento dei dati.

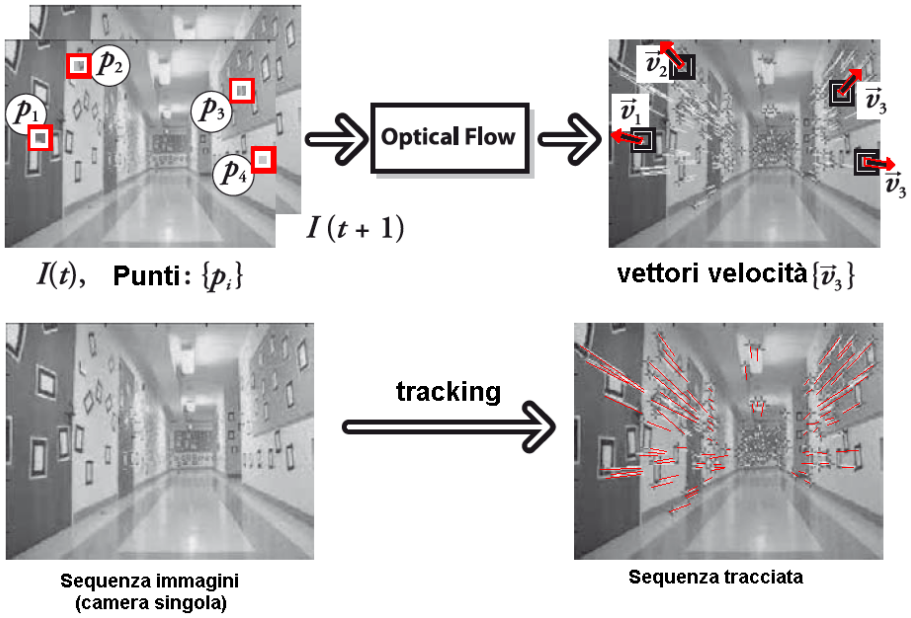


Figura 2.2: (alto-sinistra)individuati dei punti cui tenere traccia; (alto-destra) i punti sono tracciati nel tempo e il loro movimento è convertito in vettore velocità; (basso-sinistra) singola immagine di un corridoio; (basso-sinistra) sequenza tracciata con i vettori di flusso; il movimento è provocato da una camera che si sposta dal fondo del corridoio. Immagine originale di Jean-Yves Bouguet.

2.2.1 Individuazione dei punti da tracciare

Per tenere traccia del movimento, occorre prima definire dei punti chiave, ovvero punti con caratteristiche particolari che possono essere individuati durante lo scorrere dei frame. Solitamente questi punti rappresentano, ad esempio, degli angoli (corner), questo perchè presentano la stessa configurazione, in ogni inquadratura della scena osservata. Per ottenere questi punti si utilizzano algoritmi di corner detection (si individuano angoli) o edge detection (si individuano bordi); è preferibile utilizzare la rilevazione degli angoli in quanto, se si vuole analizzare il moto di un flusso video, la conformazione di un bordo produce un'ambiguità data dall'impossibilità di valutare il moto lungo la direzione parallela al bordo stesso. Questo è noto in letteratura come **problema dell'apertura** e sorge quando si ha una piccola apertura, o finestra, in cui misurare il movimento (si veda figura 2.3). Quando viene rilevato un movimento con una piccola apertura, nel caso in cui viene rilevato un bordo, non si hanno informazioni sufficienti per determinare in che direzione l'intero oggetto si sta muovendo.

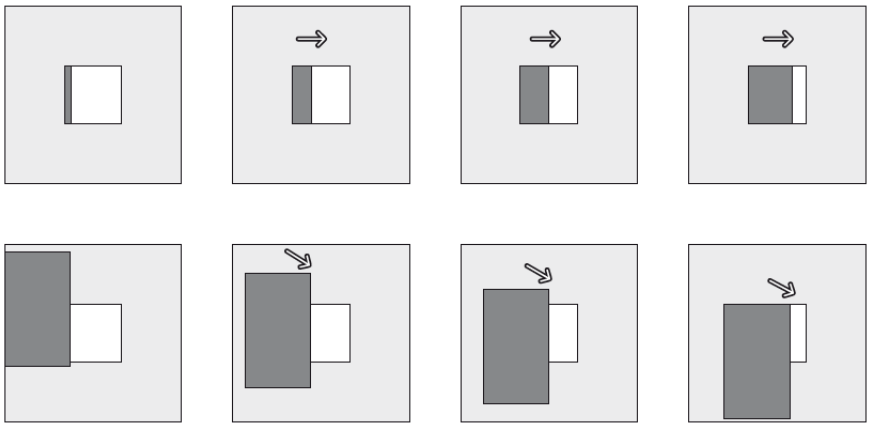


Figura 2.3: Problema di apertura della finestra: (riga sopra) si può determinare solamente un movimento del bordo proveniente da destra, mentre si perde (riga sotto) il rilevamento verso il basso del blocco grigio.

Questo non accade individuando degli angoli, ecco perchè sono preferibili filtri di corner detection.

2.2.2 Calcolo del flusso ottico

Per l'applicazione del flusso ottico devono sempre valere due ipotesi fondamentali:

1. la luminosità della scena deve rimanere costante (variazioni di luminosità potrebbero essere interpretate come uno spostamento);

2. ogni punto in esame resta in vista.

L'equazione di vincolo del flusso ottico è stata proposta per la prima volta in [7], e consiste nella descrizione matematica del moto mediante un'equazione di derivate parziali. L'equazione considera le coordinate cartesiane della sequanza di immagini (x, y), il tempo t e la brillantezza b. Ogni immagine viene modellata come funzione continua di spazio e tempo $b\,(x, y, t)$. Se un oggetto è in moto in un tempo infinitesimo δt, questo effettuerà uno spostamento lungo gli assi pari a δx e δy. Se, per ipotesi, i livelli di grigio dell'oggetto non si sono modificati, ovvero non ci sono state variazioni significative di illuminazione, allora vale che:

$$b\,(x, y, t) = b(x + \delta x, y + \delta y, t + \delta t). \tag{2.3}$$

Sviluppando in serie di Taylor, si ottiene

$$b(x + \delta x, y + \delta y, t + \delta t) = b\,(x, y, t) + \delta x \frac{\delta b}{\delta x} + y \frac{\delta b}{\delta y} + t \frac{\delta b}{\delta t} + O_s. \tag{2.4}$$

Supponendo i termini di ordine superiore O_s trascurabili, si ricava la derivata della funzione b rispetto al tempo

$$\frac{\delta b}{\delta t} = \frac{b(x + \delta x, y + \delta y, t + \delta t) - b\,(x, y, t)}{\delta t} = \frac{\delta x}{\delta t}\frac{\delta b}{\delta x} + \frac{\delta y}{\delta t}\frac{\delta b}{\delta y} + \frac{\delta b}{\delta t}. \tag{2.5}$$

Dal momento però che vale l'uguaglianza in (2.3), quest'ultima quantità deve risultare nulla, e quindi si ottiene:

$$v_x \frac{\delta b}{\delta x} + v_y \frac{\delta b}{\delta y} = -\frac{\delta b}{\delta t} \tag{2.6}$$

dove $u = \frac{\delta x}{\delta t}$ e $v = v_x \frac{\delta y}{\delta t}$. Questa equazione trovata è nota come equazione del vincolo di flusso ottico e definisce una linea, nello spazio delle velocità,

perpendicolare al gradiente locale. L'equazione sta ad indicare che la variazione di intensità di grigio in un punto, dovuta allo spostamento dalla posizione di partenza (x, y) di un certo δ, è uguale e contraria alla variazione di livello di grigio nel tempo δt del punto (x, y), cioè $\frac{\delta b}{\delta t}$. Il limite di questo metodo è costituito dal fatto che è possibile calcolare il flusso solamente nella direzione del gradiente; se l'intensità luminosa rimane costante nella direzione dello spostamento, non si notano differenze.

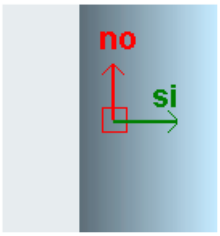


Figura 2.4: movimenti rilevabili e non rilevabili. La rotazione viene rilevata come traslazione

Non vi è una univoca soluzione per il flusso ottico, diverse tecniche si differenziano per vincoli aggiuntivi imposti.

2.2.3 Tecniche

Il calcolo del flusso ottico denso non è facile. Si consideri, per esempio, il moto di un foglio di carta bianco. Numerosi pixel bianchi in un frame, rimangono bianchi anche nel frame successivo. Solo i bordi possono cambiare, e di questi, solo quelli perpendicolari alla direzione di movimento. Risulta quindi che i metodi chiamati "*densi*", necessitano di un metodo di interpolazione tra i punti da tracciare, in modo tale da stimare anche quelli che risultano più ambigui. Tali difficoltà inferiscono in maniera significativa negli alti costi computazionali. Di fatto, si utilizzano flussi ottici "*sparsi*", cioè algoritmi che specificano in anticipo il sottoinsieme di punti che devono essere monitorati. Se questi punti hanno determinate proprietà, come "angoli", l'inseguimento sarà relativamente robusto e affidabile. Al lato pratico, il tracciamento ottico denso è relegato al solo interesse accademico.

2.2.3.1 Tecnica di Horn e Schunck

Horn e Schunck furono i primi autori a formulare l'equazione di vincolo di flusso ottico. In questo metodo si assume l'uniformità, nel tempo, del flusso ottico stimato. Questo vincolo può essere espresso mediante un integrale doppio sull'immagine

$$S = \iint \left[\left(\frac{\delta u}{\delta x}\right)^2 + \left(\frac{\delta u}{\delta y}\right)^2 + \left(\frac{\delta v}{\delta x}\right)^2 + \left(\frac{\delta v}{\delta y}\right)^2 \right] dxdy. \tag{2.7}$$

La soluzione che si ottiene dal calcolo di questo integrale, può essere non in linea con quanto imposto dall'equazione di vincolo di flusso, e il valore di tale scostamento è espresso come:

$$C = \iint \left(\frac{\delta b}{\delta x}u + \frac{\delta b}{\delta y}v + \frac{\delta b}{\delta t}\right)^2 dxdy \tag{2.8}$$

Il valore del funzionale di costo $f_c = S + \lambda C$, dove λ è un moltiplicatore di Lagrange, deve essere minimizzato: solitamente è settato ad un valore piccolo per immagini rumorose. Per minimizzare f_c si deve risolvere un sistema di due equazioni differenziali del secondo ordine per l'intera immagine

$$\begin{cases} \frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} = \lambda \left(\frac{\delta b}{\delta x}u + \frac{\delta b}{\delta y}v + \frac{\delta b}{\delta t}\right) \frac{\delta b}{\delta x} \\ \frac{\delta^2 v}{\delta x^2} + \frac{\delta^2 v}{\delta y^2} = \lambda \left(\frac{\delta b}{\delta x}u + \frac{\delta b}{\delta y}v + \frac{\delta b}{\delta t}\right) \frac{\delta b}{\delta y}. \end{cases} \tag{2.9}$$

Si possono applicare metodi iterativi per minimizzare il funzionale di ogni singolo pixel. Questo metodo per due sole immagini è molto costoso, in termini computazionali, mentre per una lunga sequenza di immagini l'iterazione può essere eseguita una sola volta per coppia e si considera il risultato dell'iterazione precedente come stima iniziale. Come dimostrato da [8], la stima del flusso ottico con questo metodo è molto sensibile alla presenza di rumore sull'immagine; inoltre, quando lo spostamento tra due immagini è grande, la soluzione ottenuta minimizzando f_c perde di precisione. Vi è richiesto inoltre, anche un calcolo molto accurato delle derivate parziali effettuato con metodi numerici, cosa che fa aumentare, in maniera considerevole, il costo computazionale di ogni iterazione.

2.2.3.2 Tecnica di Lucas e Kanade

Le precondizioni della tecnica di Horn e Schunck circa l'uniformità del flusso ottico, portano un costo computazionale elevato perchè è richiesto lo svolgimento di due equazioni differenziali del secondo ordine per ogni pixel dell'intera immagine, e tutto questo, ad ogni iterazione. La precondizione invece di Lucas e Kanade è la seguente: si suppone che tutti i gruppi di pixel adiacenti nell'immagine abbiano tutti la stessa velocità durante un certo spostamento; usando quindi l'equazione di flusso ottico per questi gruppi, si riesce a trasformare il problema, da un sistema di equazioni differenziali, ad un semplice problema lineare. La soluzione è ricercata sfruttando la proprietà di località lineare del sistema di equazioni differenziali. Se il sistema non è singolare, la soluzione esiste ed è unica anche per gruppi di due soli pixel. Indichiamo con $W(x, y)$ la finestra gaussiana usata per raggruppare i pixel e con b_x, b_y e b_t la funzione di brillantezza definita precedentemente. Si considera il seguente sistema lineare:

$$\begin{cases} \sum_{x,y} W(x,y)\, b_x b_y u + \sum_{x,y} W(x,y)\, b_y^2 v = -\sum_{x,y} W(x,y)\, b_y b_t \\ \sum_{x,y} W(x,y)\, b_x^2 u + \sum_{x,y} W(x,y)\, b_x b_y v = -\sum_{x,y} W(x,y)\, b_x b_t. \end{cases} \tag{2.10}$$

Iterando l'algoritmo si migliora l'accuratezza del sistema perchè lo scostamento ricavato dall'iterazione precedente viene ora usato per stimare una nuova posizione della finestra di ricerca nella seconda immagine, da cui viene sottratta la finestra della prima. In presenza di rumore questo metodo è robusto e computazionalmente efficiente, presenta però lo svantaggio di non essere applicato nei punti in cui il gradiente si annulla.

2.2.3.3 Tecnica di Lucas-Kanade immagine piramidale

Prima di affrontare la tecnica, introduciamo brevemente il concetto di immagine piramidale.

Immagine Piramidale

Si tratta della rappresentazione di una immagine avente differenti livelli di risoluzione.

Figura 2.5: immagine con risoluzione piramidale

Vi è una raccolta di immagini, tutte derivanti da una singola immagine di origine, che vengono successivamente ricampionate fino ad un punto desiderato [9]; ovviamente un punto di arresto potrebbe essere anche un singolo pixel dell'immagine. Esistono due tipologie di piramidi in letteratura: Gaussiana [10] e Laplaciana [11]. La piramidazione Gaussiana viene utilizzata per eseguire il campionamento verso il basso delle immagini, mentre la piramidazione laplaciana è necessaria quando si vuole ricostruire una immagine con un campionamento verso l'alto. Per produrre il livello $(i+1)$ nella piramide gaussiana (si denota questo livello con $Level_{i+1}$) partendo dal livello $Level_i$ della piramide, si effettua una convoluzione di $Level_i$ con filtro gaussiano e, successivamente, vengono rimosse le righe e le colonne numerate pari. Da questo segue immediatamente che, ogni immagine, è esattamente un quarto della superficie di quella precedente. Iterando questo processo a partire dall'immagine in ingresso[4], si produce l'intera piramide. Ogni elemento di un livello della piramide, a esclusione della base, riceve l'input dai suoi elementi figli.

Le piramidi gaussiane e laplaciane sono illustrate schematicamente in figura 2.6. Viene mostrato inoltre, il processo inverso per recuperare l'immagine originale a partire da sotto-immagini.

[4]Livello 0 e risoluzione più alta

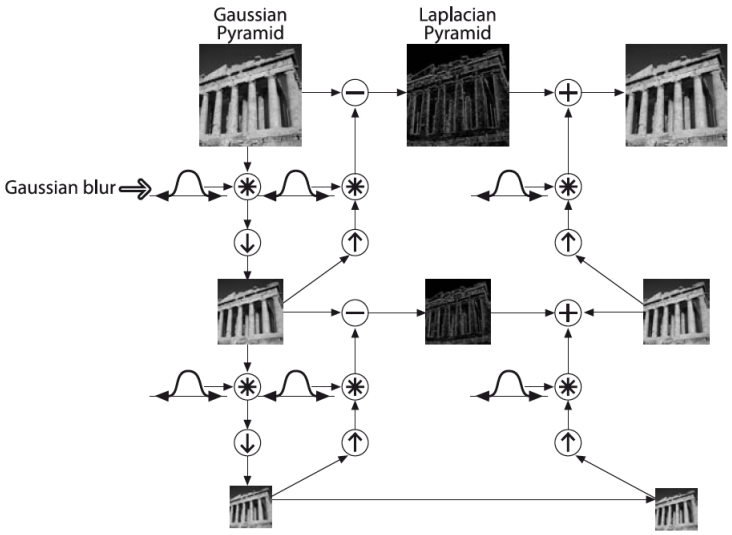


Figura 2.6: Piramide gaussiana e la sua inversa laplaciana

Algoritmo

L'idea, come è descritta in [17], è basata su tre assunzioni:

1. **La luminosità deve essere costante**: i pixel che si riferiscono a oggetti all'interno dell'immagine non cambiano aspetto nel movimento tra frame successivi. Per immagini in scala di grigio questo significa ipotizzare che la luminosità di un pixel non cambia tra i vari frame;

2. **Persistenza temporale o piccoli movimenti**: si considera che il movimento di una porzione di immagine, tra frame successivi, sia sufficientemente lento. In altri termini, gli incrementi temporali sono sufficientemente rapidi da poter supporre che il movimento di un oggetto sia piccolo tra un frame e l'altro;

3. **Coerenza spaziale**: i punti vicini nell'immagine appartengono alla stessa superficie, hanno movimenti simili, ed hanno in generale un comportamento "rigido".

Il flusso ottico è inizialmente calcolato al livello più alto della piramide, ossia $Level_n$. Ottenuta questa stima, la si passa in input al livello piramidale sottostante; questo calcolo viene interpretato come ipotesi sul probabile cambiamento

di posizione riguardo il pixel. Partendo da questi primi passi, il flusso ottico si raffina e il risultato viene propagato verso il basso. Questa operazione di raffinamento viene eseguita finchè non si raggiunge il livello piramidale più basso, $Level_0$, che è l'immagine originale, e la stima qui calcolata sarà quella effettivamente cercata. Ad ogni livello piramidale, la stima viene calcolata con l'algoritmo di Lucas-Kanade. Il vincolo di avere un campo di velocità piccolo e uguale tra gruppi di pixel vicini, è sicuramente valido a livello più alto della piramide; avendo una buona stima iniziale nei livelli più bassi, il flusso ottico converge velocemente al valore che minimizza il valore di posizionamento dei pixel nell'immagine di origine.

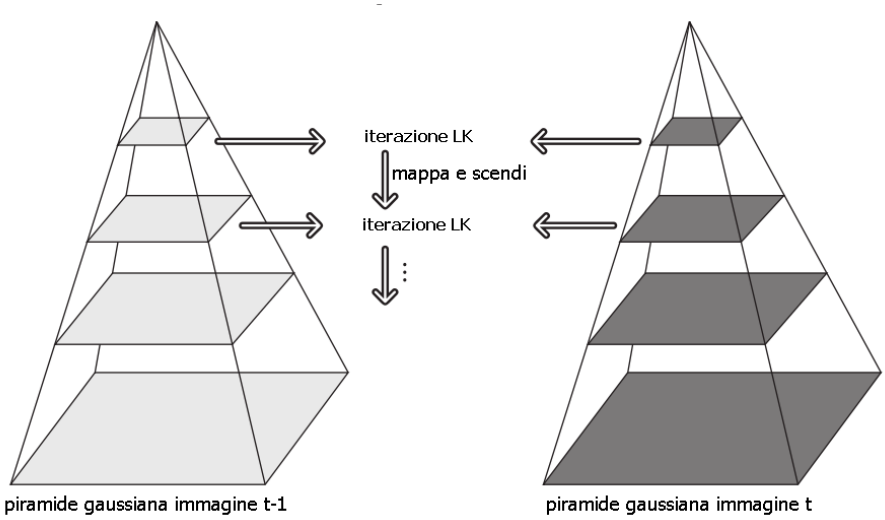


Figura 2.7: Procedimento di stima del flusso ottico

2.3 Mean-shift e Camshift

Mean-shift e Camshift sono delle metodologie che prendono in considerazione una regione dell'immagine in cui si trova l'oggetto da tracciare, basandosi ad esempio, sull'istogramma del colore, in quanto l'istogramma del colore può essere una buona rappresentazione statistica dell'aspetto dell'oggetto. Si definisce così una descrizione globale dell'oggetto da tracciare, senza fornire informazioni spaziali. Questo ovviamente comporta il fatto che se l'oggetto da tracciare ha una statistica simile, a livello di istogramma, a una regione dello sfondo, il tracking potrebbe fallire. In aggiunta, vi sono delle difficoltà nel tracciamento dovute a cambi dell'aspetto, come per esempio un uomo che toglie gli occhiali da sole, oppure un cambio di illuminazione dell'ambiente. Un aspetto positivo è che questo tipo di tracking è molto robusto alle deformazioni dell'oggetto da tracciare, dal momento che le deformazioni non modicano la statistica. Lavori con l'uso di istogramma basato sul colore, sono stati svolti da [12]: vengono confrontate tutte le regioni dell'immagine con l'istogramma rappresentante l'aspetto dell'oggetto e vengono selezionate le regioni più simili al modello. Il risultato di questo tracking è una distribuzione di probabilità che indica le posizioni più probabili assunte dall'oggetto.

Vediamo ora nel dettaglio le due differenti metodologie.

Mean-Shift

Il Mean-shift è un metodo che consiste nel trovare estremi locali della densità di distribuzione in un insieme di dati [29]. La definizione formale la si può trovare in [13]. L'algoritmo è composto di quattro parti:

1. Scegliere una finestra di ricerca e delle relative caratteristiche:
   (a) posizione iniziale;
   (b) tipo (uniforme, polinomiale, esponenziale, o gaussiana);
   (c) forma (simmetrica o asimmetrica, eventualmente ruotata, arrotondata o rettangolare);
   (d) dimensioni (misura in cui esso si esce o si interrompe);

2. Calcolare il centro di massa pesato della finestra;

3. Centrare la finestra con il centro di massa;

4. Ritornare al passo 2., finchè non ci si ferma. Ovviamente le iterazioni sono in genere limitate ad un numero massimo che garantisca la convergenza;

Formalmente, questo algoritmo è legato alla disciplina della stima di densità di kernel, dove per *kernel* ci si riferisce ad un filtro spaziale, detto anche *maschera*, che è una regione rettangolare caratterizzata da un'operazione predefinita. L'operazione viene eseguita sui pixel dell'immagine corrispondenti alla suddetta regione, e le modifiche indotte dal filtro, generano la cosiddetta immagine filtrata. Il filtro opera sovrapponendosi all'immagine, partendo generalmente dall'angolo in alto a sinistra. Quando il punto centrale (detto anchor point) coincide con il pixel da modificare, quest'ultimo verrà trasformato nel nuovo pixel dell'immagine di output. Il kernel si sposta da sinistra verso destra, e procede in questo modo fino a termine riga, per poi ripartire dalla riga successiva modificando i valori di input nel modo indicato. Il procedimento si ripete in questo modo per tutti i punti dell'immagine. L'algoritmo Mean-shift diverge dalla stima di densità data dal kernel perchè questo cerca solo di stimare il gradiente (direzione della variazione) della distribuzione dei dati. Quando questo cambiamento assume il valore pari a 0, si è in una situazione stabile (anche se probabilmente locale) di picco della distribuzione. Ci potrebbero essere, comunque, altri picchi vicini o in altre scale. La figura 2.8 mostra le equazioni utilizzate da mean-shift. Queste equazioni possono essere semplificate tenendo conto di un kernel rettangolare, che riduce il calcolo del centro di massa della distribuzione dei pixel.

$$X_c = \frac{M_{10}}{M_{00}}, Y_c = \frac{M_{01}}{M_{00}}. \tag{2.11}$$

Il momento zero è calcolato come $M_{00} = \sum_x \sum_y I(x, y)$ mentre i momenti primi sono $M_{10} = \sum_x \sum_y xI(x, y)$ e $M_{01} = \sum_x \sum_y yI(x, y)$.

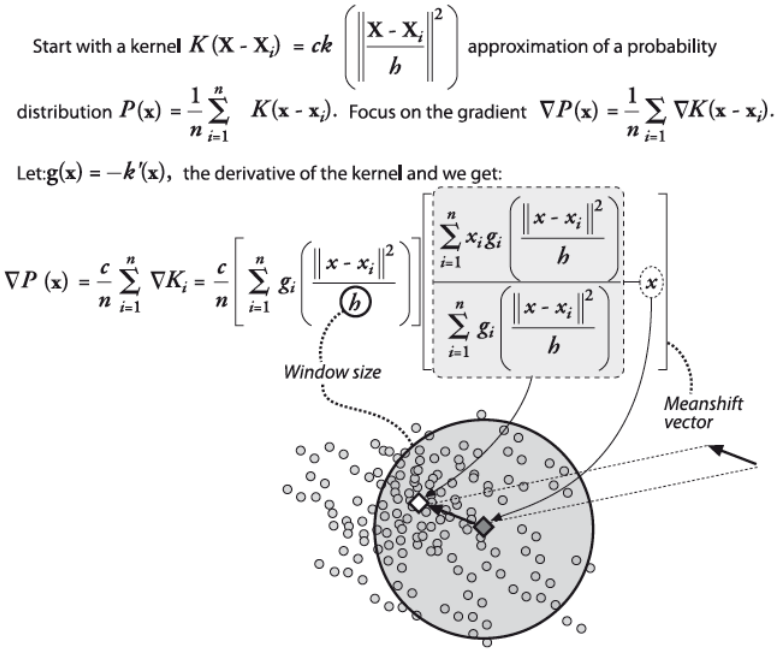


Figura 2.8: equazioni di mean-shift e loro rappresentazione, tratti da [14]

In questo caso si inquadra la finestra con il centro di massa calcolato all'interno della finestra stessa. Questo movimento va a modificare ciò che è al di sotto della finestra e quindi è necessario iterare questo processo di ricentramento. La posizione di convergenza è ad un massimo locale (picco locale) della distribuzione nella finestra. Diversificando le dimensioni della finestra si trovano diversi picchi su cui decidere.

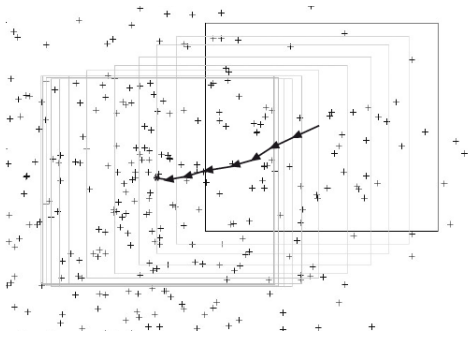


Figura 2.9: partendo da una finestra iniziale in una matrice bidimensionale, vi è una modalità di ricentramento sul picco locale della distribuzione data, fino a convergere

Nella figura 2.9, vi è un esempio di distribuzione bidimensionale di dati e una prima finestra rettangolare. Le frecce indicano il processo di convergenza su un picco locale nella distribuzione. Da osservare che questo picco è statisticamente solido, cioè i punti fuori dalla finestra di mean-shift, non sono affetti da convergenza[5].

Camshift (continuously adaptive mean-shift)

Questo algoritmo differisce dal Mean-shift perchè la dimensione della finestra di ricerca si autoregola. Se si dispone di distribuzioni ben segmentate (come ad esempio tratti del viso che rimangono compatti), allora questo algoritmo si regolerà automaticamente per la dimensione della faccia della persona, sia che questa si avvicini oppure si allontani dalla camera. In questo modo l'algoritmo cerca di anticipare il movimento dell'oggetto per rintracciarlo, il più rapidamente possibile, nella scena successiva. Questo algoritmo è in grado di monitorare correttamente anche movimenti rapidi.

Camshift traccia la tonalità di un oggetto. L'algoritmo è implementato in cinque passi:

1. Si calcola la posizione iniziale della finestra di ricerca 2D;

2. La distribuzione di probabilità del colore è calcolata per una regione leggermente più grande di quella della finestra di ricerca, utilizzando mean-shift;

3. Mean-shift è calcolato sull'area, fino al raggiungimento di una convergenza. I momenti zero e le coordinate del centroide sono calcolati e memorizzati;

4. La finestra di ricerca del successivo frame è centrata attorno al centroide e la dimensione è scalata da una funzione di movimento zero;

5. Ritorna allo step 2;

[5]in pratica l'algoritmo non è condizionato da punti lontani

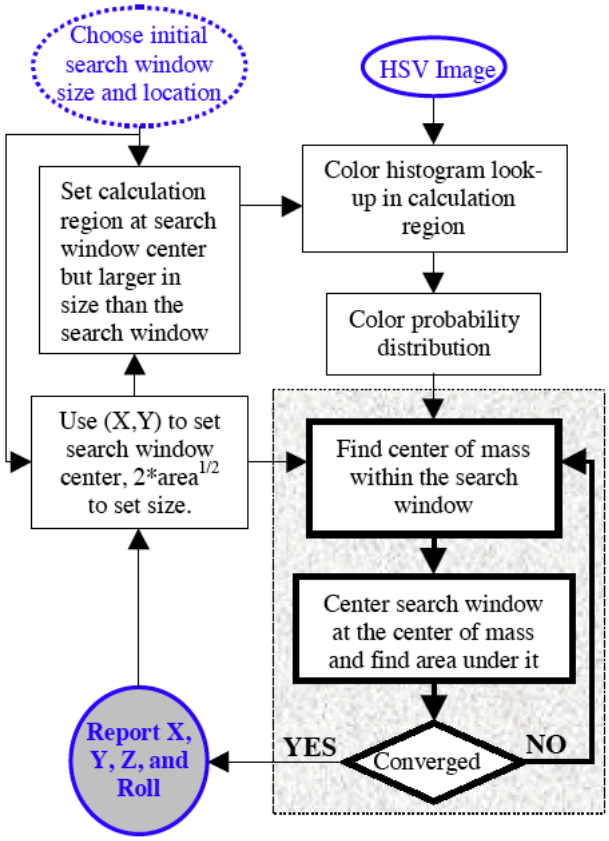


Figura 2.10: diagramma a blocchi dell'algoritmo di camshift per il tracciamento di un colore [15]

2.4 Riconoscimento di posizione: algoritmo per microsoft Kinect

L'algoritmo di Kinect [21] è in grado di prevedere in maniera rapida le posizioni assunte dal corpo nello spazio tramite l'utilizzo di una immagine di profondità, attraverso una fase di training e di stima delle posizioni, basata su etichette. In [21] viene associato il problema di stima della postura, con un problema di classificazione di pixel che rappresentano la profondità, ricorrendo all'utilizzo di una "depth cam".

Nell'ultimo ventennio si sono sperimentati vari modi per stimare la posa di una

persona attraverso una singola camera [27, 28], anche attraverso modelli stocastici combinati con particle filter[6] [31]e standard Condensation[7] [22, 32] e l'utilizzo di regressione lineare su vettori di descrizione della forma, estratti automaticamente da sagome di immagini [23, 30]. Il lavoro di stimare la terza dimensione è stato recentemente molto semplificato con l'introduzione di telecamere real-time di profondità [24, 25, 26]. A partire da una immagine di profondità data in input, tramite un sistema probabilistico, viene effettuata una etichettatura delle parti del corpo, spazialmente localizzate nei pressi della posizione della persona. Valutare la segmentazione in parti del corpo, tramite l'uso di classificazione che considera i pixel, evita una ricerca combinatoria sulle differenti articolazioni del corpo. Per i dati di training, sono state prese immagini di profondità realistiche di persone ritratte in differenti pose[8] che, successivamente, vengono campionate per essere utilizzate da alberi decisionali in fase di apprendimento dell'algoritmo. Il corpo umano è in grado di produrre innumerevoli posizioni, in questo approccio si è deciso di catturare molte azioni umane in un database. Utilizzando un processo di rendering randomizzato, si etichettano le immagini che serviranno da campioni per l'addestramento dell'algoritmo, come mostrato in figura (2.11):

Figura 2.11: processo di renderizzazione e stima delle parti del corpo

Vengono marcate diverse parti del corpo cromaticamente, alcune di queste

[6]Filtro noto come metodo Monte Carlo: è una tecnica di stima basata su modelli di simulazione. Simili a un modello di Markov nascosto (HMM), ma in generale lo spazio degli stati è continuo anzichè discreto, e non sufficientemente ristretto a rendere l'inferenza trattabile, come ad esempio, in un sistema lineare dinamico, dove lo spazio degli stati è limitato a distribuzioni gaussiane e di conseguenza l'inferenza può essere fatta in modo efficiente usando un filtro di Kalman.

[7]L'applicazione principale è quella di rilevare e monitorare il contorno di oggetti in movimento in un ambiente disordinato; è un algoritmo probabilistico che tenta di risolvere il problema identificare quali pixel in un'immagine costituiscono il contorno di un oggetto.

[8]al momento le posture sono per lo più in piedi o al massimo da seduti.

definiscono giunture scheletriche di interesse primario, mentre altre sono utilizzate per riempire i vuoti, oppure per prevedere altre articolazioni, magari occluse in una particolare posa. A questo punto, etichettate le varie parti del corpo di interesse, si eseguono degli algoritmi di classificazione. Una considerazione da fare è la seguente: le parti etichettate devono essere sufficientemente piccole, in modo tale da fornire una accurata localizzazione, però allo stesso tempo non devono essere troppo numerose per non limitare e compromettere una buona classificazione.

La classificazione sui pixel per determinare la postura, avviene in questo modo: dato un certo pixel x, si è sviluppato un confrontatore di profondità

$$f_\theta(I, x) = d_I\left(x + \frac{u}{d_I(x)}\right) - d_I\left(x + \frac{v}{d_I(x)}\right), \tag{2.12}$$

dove $d_I(x)$è la profondità del pixel x nell'immagine I e il parametro $\theta = (u, v)$ descrive l'offset tra u e v; $\frac{1}{d_I(x)}$ è una normalizzazione che ha il compito di assicurare che le caratteristiche di profondità rimangano invariante. Con questa formula si cerca di stimare, localmente, dove si trovano, o sono posizionate, le articolazioni inerenti ad un'area etichettata precedentemente come giuntura, piuttosto che lo sfondo. Nell'immagine 2.12, nella parte (a) le due stime ritornano una grande differenza, segno della distinzione tra punti dello sfondo e punti del corpo, mentre in (b) si hanno risposte simili, quindi i due punti gialli sotto esame, possono essere compresi come parte del corpo.

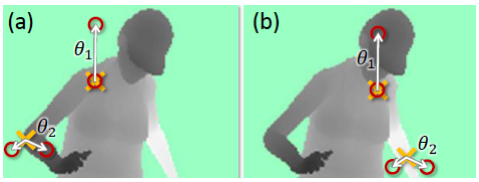


Figura 2.12: stima sulla profondità per individuare le parti del corpo

Dal momento che il classificatore non utilizza informazioni temporali, questo sistema analizza posizioni statiche e non di movimento. Vi è una precondizione secondo cui, il cambiamento di posa si verifica se viene rilevato un cambiamento superiore a 5 cm tra un frame ed il suo successivo. Una volta che il classificatore ha stimato una posa, questa non cambierà se il movimento sarà inferiore a 5 cm.

Verranno quindi scartate le posizioni definite "molto simili" e troppo ravvicinate. Altro vincolo da tenere in considerazione è che per motivi tecnici dovuti alla camera di profondità, ci si deve portare, per un corretto uso, ad una distanza di minimo due metri dal sensore.

2.5 Valutazioni e considerazioni

Certamente tutti i metodi, elencati precedentemente, hanno un loro ben specifico ambito di applicazione e sono tra loro molto diversi, sia come approccio al problema, sia come metodologia risolutiva applicata. Molti di questi, sono utilizzati anche congiuntamente, al fine di strutturare applicazioni, che sfruttano al meglio l'uso di ogni singola tecnologia. In questa tesi, per ovvi motivi, sono state definite solamente delle panoramiche introduttive per parlare degli aspetti positivi ma anche dei vincoli e limitazioni che ogni approccio possiede, se contestualizzato in un ambito diverso da quello per cui è stato creato. L'utilizzo di un tracciamento basato su skin detection può essere molto utile in ambito di video sorveglianza oppure quando il goal dell'applicazione è determinare il numero di persone in una stanza, piuttosto che la presenza di parti del corpo all'interno della scena. Quando però lo scopo dell'applicazione è seguire solamente delle aree limitatamente piccole, rispetto al corpo, la skin detection mostra un limite dovuto al fatto di non riuscire a discriminare le varie articolazioni, in quanto il filtro si limita a definire i blob[9] e non è in grado, da solo, di stabilire che parti del corpo si stanno prendendo in considerazione. L'illuminazione gioca un ruolo molto importante e condiziona, in maniera sostanziale, la resa e l'accuratezza della stima nel tracciamento di un determinato colore. La presenza di particolari indumenti sulla scena, inoltre, può compromettere, anche in modo considerevole la buona riuscita di un riconoscimento: indumenti rossi come felpe, o sciarpe, per esempio possono essere scambiati per pelle umana da un algoritmo di skin detection. A questo punto, se si limita troppo l'intervallo di tolleranza, non si riescono a catturare le varie tinte della pelle (persone di colore o altre etnie) con la conseguenza di far fallire l'algoritmo; se invece, l'intervallo diviene troppo grande, ovvero trop-

[9] aree continue con determinate proprietà

po tollerante, si rilevano frequentemente falsi positivi. In generale, l'utilizzo di skin detection, è appropriato quando si deve rilevare la presenza di persone, o grandi aree, all'interno della scena, ma risulta non appetibile quando si vuole suddividere, discriminare e identificare le sotto parti di aree rilevate.

Soluzioni che adottano l'optical flow sono particolarmente adatte alla rilevazione di movimento sulla scena. Un ambito molto diffuso in cui è usato, è la video sorveglianza. Anche in questo caso, il focus è più rivolto al rilevamento di movimento, che non al discriminare un movimento. Ci sono tuttavia implementazioni che utilizzano questo metodo come tracciamento dei movimenti facciali [33]. C'è da dire che, se si vuole individuare un movimento specifico, l'optical flow si comporta bene se non ci sono, all'interno della scena, altri movimenti che creano rumore e distorcono la stima del movimento. Nel caso del riconoscimento dei gesti facciali, se la telecamera è puntata di fronte al viso dell'utente e non inquadra molto altro, allora il rilevamento di movimento avviene in maniera abbastanza chiara; se invece la telecamera è posizionata lontano dall'utente (per esempio inquadra tutta la persona) allora il riconoscimento dei gesti del volto può essere distorto dal movimento del corpo. Vi è anche da considerare il caso in cui l'area sensibile al movimento non è tutta l'area inquadrata, ma solamente una piccola parte, quella che in termini tecnici viene definita ROI[10]; in questa situazione però ci possono essere varie situazioni: o la ROI si sposta con lo spostamento della persona, oppure la ROI rimane fissa. Con una ROI fissa, si vincola l'utente a stare obbligatoriamente in quella posizione, mentre con una dinamica si dà maggiore libertà di movimento. Un punto però a sfavore, che vincola le implementazioni con ROI in movimento, riguarda la strutturazione e costruzione dell'optical flow: la distanza di un particolare corner di tracciamento, tra un frame ed il suo successivo, deve essere ridotta. Se l'applicazione supporta movimenti lenti o moderatamente lenti, allora ha senso applicare questo metodo, altrimenti su applicazioni che richiedono di riconoscere movimenti veloci, non è consigliabile adottare questo metodo, in quanto si rischia molto spesso di perdere il corner desiderato. A questo proposito si era provato, all'inizio della tesi ad implemetare una soluzione di tracciamento della mano, con telecamera che inquadrava la persona per intero, utilizzando ROI dinamiche. I risultati hanno evidenziato quanto

[10]Region Of Interest - Regione di Interesse

affermato in precedenza: con movimenti veloci molto spesso la mano usciva dalla ROI, perchè quest'ultima non era in grado di calcolare il movimento fatto al suo interno e stimare un nuovo centro di ROI, dove poter riposizionare l'area sensibile. Non c'erano sufficienti frame per determinare lo spostamento dei corner presi come riferimento. Anche qui va detto che se viene scelta una dimensione di ROI troppo piccola, non si riescono a determinare molti movimenti, viceversa, ingrandendo troppo l'area, si rischia di introdurre rumore dovuto a movimenti involontari e secondari, rispetto a quello che si sta eseguendo; inoltre la dimensione della regione di interesse dipende anche dalla distanza dalla telecamera di cattura dell'immagine. Vista la debole stima di tracciamento si è scelto di sviluppare una soluzione più robusta e flessibile, che è l'oggetto di discussione di questa tesi.

La metodologia di Mean-shift e Camshift si basa anch'essa sui livelli di tonalità, saturazione e tinta e, come tale, è soggetta alle limitazioni e vincoli che caratterizzano il riconoscimento basato sul colore. Un approccio sperimentato, è stato quello di tracciare una mano in movimento davanti al corpo di un utente, il quale indossava un indumento di colore rosso. Per come è strutturato Camshift standard, cioè senza considerare medotologie ibride che aggiungono informazioni e tecniche statistiche per discriminare i movimenti effettuati dai pixel rilevati come sensibili, è risultato che l'area selezionata era limitata alla mano. Successivamente, quando il movimento iniziava, e la mano si sovrapponeva all'indumento colorato, l'area individuata da camshift si allargava troppo facendo perdere così la precisione richiesta per il tracciamento. La soluzione era quindi di ricalibrare di nuovo l'analisi della stima del colore da tenere in considerazione.

Figura 2.13: (sinistra) camshift appena settato; (destra) camshift ottenuto con movimento mano davanti al busto. Il colore dell'indumento crea disturbi di stima e si perde di precisione nel determinare la mano durante l'esecuzione del movimento

Un passo avanti notevole, rivolto al mondo videoludico, è stato fatto costruendo dispositivi via via sempre più sofisticati e non invasivi per dare al giocatore l'illusione di interagire in prima persona con il gioco. Partendo da Wii[11] e passando da PlayStation Move[12], si è arrivati ad una dimensione di gioco libera da controller fisici. Microsoft Kinect[13], di cui sopra sono state descritte delle peculiarità, rappresenta un dispositivo sia hardware (camera di profondità e sensori audio) che software (machine learning, streaming di dati catturati dall'hardware che sono filtrati dal middleware e forniti al programmatori sottoforma di API[14]). Le pose assunte da un giocatore danno l'impressione all'utente di essere lui stesso il controller. Dall'uscita della prima versione sono stati fatti numerosissimi "hack"[15] che spaziano dalla gestione di media center casalinghi, per la gestione di film o musica, allo sviluppo di dispositivi in campo medico, come un visualizzatore e gestore di lastre ai raggi X ospedalieri che, senza toccare nessun dispositivo fisico, permettono al dottore di analizzarle anche se questo ha le mani occupate. Un altro tipo di hack è stato compiuto nel campo della robotica, sino ad arrivare alle performance artistiche e museali. Tutte queste operazioni hanno in comune il fatto che, chi interagisce con il dispositivo, deve eseguire movimenti "ampi", in modo tale da consentire al classificatore di pose, di stimare la posizione più adatta. Lo scopo di questa tesi è identificare movimenti che vengono compiuti in uno spazio ristretto, indipendentemente dal fatto che siano veloci o lenti, permettendo inoltre alla telecamera di cattura dei frame di essere abbastanza vicina da riprendere tutti i "piccoli gesti" eseguiti. L'approccio kinect non riesce al momento a soddisfare questi requisiti, per motivi implementativi: non è possibile stimare pose che si discostano tra di loro meno di 5 centimetri di distanza; inoltre la distanza minima che deve essere rispettata tra la posizione del rilevatore e l'utente è di almeno due metri[16].

Quello che si vuole realizzare è un approccio nuovo in grado di catturare movimenti fini su spazio ristretto, nel nostro caso davanti al corpo dell'utente,

[11] http://www.nintendo.it/NOE/it_IT/
[12] http://it.playstation.com/psmove/
[13] http://www.xbox.com/it-IT/kinect
[14] con rilascio di apposito SDK ufficiale e vincoli legali di utilizzo.
[15] termine usato per indicare personalizzazioni e usi al di fuori del contesto videoludico
[16] Nelle specifiche ufficiali è stimata a 1,80 cm però questo è un limite inferiore un po' basso perchè considera situazioni di illuminazione piena.

adottando tecniche non invasive senza indumenti e marcatori particolari, e permettendo, a chi deve cattuare determinati gesti, di poter scegliere se posizionare la telecamera nelle strette vicinanze[17] dell'esecutore, oppure di lasciargli più libertà e posizionarla più lontano[18].

La base di partenza di questo lavoro di tesi si ispira a [1, 34]. Sin da subito ci si è messi nell'ottica di realizzare un sistema di tracciamento dei gesti che non fosse costoso a livello hardware, e che non fosse invasivo. In [1] si è posto il problema di identificare i gesti di una sfoglina nell'arte della preparazione dei tortellini. L'obiettivo che poi ha ispirato il lavoro di tesi è quello di cercare un processo di apprendimento nella preparazione di oggetti ricercati, sviluppati da persone altamente qualificate, che si avvalga della tecnologia, in favore di tutte quelle persone che vogliono imparare i movimenti e i gesti di abili costruttori. L'interesse è totalmente rivolto alla codifica di piccoli gesti e movimenti, generalmente eseguiti nelle strette vicinanze del corpo, con spostamenti ridotti e precisi, cercando di catturare quel movimento "puro" nell'esecuzione di un gesto. Chiaramente, visto i problemi tecnici emersi dall'analisi di metodologie esistenti, lo scopo è quello di proporre un approccio diverso, in grado di fornire un'alternativa nuova riguardo la codifica dei gesti. La scelta di non adottare l'uso di marcatori è giustificata dal fatto che, trattando movimenti fini, l'esecutore può riscontrare difficoltà o disturbi nell'eseguire un lavoro con addosso guanti o indumenti invasivi e ingombranti.

[17] Intorno ai 50 centimetri.
[18] Intorno a 1,5 metri

Capitolo 3

Architettura e esempio di utilizzo dell'applicazione

Il sistema sviluppato in questa tesi è rivolto al riconoscimento di gesti fini, ovvero gesti che richiedono piccoli movimenti compiuti all'interno di un'area molto ristretta, come possono essere, per esempio, i movimenti effettuati dalle mani davanti al corpo. Nello specifico, la grana fine del sistema è in grado di codificare il movimento o il gesto effettuato da una singola mano, oppure da entrambe. Il processo di riconoscimento si compone di due parti fondamentali:

1. tracciamento del movimento o dei movimenti effettuati;

2. codifica del gesto eseguito.

In particolare, con questa metodologia, si è in grado di stimare con precisione il numero di ripetizioni di una data azione, indipendentemente dal fatto che questa sia eseguita lentamente o velocemente; inoltre, viene registrata la sequenza di posizioni che sono state già raggiunte da una mano, o da tutte e due, all'interno di una data azione. Tutto questo può essere utilizzato in diversi ambiti, descritti nella sezione 3.2, dove vengono riportati esempi di utilizzo e casi d'uso. La distinzione di gesti effettuati con una singola mano oppure con entrambe è affidata al sistema, quindi non c'è bisogno di impostare o settare nulla a livello

di inizializzazione. Tutto quello che si deve fare è posizionare una telecamera, oppure una semplicissima webcam[1], e avviare l'applicazione.

3.1 Descrizione sull'utilizzo dell'applicazione

Le metodologie prese in esame nel capitolo precedente, hanno fatto emergere aspetti positivi e negativi nell'uso di questa o quella applicazione. A questo punto è bene rapportare il sistema creato agli altri approcci citati, in modo tale da collocare anche questa nuova realizzazione, in un ambito con più ampio respiro.

Sensibilità alla luce

Trattandosi di un dispositivo che utilizza una camera per catturare i vari frame, la sensibilità è determinata solamente dal dispositivo fisico utilizzato. Adottando apparecchi di fascia medio alta, si avrà una sensibilità maggiore alla luce, e sarà quindi necessario settare vari parametri per ottenere una resa ottimale. Nel caso si opti per una fascia low-cost, è risaputo che queste sono un po' più "robuste", nel senso che non avendo molte funzioni o settaggi particolari, montano dei filtri standard che tengono abbastanza bene il refresh dell'immagine, a discapito però di una riduzione, anche significativa, della resa e qualità dei frame.

Sensibilità al colore

A causa della sensibilità al colore, viene tenuto in considerazione il fatto che si possano subire alterazioni nella stima della posizione delle mani, dovute alle scelte algoritmiche che prendono in considerazioni modelli di rappresentazione digitale del colore. Il sistema, sotto questo aspetto, non considera filtri di skin detection o camshift, e l'utente non è vincolato a scegliere un particolare vestiario.

[1]Non sono richieste particolari proprietà, dai test eseguiti è risultata una stima molto buona anche con l'utilizzo di webcam low cost.

Utilizzo di algoritmi per il flusso ottico

Non viene considerato il calcolo del flusso ottico dal momento che la camera, può essere sia vicina che lontana dall'utente; questo può introdurre rumore o stime non corrette che portano a deviare il calcolo del movimento fatto. Trattandosi di movimenti che possono essere sia veloci che lenti, un approccio con ROI dinamiche non è stato preso in considerazione perchè se il movimento risulta troppo veloce, l'algoritmo di optical flow perde i corner di riferimento e si otterrebbe un tracciamento molto grossolano se non addirittura erroneo. La versione del sistema, precedente a quello proposto della tesi, implementava questa metodologia, ma da test effettuati non soddisfava le specifiche di rendimento che ci si proponeva di soddisfare.

Rilevamento di piccoli movimenti

Il metodo da noi proposto è in grado di rilevare movimenti anche molto piccoli della mano, dell'ordine di 1 centimetro. Come abbiamo già visto, questo non è possibile con kinect, non solo perchè l'algoritmo non è in grado di stimare movimenti al di sotto di uno scostamento di 5 centimetri, ma anche perchè, per come è al momento sviluppato, riconosce solamente snodi e giunture grandi del corpo.

Posizionamento camera

Il posizionamento della camera può variare da circa 45 cm in su, sino ad arrivare a 2m circa[2]. Non ci sono vincoli particolari sulla distanza tra utente e camera. Va solamente detto che essere troppo vicini, circa 15/20 cm non ha molto senso perchè, da quella vicinanza, non si riescono a catturare molte tipologie di gesti delle mani, in quanto escono subito dall'ottica del dispositivo.

[2]Aumetando la distanza si perdono molti passaggi e dettagli dei movimenti fini.

3.2 Applicazione del sistema nel contesto reale

Prendendo spunto da [1], si è pensato che il mondo artigiano potesse essere interessato alla sperimentazione e l'utilizzo di questo nuovo approccio. Di seguito viene riportato il contesto applicativo nel quale il riconoscimento gesti può essere molto utile.

3.2.1 Il contesto artigianale "Made in Italy"

L'artigianato italiano è frutto di quel connubio tra cultura umanistica e cultura tecnologica che è conosciuto in tutto il mondo come "Made in Italy". Al giorno d'oggi il metodo di apprendimento "maestro-apprendista" non è sempre facile da trovare; ecco che preservare e codificare le abilità artigianali diventa veramente importante per non perdere il patrimonio storico e artistico che sin dal passato si è tramandato all'interno delle imprese artigiane. Una applicazione interessante dell'approccio proposto in questa tesi è la codifica della conoscenza manuale e artigianale. Il sapere artigiano, come nell'idea del modello Bauhaus, è frutto di esperienza e tecnica appresa nel corso del tempo attraverso un percorso tradizionale e consolidato nel progettare e realizzare un particolare prodotto. Non è facile insegnare e tramandare le abilità e le conoscenze apprese da artigiani specializzati capaci di creare e sviluppare, con materiali ricercati, sofisticati prodotti artigianali e unici, in un impeccabile bilanciamento tra tradizione e modernità. Questa difficoltà si riscontra soprattutto al giorno d'oggi, dal momento che il modello di insegnamento "maestro-apprendista" non è più comune come una volta.

Date queste premesse è chiaro che all'interno di un'azienda artigianale è di notevole importanza l'archivio, cioè una collezione di saperi materiali e immateriali; questo è il primo passo in cui la tecnologia può fare la differenza, codificando e preservando l'esperienza in supporti digitali. Attraverso l'archivio è possibile, oltre a preservare informazioni e tecniche, valutare e discutere i processi di design e re-design di un particolare prodotto. Vi è in questo modo un nuovo modello di produzione che coniuga l'unicità del prodotto artigianale di tradizione con l'utilizzo delle nuove tecnologie. E' come se esistesse una fusione tra il concetto di "analogico" e quello di "digitale". Per analogico si intende, in questo contesto, il

saper fare tradizionale, la cultura materiale che si è tramandata di generazione in generazione all'interno delle piccole e medie imprese artigiane. Per digitale si intende, invece, le innovazioni che attraverso l'uso della tecnologia contaminano il mondo analogico. Il mix di questi due aspetti crea un mondo di produzione nuovo, come dimostra l'attuale fenomeno dei *makers*[3].

L'incontro tra l'unicità del prodotto artigianale tradizionale con le nuove tecnologie è già stato trattato in diversi lavori, come ad esempio [35] riguardante il lavoro a maglia, il cui scopo però è quello di far risaltare l'aspetto sociale piuttosto che quello di ricavare, attraverso la codifica, la comprensione ed il riconoscimento dei gesti compiuti. In [36, 37] emerge invece un nuovo approccio, che fonde e tenta di creare le basi di una ricerca che congiunge le emozioni e le conoscenze artigianali, basate su dettagli tecnici e rigorose analisi, con quell'estro che solo lavorando per tanti anni in un settore si può acquisire.

In questo lavoro di tesi, invece, si vuole focalizzare l'attenzione sulla codifica pura di un gesto, piuttosto che sulla sua divulgazione. Inoltre si utilizzano qui tecnologie non invasive, come invece viene fatto in [19] attraverso l'utilizzo di sensori, marcatori o guanti indossabili.

Dai test effettuati sono emersi risultati molto positivi ed incoraggianti che vengono esposti in [38], che si basa proprio sulla presente tesi. Sono stati poi effettuati ulteriori estensioni dell'applicazione, che è ora in grado di discriminare la presenza una o due mani in movimento all'interno della scena, come descritto in [39].

3.3 Casi d'uso ed esempi di movimenti

Una applicazione reale del dispositivo proposto è quella della realizzazione di scarpe artigianali di alta moda. Citando Stefano Bemer[4], quello che riesce a trasmettere una scarpa fatta a mano è un quid che nessuna industria potrà arrivare a copiare, perchè è prima di tutto, una scarpa fatta per una persona, non è una

[3] *Con questo nome vengono definiti i nuovi artigiani, maker appunto, che sanno gestire i nuovi linguaggi informatici e cooperare in rete*

[4] http://www.bemers.it/ è considerato tra i più noti calzolai d'Europa per la sua qualità e professionalità dimostrata nella lavorazione su misura.

scarpa fatta per milioni di persone o per un mercato, è fatta per un individuo. Principalmente è fatta con dei canoni di lavorazione così tradizionali, attenti a tutti i dettagli, sia estetici ma soprattutto tecnici. Il bello di una scarpa fatta a mano, infatti, non è quello che si vede esternamente ma quello che c'è dentro: componenti, colle, cuciture, pelli. Tutto questo sapere artigiano è frutto di anni di lavoro e sperimentazione. Mediante l'applicazione sviluppata, si vuole contribuire così a codificare e mantenere questi saperi. Sono stati scelti dei casi d'uso, a titolo di esempio, atti ad illustrare il sistema e le sue caratteristiche.

3.3.1 Caso 1: Martellare la suola delle scarpe

Nel caso in questione, si considera l'atto di martellare da parte di un artigiano, intento a creare la forma della scarpa, imprimendo il collante in maniera più solida.

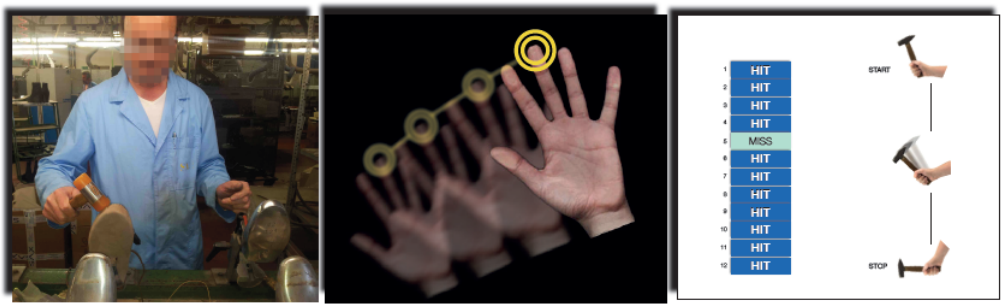


Figura 3.1: da sinistra a destra: (a) gesto del martellare, (b) tracciamento della posizione della mano, (c) performance del sistema con l'azione del martellare

Nel primo riquadro a sinistra viene mostrato l'artigiano all'opera nel martellare un paio di scarpe. La camera è posizionata ad una distanza di 80 cm. In questo caso il movimento è eseguito da una singola mano, mentre l'altra rimane pressochè ferma. Il sistema di tracciamento tiene conto del movimento frame per frame. I parametri "hit" e "miss" indicano se l'algoritmo ha rilevato o meno il movimento. In questo caso il movimento è rappresentato da un colpo di martello dall'alto verso il basso. Essendo il colpo abbastanza rapido, il movimento viene rappresentato da 12 frame. Con il nostro approccio, solamente in un frame non

è stata stimata la posizione corretta della mano, in tutti gli altri l'operazione è andata a buon fine (riquadro (c) della figura 3.1).

3.3.2 Caso 2: Cucire un orlo in punta alla scarpa

Qui viene considerata l'azione del cucire. Più precisamente viene effettuata una cucitura di un orlo sulla punta della scarpa per scopi estetici.

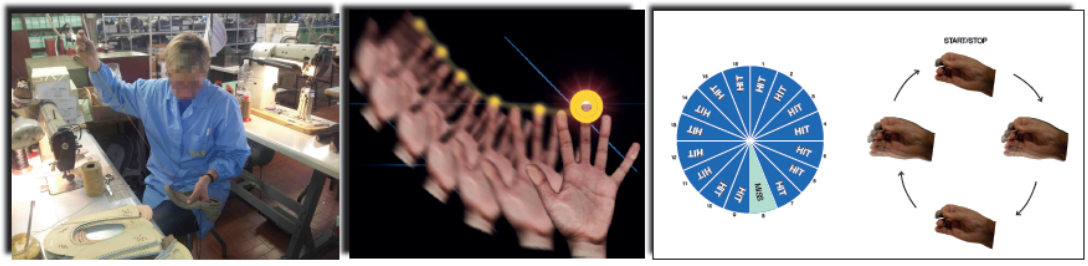


Figura 3.2: da sinistra a destra: (a) gesto del cucire, (b) tracciamento della posizione della mano, (c) performance del sistema con l'azione del cucire

La prima immagine riguarda l'azione compiuta dall'artigiano mentre cuce. La camera è posizionata ad una distanza di circa 1 metro dall'esecutore, in modo tale da catturare i movimenti ampi della cucitura (ad esempio, il momento in cui si tende il filo per stringere il nodo fatto) e non "perdere di vista" la mano in movimento. Anche in questa tipologia di movimento è richiesto il movimento principalmente di una mano; la seconda esegue piccoli movimenti per tenere ferma la scarpa, ma ai fini del gesto da riconoscere, sono irrilevanti. Si è scelto di schematizzare il movimento di cucitura e considerarlo come una rotazione: si parte dal punto in cui si infila l'ago nella pelle e si compie il movimento sino a tornare nella posizione dove si è inserito l'ago. In questo caso il gesto è rappresentato da 16 frame e di questi, solamente in una occorrenza, il sistema non è stato in grado di stimare una posizione della mano correttamente(riquadro (c) della figura 3.2).

3.3.3 Caso 3: Intagliare la suola della scarpa

Il gesto prevede la creazione di un solco ad una distanza di pochi centimetri dal bordo della suola. Utilizzando questo tracciato, attraverso dei nodi e delle cuciture, si va ad unire la pelle alla suola, per creare un corpo unico. La distanza della camera, rispetto all'artigiano, è di circa 50 cm. In questo modo si possono catturare i piccoli movimenti, tipici di questa azione. Il gesto è compiuto principalmente da una mano, la seconda funge solamente da perno per tenere salda la presa della scarpa, e i suoi movimenti sono trascurabili.

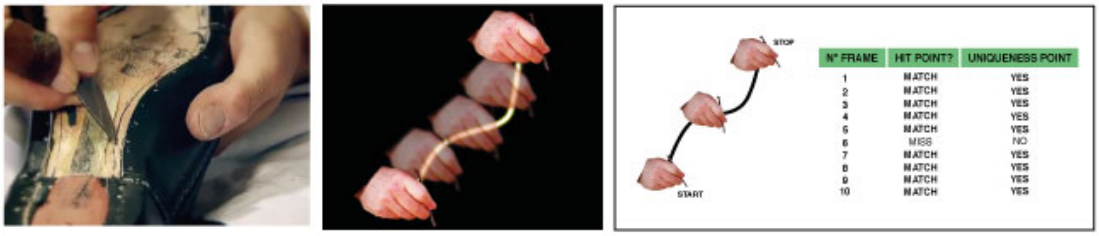


Figura 3.3: da sinistra a destra: (a) movimento di intaglio, (b) tracciamento di una mano, (c) performance del sistema con l'azione di intaglio

La prima foto si riferisce sempre alla visualizzazione del movimento effettuato dall'artigiano. La seconda, sta ad indicare l'effetto di tracciamento del movimento da parte dell'applicazione: in questo caso, per avere delle percentuali sulla bontà del metodo, si è schematizzato il movimento facendolo partire dal basso, per poi seguire le curve della suola sino alla punta in alto. Dal riquadro (c) della figura 3.3 si può constatare che su 10 frame utilizzati per registrare il movimento, anche qui solamente in un frame l'algoritmo non riusciva a stimare correttamente la posizione della mano[5].

3.3.4 Caso 4: Annodare i fili

Questo movimento segue, in successione, quello del caso 3. Una volta che viene creato il solco sulla suola, viene fatto passare il filo tra la pelle della scarpa e la suola stessa. L'intreccio del filo e il successivo nodo servono per saldare,

[5]In alcuni test erano al massimo due.

almeno in prima battuta, la pelle alla suola. Successivamente poi con l'uso di colle e cuciture, si andrà a rifinire e saldare questa unione. La camera, in questo caso, viene posta ad una distanza di circa 1,20 cm. Qui la distanza è maggiore rispetto agli altri casi perchè il movimento richiede che le due mani divergano, in movimento, tra loro; dato che il filo è considerevolmente lungo, per evitare di far uscire le mani dall'obiettivo, è bene tenere un campo visivo più ampio.

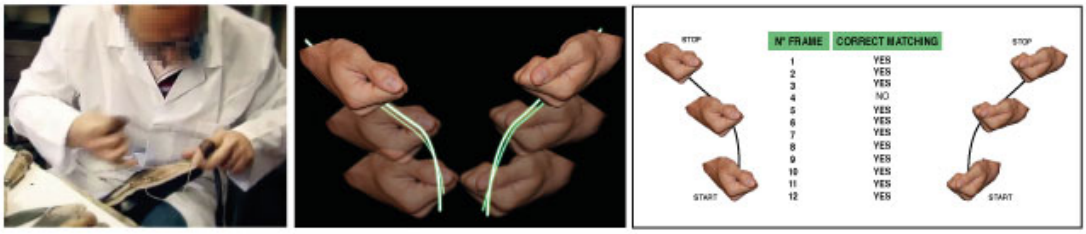


Figura 3.4: da sinistra a destra: (a) movimento reale, (b) tracciamento delle due mani, (c) performance del sistema con l'azione di intaglio

In questo caso si tratta di rilevare, contemporaneamente, il movimento delle due mani. Il sistema è in grado di determinare, nella scena inquadrata, il fatto che il gesto è svolto da due mani e le mappa a video, tenendo conto, di volta in volta, della posizione assunta da ciascuna mano. Anche in questo caso d'uso la bontà della stima della posizione della mano è molto buona. Nei test effettuati, su un valore che varia tra i 12 e 20 frame, si può riscontrare che al più non si riesce a stimare una posizione nel 3% dei casi.

3.3.5 Screenshot della stima di posizione delle mani

Il primo screenshot che viene illustrato in figura 3.5, riguarda il *Caso d'Uso 3*. Sono stati presi solamente 6 frame per far capire come il sistema calcoli la posizione. La webcam utilizzata è caratterizzata da un frame rate di 30 fps ed è posta ad una distanza di 50 cm dall'esecutore.

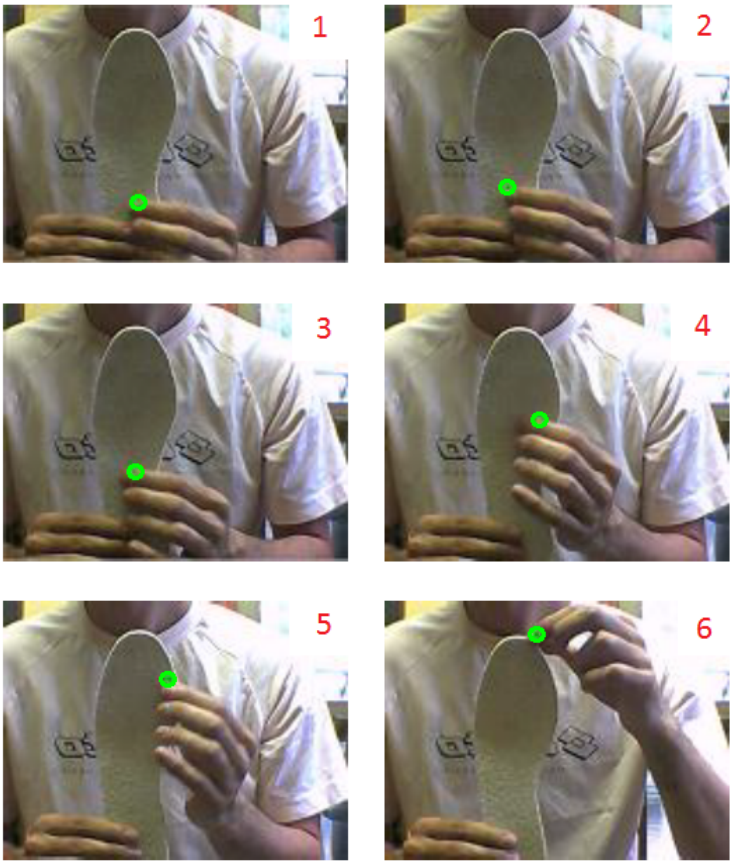


Figura 3.5: Sezioni di frame con determinazione della posizione della mano in movimento

Il secondo sceenshot mostra il *Caso d'Uso 4*. La rilevazione coinvolge il movimento delle due mani, e segue il movimento schematizzato nella 3.4 (c). Per ragioni visive, e di comprensione, i puntatori delle due mani sono di colore diverso. L'agoritmo utilizza un valore di soglia, al di sotto del quale, le mani si considerano unite, o comunque molto vicine. A livello illustrativo, questo viene visualizzato solamente con un puntatore (destro o sinistro non ha importanza), mentre l'altro viene omesso. Si veda a tal proposito il riquadro 1 o 8 della figura 3.6, che corrispondono all'inizio ed alla fine del movimento, in cui le mani sono quasi unite.

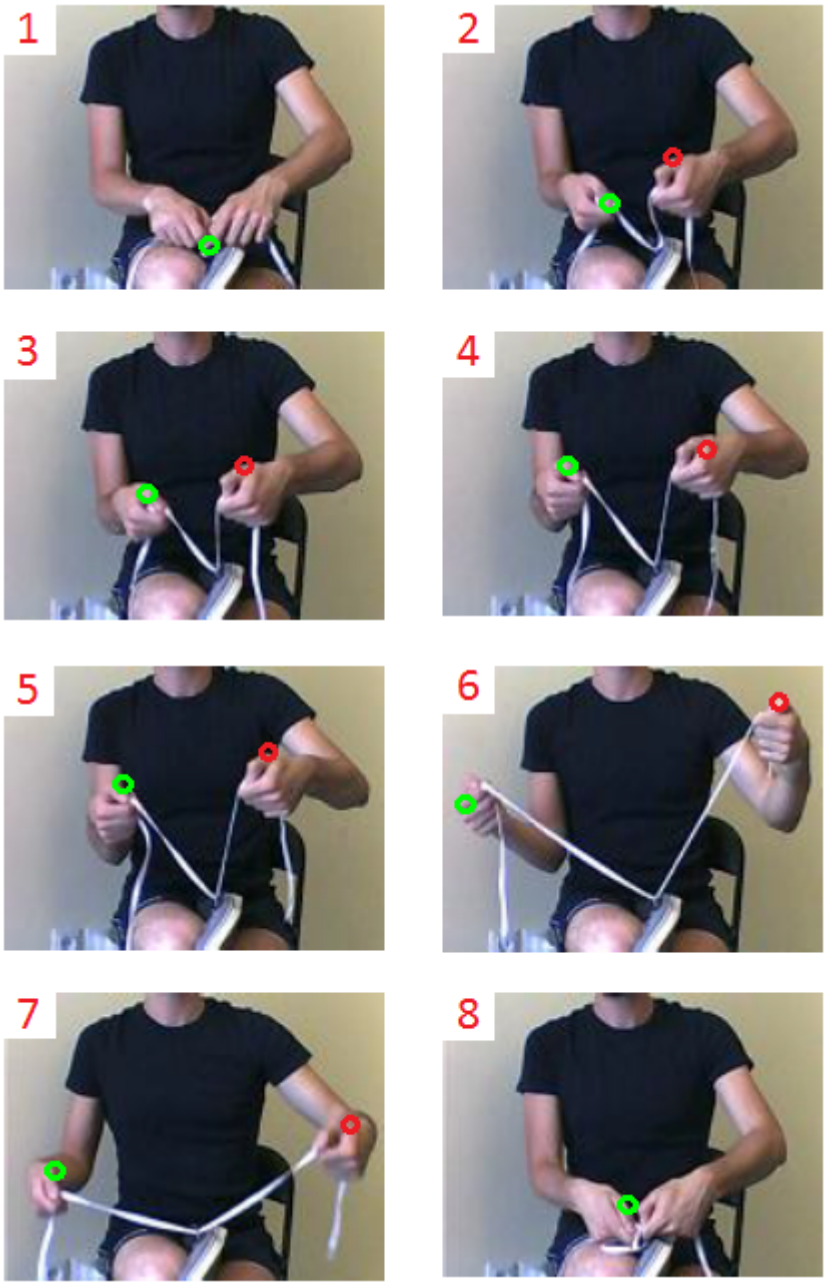


Figura 3.6: Sequenza con frame più significativi del movimento spiegato in fig. 3.4

Capitolo 4

Descrizione implementativa

4.1 Struttura del prototipo

In questa parte viene descritta, attraverso il diagramma UML di classe (figura 4.2), la struttura principale del prototipo sviluppato in questo lavoro di tesi. Si utilizzano tre pacchetti della libreria OpenCV: cv, cxcore e HighGUI che, forniscono API[1] per trattare le immagini catturate dalla camera, gestiscono tipi di dato rappresentanti immagini e quelle operazioni inerenti l'analisi delle immagini.

Di seguito sono riportate le classi create:

- *Area*: rappresenta una area sensibile al movimento, tiene conto della posizione a video dell'area e si occupa della renderizzazione a video di un particolare punto;
- *CellMatrix*: rappresenta una cella della matrice delle occorrenze che viene utilizzata per analizzare il movimento;
- *FrameProcessor*: gestisce i frame presi in input e attua dei filtri per eliminare rumori;
- *Grid*: rappresenta una griglia formata da aree sensibili al movimento;
- *Kalman*: calcola il filtro di Kalman utilizzando precisi parametri di configurazione settati dal prototipo;

[1] Application Program Interface

- *ManageKmeans*: calcola l'algoritmo di clusterizzazione Kmeans con $K = 2$;
- *OccurrenceMatrix*: rappresenta la matrice totale relativa alla griglia di rilevamento Grid;
- *StreamVideo*: fornisce i frame catturati dalla camera e si occupa della gestione di finestre;

Nella figura 4.1, viene illustrato la parte di "front-end" di rilevazione del movimento. La classe Grid è una griglia, visualizzabile a video, che rileva il movimento compiuto. La griglia è formata da tante aree sensibili definite dalla classe Area.

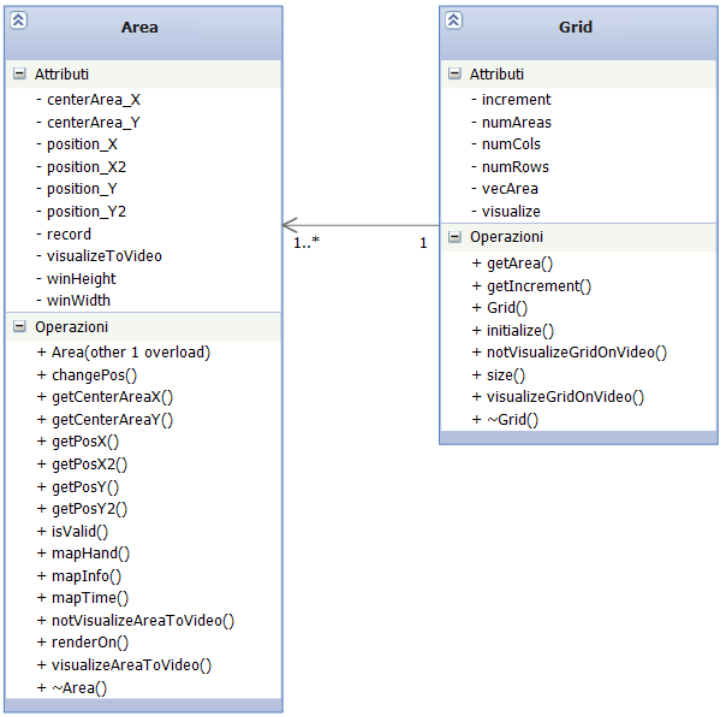


Figura 4.1: Relazione tra la griglia di rilevamento e le componenti di tale griglia

Circa la parte di "back-end" della rilevazione del movimento, nella figura 4.3 viene definita la relazione che intercorre tra la matrice delle occorrenze, formata da aree di tipo CellMatrix, e la Grid composta da aree sensibili. Le celle di tipo CellMatrix, vengono anche utilizzate per la clusterizzazione[2]. La definizione data

[2]Si veda paragrafo 4.2.2

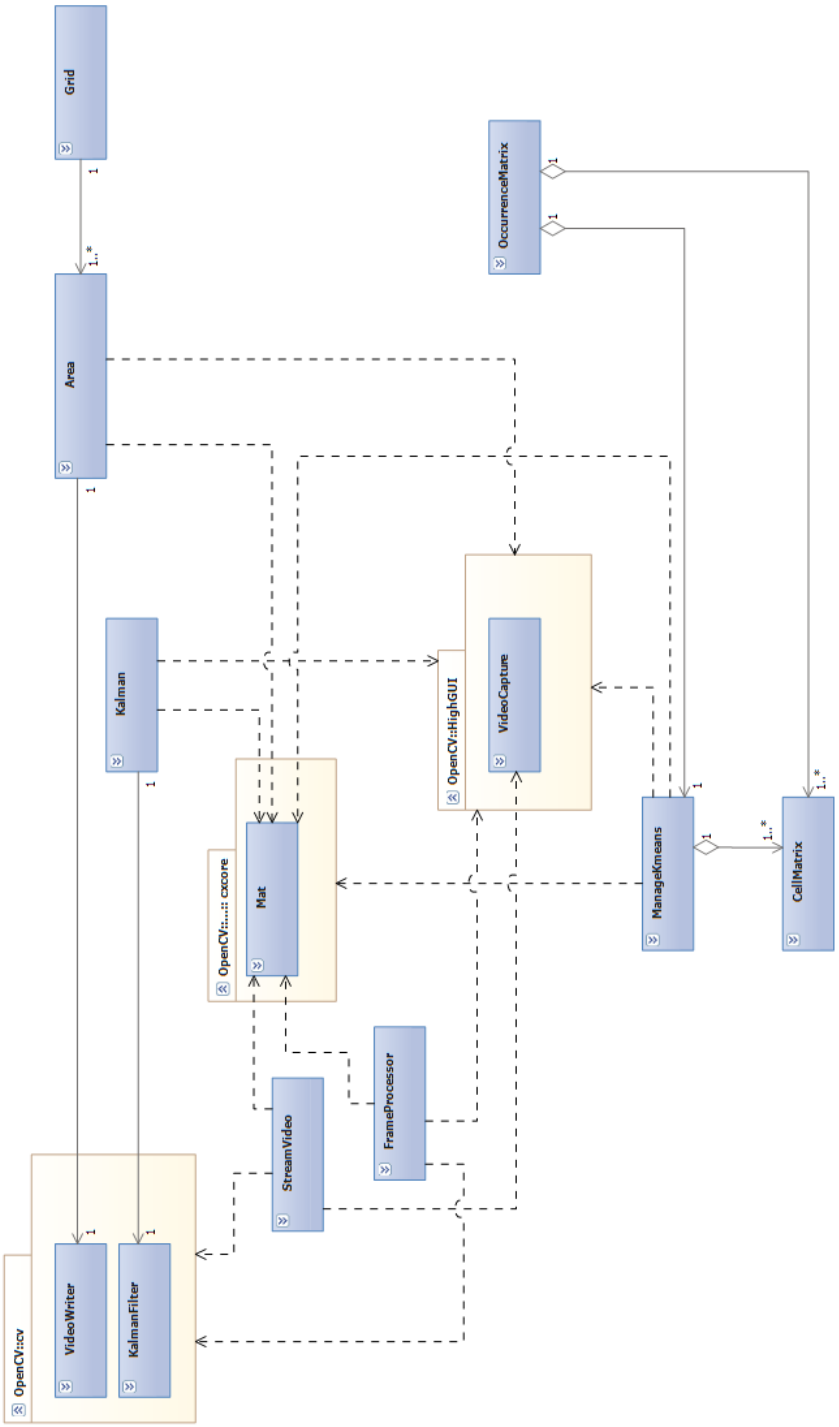


Figura 4.2: Diagramma UML delle classi

qui di “front-end” e “back-end” si riferisce al fatto che la rilevazione può essere vista in due modi:

1. **visuale**: il rilevamento è catturato calcolando i pixel rilevanti il movimento (front-end);

2. **matriciale**: il rilevamento è visto come una serire di occorrenze poste in una matrice (back-end);

Nel prototipo il rilevamento visuale precede quello matriciale: gli algoritmi di filtraggio dei punti più interessanti circa la stima del movimento, sono calcolati utilizzando il matriciale e successivamente vengono mappati i risultati nel visuale. Ciclicamente, i passaggi sono i seguenti:

$$\text{Visuale} \longmapsto \text{Matriciale} \longmapsto \text{filtraggio(Matriciale)} \longmapsto \text{Visuale}.$$

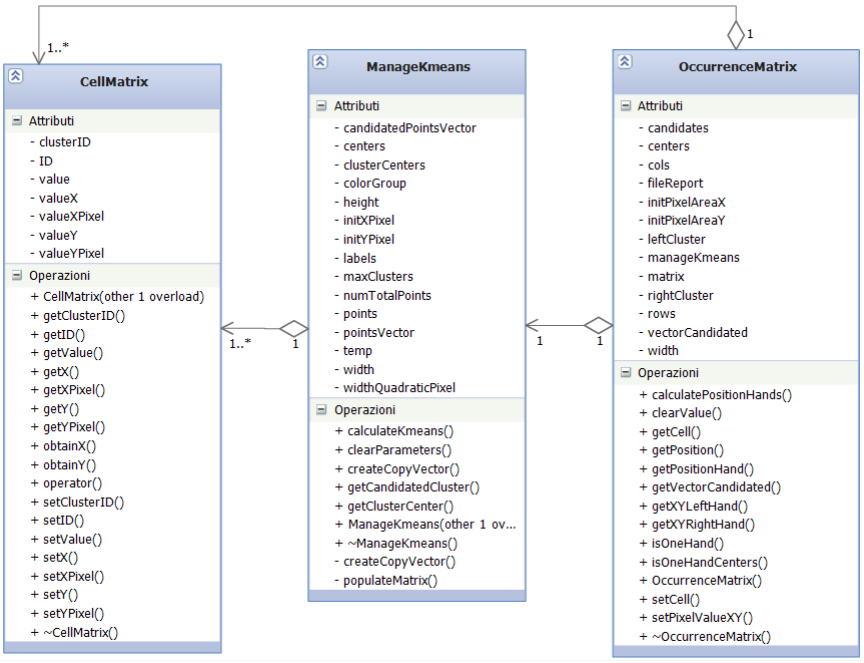


Figura 4.3: Specifica relazione tra le classi CellMatrix, ManageKmeans e OccurrenceMatrix

4.2 Tecnologie e Metodologie utilizzate

In questa sezione vengono presentate le librerie e gli algoritmi impiegati in questo lavoro di tesi. La trattazione di ogni sottosezione ha lo scopo di far comprendere il motivo per cui un dato approccio è stato creato e come è stato poi impiegato.

4.2.1 OpenCV: libreria per la computer vision

La visione artificiale sta rapidamente crescendo grazie all'alta resa e al basso costo dei dispositivi di cattura dell'immagine e al fatto che numerosi algoritmi sono ormai considerati maturi. OpenCV[3] [40, 41] è una libreria open-source, il cui principale scopo è la visione artificiale real-time, sviluppata per architetture Intel. Il suo campo di utilizzo spazia in differenti aree di ricerca, alcuni esempi riguardano: l'interazione uomo-macchina, l' identificazione e riconoscimento di oggetti, il face recognition, il tracciamento del moto, gli algoritmi per la robotica. OpenCV implementa un significativo insieme di tool per l'analisi delle immagini ed è compatibile con Image Processing Library (IPL)[4]. Gli algoritmi implementati sono relativi allo "stato dell'arte" e riguardano la binarizzazione, il filtraggio, la gestione di immagini piramidali, le tecniche per la calibrazione della camera, il tracciamento di features, l'analisi di forme, la ricostruzione 3D[5], il riconoscimento di oggetti con catene nascoste di markov e gli istogrammi. Originariamente, la libreria era scritta solamente in linguaggio C; successivamente sono stati scritti wrapper per altri linguaggi come C#, Python, Ruby e Java. A partire dalla versione 2.0, OpenCV include le interfacce tradizionali della versione 1, e inoltre, anche le nuove interfacce C++ orientate agli oggetti. Attraverso l'uso di queste nuove interfacce, si riduce il numero di linee di codice necessarie alla chiamata di determinate procedure, e vengono ridotti gli errori comuni di programmazione riguardo il fenomeno della "memory leak", tramite allocazioni e deallocazioni di strutture dati. Per ottimizzare ancora di più le prestazioni di calcolo, sono state

[3] Acronimo che sta per Open Computer Vision.

[4] Libreria sempre sviluppata da Intel che implementa operazioni di basso livello su immagini digitali.

[5] View Morphing.

implementate le interfacce per la libreria CUDA[6]. OpenCV implementa progetti e risultati di ricerca scientifici che riguardano la computer vision e il machine learning, fornendo così risorse preziose anche al di fuori di particolari centri di ricerca.

OpenCV è sostanzialmente strutturato in cinque componenti principali, quattro dei quali sono illustrati nella figura 4.4. Il componente CV tratta l'elaborazione di immagini e algoritmi di computer vision, ML è la libreria di machine learning, che include molti classificatori statistici e strumenti di clustering. HighGUI contiene le routine di I / O e le funzioni per memorizzare e caricare video o immagini; CXCore contiene le strutture dati di base; CVAux contiene algoritmi sperimentali come quello riguardante la segmentazione del background con il foreground.

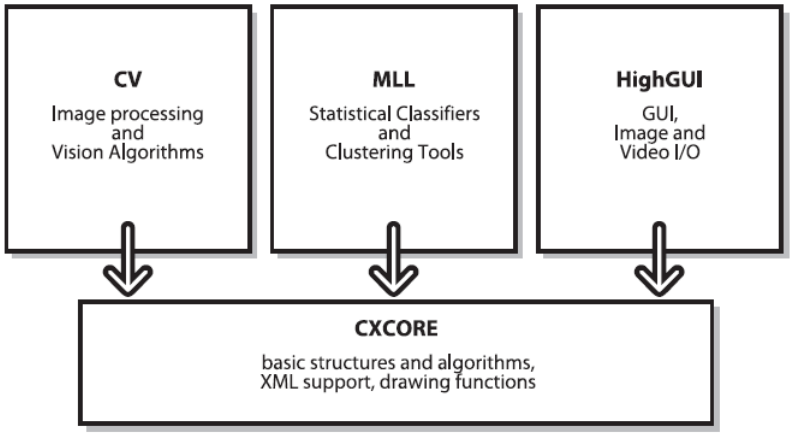


Figura 4.4: Struttura OpenCV

Struttura matriciale utilizzata: CvMat

Il concetto di una matrice, in OpenCV, è più astratto del concetto di matrice nell'algebra lineare. In particolare, gli elementi non devono essere semplici numeri.

[6]Acronimo di Computer Unified Device Architecture: è una architettura hardware per l'elaborazione parallela creata da NVIDIA. Tramite l'ambiente di sviluppo CUDA, i programmatori software, sono in grado di scrivere applicazioni capaci di eseguire calcolo parallelo sulle GPU delle schede video NVIDIA.

Ad esempio, la routine che crea una nuova matrice bidimensionale ha il seguente prototipo:

```
cvMat* cvCreateMat(int rows, int cols, int type);
```

Questo tipo, può essere uno qualsiasi di una lunga lista di tipi predefiniti. La matrice può consistere di float a 32-bit (CV_32FC1), di interi senza segno a 8-bit (CV_8UC3), o di innumerevoli altri elementi. Un elemento di CvMat non è necessariamente un singolo numero, ma può rappresentare più valori per un singola voce nella matrice e permette di rappresentare più canali di colore in un'immagine RGB. Il modo più semplice per accedere ad un elemento è l'utilizzo della macro CV_MAT_ELEM (). Questa macro prende la matrice, il tipo di elemento da recuperare, la riga e la colonna restituendo l'elemento.

```
cvMat* mat = cvCreateMat(5, 5, CV_32FC1);
float element_3_2 = CV_MAT_ELEM(*mat, float, 3, 2);
```

Questa struttura sarà la base per rappresentare il contenuto dei frame e altre immagini elaborate con la libreria.

Gaussian-Blur

La sfocatura[7], è un'operazione semplice e frequente nell'analisi delle immagini. Attraverso una funzione gaussiana, si ottiene una immagine sfocata. Solitamente è utilizzata per ridurre il rumore generato dalla videocamera. La sfocatura è molto importante quando si vuole ridurre la risoluzione di una immagine in una tipologia particolare (ad esempio, piramidale). Il filtro gaussiano è sicuramente il più utile; esso è caratterizzato dalla convoluzione di ogni punto dell'array di input con un kernel gaussiano.

[7]In questo contesto si può leggere anche soppressione.

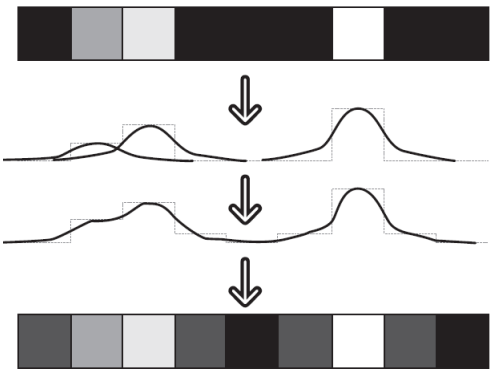


Figura 4.5: Sfocatura gaussiana con un array unidimensionale

Il nuovo valore del pixel è la media pesata dei valori nel suo intorno. Tali pesi sono distribuiti secondo una funzione gaussiana, più è larga la campana e maggiore sarà l'effetto di smooting.

Absdiff

Absdiff è la funzione che calcola la differenza assoluta tra due immagini. Viene utilizzata nel calcolo della sottrazione di sfondo e consiste nel sottrarre un frame da un altro. Quello che rimane dalla differenza sarà etichettato come foreground, cioè come elementi in primo piano. Per semplicità, supponiamo di avere una immagine a tre canali: FrameTime1, FrameTime2 e FrameForeground. FrameTime1 rappresenta il primo frame catturato in scala di grigi, FrameTime2 invece rappresenta il frame catturato all'istante attuale, anch'esso in scala di grigi. Si può usare il seguente codice per determinare il valore assoluto di oggetti in primo piano e metterlo in FrameForeground:

```
1 absdiff(this->img1, this->img2, this->processedColorImage);
```

Threshold

Spesso, si ha la necessità di lavorare con immagini filtrate, e si desidera applicare una soglia a determinati valori o pixel di un'immagine; se questi valori sono maggiori o uguali del valore di soglia vengono accettati, altrimenti vengono rifiutati. La funzione threshold in OpenCV mette in atto questi discriminanti[43]. L'idea

di base è che, dato un array e un preciso valore di soglia, si confronta ogni elemento della matrice per determinare se questo è inferiore o superiore alla soglia. Come mostrato nella figura 4.6, ogni tipo di filtro corrisponde ad una particolare operazione di confronto tra il pixel e la soglia.

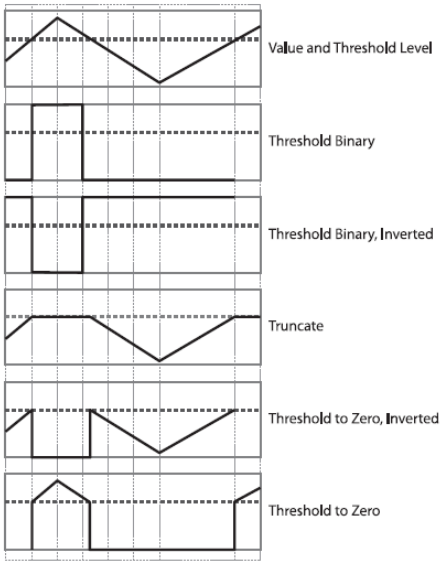


Figura 4.6: Tipologie di filtro in OpenCV

La chiamata a funzione in OpenCV è la seguente:

```
/*esecuzione di un filtro soglia binario sull'immagine
  per lasciare solo pixel bianchi e neri; mette
  in this->processedImageGrayScale, il risultato del
  filtraggio di ogni pixel di della stessa immagine.
  CV_THRESH_BINARY indica di applicare un threshold
  binario
*/

threshold(this->processedImageGrayScale, this->
   processedImageGrayScale, 40, 255, CV_THRESH_BINARY);
```

4.2.2 K-means: algoritmo di clustering

Il clustering è il processo di raggruppamento di un insieme di oggetti in classi di oggetti simili. Esistono due tipologie di clustering:

1. clustering basato sulla distanza: l'appartenenza ad un gruppo è caratterizzata dalla vicinanza di quell'oggetto al gruppo;

2. clustering concettuale: gli oggetti appartengono ad un cluster se questo definisce un concetto comune ai diversi oggetti.

Questo algoritmo di clustering permette di suddividere gruppi di oggetti in K partizioni, sulla base dei loro attributi. Si assume che gli attributi degli oggetti possano essere rappresentati come vettori, e che quindi formino uno spazio vettoriale. L'obiettivo è quello di minimizzare la varianza totale intra-cluster. Ogni cluster viene identificato mediante un centro (punto medio). Vi è una procedura iterativa dove, inizialmente, vengono create K partizioni e, successivamente, ad ogni partizione si assegnano i punti d'ingresso in modo random, o utilizzando delle euristiche. Viene quindi calcolato il centroide di ogni gruppo; si costruisce così una nuova partizione, associando ogni punto dato in input al cluster che ha il centro più vicino ad esso. Fino alla convergenza dell'algoritmo, vengono ricalcolati i centroidi per i nuovi cluster.

Formalmente, dati N oggetti con i attributi, visti come vettori in uno spazio vettoriale $i-$dimensional e dato l'insieme degli oggetti $X = X_1, ..., X_N$, una partizione degli oggetti è un gruppo $P = P_1, ..., P_K$ che soddisfa:

- $\bigcup_{1}^{K} P_i = X$, ovvero l'unione di tutti i cluster deve contenere tutti gli oggetti di partenza;
- $\bigcap_{1}^{K} P_i = \emptyset$, ovvero ogni elemento può appartenere ad un solo cluster;
- $\emptyset \subset P_i \subset X$, ovvero almeno un oggetto deve appartenere ad un cluster e nessun cluster può contenere tutti gli oggetti.

Una partizione è rappresentata con una matrice $U \epsilon \mathbb{N}^{\mathbb{K} \times \mathbb{N}}$, dove il generico elemento $l_{i,j} = 0, 1$ indica l'appartenenza dell'oggetto j al cluster i. Con $C = C_1, ..., C_K$

si indica l'insieme dei K centroidi. La funzione obiettivo sarà:

$$V(U,C) = \sum_{i=1}^{K} \sum_{X_j \epsilon P_i} \|X_j - C_i\|^2. \tag{4.1}$$

Con questa formula, si calcola il minimo seguendo la procedura iterativa[46, 47]:

- Generare U_v e C_v casuali;
- Calcolare U_n che minimizza $V(U, C_v)$;
- Calcolare C_n che minimizza $V(U_v, C)$;
- Se si arriva a conversione, allora ci si ferma, altrimenti si pone $U_v = U_n$, $C_v = C_n$ e si ritorna al secondo step.

La convergenza si raggiunge quando non c'è nessun cambiamento nella matrice U, cioè quando la differenza fra i valori della funzione obiettivo in due iterazioni successive non supera una soglia prefissata. L'algoritmo non garantisce il raggiungimento dell'ottimo globale. La qualità della soluzione finale dipende largamente dal set di cluster iniziale e può, in pratica, ottenere una soluzione ben peggiore dell'ottimo globale. Dato che l'algoritmo è di solito estremamente veloce, è possibile applicarlo più volte e, fra le soluzioni prodotte, scegliere quella più soddisfacente. Uno svantaggio dell'algoritmo è che esso richiede di scegliere, a priori, il numero di cluster (K) da trovare.

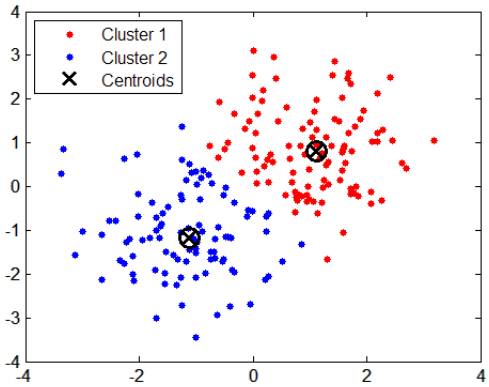


Figura 4.7: Esempio di clusterizzazione K-means con K=2, lo stesso valore che si utilizza in questo lavoro

La chiamata a funzione in OpenCV è la seguente:

```
/*points - immagine di input
  clusterCount - numero di cluster
  labels - array intero di input/output che memorizza
           gli indici cluster per ogni campione
  TermCriteria - criterio di terminazione dell'algoritmo
  3 - Flag per specificare il numero di volte che viene
      eseguito l'algoritmo usando differenti etichettature
      iniziali.
  L'algoritmo restituisce le etichette che producono la
  miglior compattezza (vedi l'ultimo parametro funzione)
  KMEANS_PP_CENTERS - Inizializzazione Arthur e
                      Vassilvitskii
  centers - Matrice di uscita dei centri di cluster,
            una riga per ogni centro di cluster
*/


kmeans(Punti, maxClusters, labels,
       TermCriteria( params ),
       3, KMEANS_PP_CENTERS, centri);
```

L'inizializzazione utilizzata è esposta in[44].

4.2.3 Filtro discreto di Kalman: filtro per valutare un sistema dinamico

Il filtro di Kalman rappresenta una soluzione ricorsiva al problema del filtraggio lineare di dati discreti [49]. Questo filtro è un insieme di equazioni matematiche che forniscono una soluzione, computazionalmente efficiente, del metodo dei minimi quadrati[8]. Vengono determinate stime degli stati passati, presenti e futuri e questi possono essere calcolati anche quando la natura del sistema che si vuole modellare non è conosciuta a priori. Il filtro di Kalman fornisce una stima di un processo utilizzando una forma di controllo basata sui *feedback*: il filtro sti-

[8]Tecnica di ottimizzazione che permette di trovare una funzione, chiamata curva di regressione, che si avvicini il più possibile ad un insieme di dati (tipicamente punti del piano).

ma il processo ad un certo istante ed ottiene un feedback che rappresenta una misurazione affetta da rumore. Le equazioni del filtro di Kalman sono di due tipi:

- *equazioni di aggiornamento del tempo*: responsabili della previsione dello stato attuale e della covarianza dell'errore, valutati in modo tale da ottenere una stima, a priori, del passo successivo;
- *equazioni di aggiornamento della misura*: responsabili del feedback perchè queste equazioni vengono impiegate per unire una nuova misurazione, con la stima a priori, al fine di ottenere una migliore stima a posteriori.

L'algoritmo del filtro viene visto come una procedura ricorsiva dove le equazioni di aggiornamento del tempo forniscono una previsione, mentre gli aggiornamenti circa la misura determinano un miglioramento della stima, introducendo informazione contenuta nella misurazione.

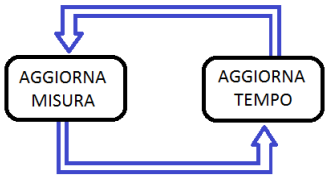


Figura 4.8: Fase ricorsiva del filtro: Predizione e Aggiornamento

L'algoritmo completo viene definito in questo modo:

- $K = K_0 > 1$, dove K tiene conto dell'istante;
- Modello del sistema con
  - **Matrice di stato** Φ_K: che descrive l'evoluzione libera della variabile di stato rispetto al suo valore attuale;
  - **Matrice degli ingressi** Ψ_K: che descrive l'evoluzione forzata della variabile di stato rispetto al valore attuale di ingresso;
  - **Matrice delle uscite** C_K: che descrive il valore assunto dalle variabili misurate, in funzione del valore attuale della variabile di stato;
- Varianze degli errori con

  - **Matrice di covarianza del disturbo sullo stato** Q_K: è la variabilità statistica del vettore dei disturbi sullo stato (vettore a media nulla) ovvero, la potenza del disturbo introdotto nel sistema che devia l'andamento delle variabili di stato, rispetto a quello prevedibile dalla conoscenza del vettore degli ingressi e dalla legge lineare che ne governa l'evoluzione;
  - **Matrice di varianza del rumore sulle misure** R_K: è la variabilità statistica del vettore dei disturbi di misura (vettore a media nulla) e rappresenta la potenza del disturbo sulla misura introdotta su ciascuna delle misure accessibili;

- Condizioni iniziali

  - x_0: vettore di stato[9] iniziale;
  - Matrice di varianza dell'errore sullo stato (al tempo 0) P_0 : rappresenta la variabilità dell'errore sulla stima dello stato conseguente ai due fattori di disturbo (errore di misura e disturbo dello stato);

Verranno di seguito riportate le formule attuate in ogni passo dell'algoritmo, riferiti al diagramma di flusso presente in fugura 4.9

1. *inizializzazione dell'algoritmo*: vengono effettuate correzioni al valore attuale
   $x_{K-1}(+) = \bar{x}_0 e P_{K-1}(+) = P_0$;

2. *predizione nuovo valore della variabile di stato*:
   $x_K(-) = \Phi_{K-1} x_{K-1}(+)$;

3. *aggiornamento della varianza dell'errore di stato*:
   $P_K(-) = \Phi_{K-1} P_{K-1}(+) \Phi_{K-1}^T + Q_{K-1}$;

4. *aggiornamento della matrice di correzione di stato*:
   $K_K = P_K(-) C_K^T [C_K P_K(-) C_K^T + R_K]^{-1}$;

[9] In un sistema lineare, lo stato rappresenta l'energia presente in uno degli accumulatori di energia interni al processo di cui il sistema lineare ne rappresenta una modellazione matematica.

5. *correzione per stima di stato:*
   $x_K(+) = x_K(-) + K_K[y_K - C_K x_K(-)]$;

6. *aggiornamento del ciclo*:
   $K = K + 1$;

7. *aggiornamento della varianza dell'errore sullo stato*:
   $P_K(+) = [1 - K_K C_K] P_K(-)$;

8. *uscita*: stato $x_K(+)$.

I simboli $(+)$ e $(-)$ corrispondono allo stato attuale del sistema: se una variabile ha associato il simbolo $(+)$ si intende che è stata applicata la correzione dovuta al valore attuale della misura; se invece, la variabile ha associato il simbolo $(-)$ significa che il valore è stimato per predizione. Alla fine dell'algoritmo, in uscita si ha un valore di x che è solamente approssimato al valore reale.

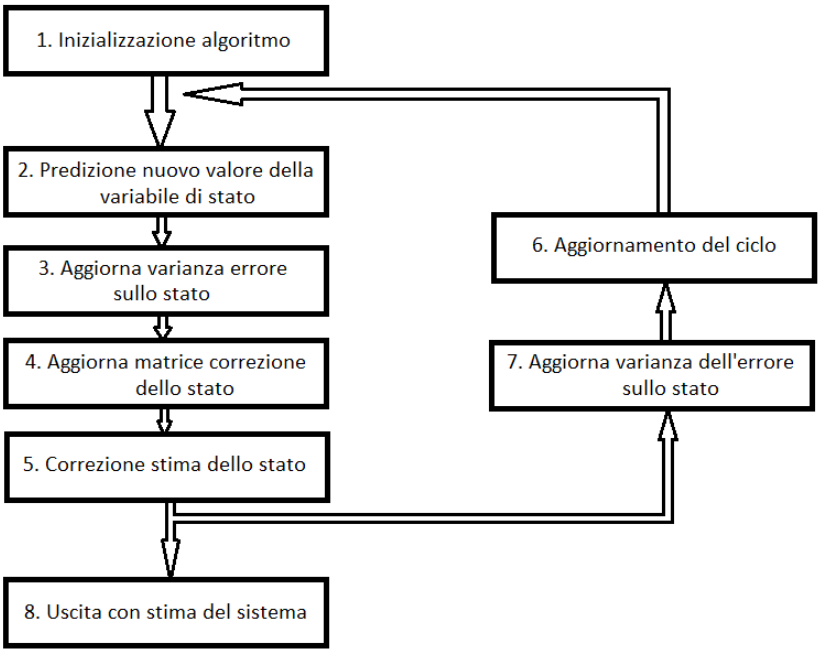


Figura 4.9: Diagramma di flusso dell'algoritmo

Un esempio di utilizzo del filtro di Kalman con OpenCV, è illustrato in figura 4.10

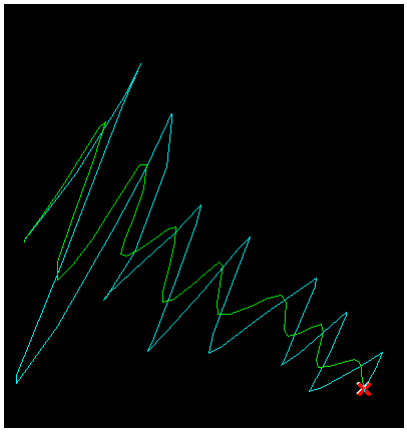


Figura 4.10: Tracciamento del filtro di Kalman (linea verde)

La linea azzurra corrisponde allo spostamento del mouse e la linea verde è la mappatura del tracciato ad opera del filtro.

A livello algoritmico, in OpenCV è sufficiente impostare

```
/*parametro 1: dimensionalità degli stati
  parametro 2: dimensionalità delle misurazioni
  parametro 3: dimensionalità del vettore di controllo
*/
KalmanFilter KF(4, 2, 0);

/*
stati (x, y, Vx, Vy) che sarebbero Posizione X, posizione Y,
velocità X, velocità Y
*/
Mat_<float> state(4, 1);

/*
processo con disturbo
*/
Mat processNoise(4, 1, CV_32F);

/*
misurazioni che sono locazioneX e locazioneY dell'oggetto
*/
Mat_<float> measurement(2,1);
measurement.setTo(Scalar(0));
```

```
23
24 // inserimento misura rilevata della coordinata X del mouse
25 KF.statePre.at<float>(0) = mouse_info.x;
26
27 // inserimento misura rilevata della coordinata Y del mouse
28 KF.statePre.at<float>(1) = mouse_info.y;
29
30 // impostazione di velocità di X = 0
31 KF.statePre.at<float>(2) = 0;
32
33 // impostazione della velocità di Y = 0
34 KF.statePre.at<float>(3) = 0;
35
36 // definizione della matrice di transizione
37 KF.transitionMatrix = *(Mat_<float>(4, 4) <<
38                   1,0,0,0,   0,1,0,0,  0,0,1,0,  0,0,0,1);
39
40 // setta matrice identità
41 setIdentity(KF.measurementMatrix);
42
43 // setta matrice identità
44 setIdentity(KF.processNoiseCov, Scalar::all(1e-2));
45
46 // setta matrice identità
47 setIdentity(KF.measurementNoiseCov, Scalar::all(1e-1));
48
49 // setta matrice identità
50 setIdentity(KF.errorCovPost, Scalar::all(.1));
51
52 /*---interno ciclo aggiornamento---*/
53
54 //calcola lo stato predetto
55 Mat prediction = KF.predict();
56
57 //crea il punto predetto
58 Point predictPt(prediction.at<float>(0),
59                     prediction.at<float>(1));
60
61 // ottiene la nuova misura
```

```
62 measurement(0) = x_reale;
63
64 // ottiene la nuova misura
65 measurement(1) = y_reale;
66
67 // crea punto stimato da misure di X e Y precedenti
68 Point measPt(measurement(0),measurement(1));
69
70 // inserisce nel vettore di punti rilevati
71 mousev.push_back(measPt);
72
73 // aggiornamento dello stato stimato dalla misura
74 Mat estimated = KF.correct(measurement);
75
76 // lo converte nel tipo Point
77 Point statePt(estimated.at<float>(0),
78                 estimated.at<float>(1));
79
80 /*
81 inserisce il punto appena creato
82 nel vettore di stato
83 */
84 kalmanv.push_back(statePt);
```

4.3 Approfondimenti algoritmici

In questa sezione viene presentato, a livello tecnico, l'approccio seguito nello sviluppo del metodo presentato in questa tesi. Le metodologie presentate nei paragrafi precedenti saranno utilizzate, in vari punti dell'applicazione. Dapprima verranno presentati, singolarmente, i diversi passaggi chiave, poi verrà contestualizzato il tutto in maniera organica.

4.3.1 Operazioni preliminari e Area Sensibile

Dati due frame consecutivi frame_{t-1} e frame_t, viene attuata, sin da subito, una comparazione. Alle due immagini viene applicato il filtro Gaussiano discusso in 4.2.1, principalmente usato per eliminare il rumore della camera di cattura.

Avendo ora frame_{t-1} e frame_t sfocati, si calcola una differenza assoluta, definita nella sezione 4.2.1, per rilevare solamente il foreground della scena; questo frame, chiamato foreground_t, rappresenta i blob in movimento nella scena. Per creare una informazione precisa, in cui ci siano solamente due tipologie di pixel (bianchi o neri), si converte il foreground_t in scala di grigi e si applica un filtro passa-basso, illustrato in 4.2.1.

Griglia di Rilevamento (front-end)

La metodologia proposta permette di individuare, in fase di configurazione, un'area che ha il compito di rilevare i movimenti. Se la griglia sarà grande come tutto il frame, allora si rileveranno i movimenti dell'intera scena inquadrata; se invece, la griglia, chiamata "sensibile"[10], è ridotta rispetto alle dimensioni del frame, allora i movimenti verranno rilevati solamente all'interno di quell'area. Nei parametri, è possibile impostare liberamente la *griglia sensibile*, e posizionarla a piacimento all'interno del frame catturato dalla camera. Risulta possibile, inoltre, settare anche la dimensione di righe e colonne della griglia.

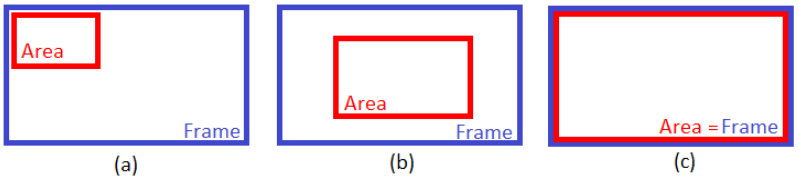


Figura 4.11: Il posizionamento dell'area sensibile può essere posizionato all'interno del frame (a e b), oppure corrispondente alla dimensione effettiva del frame (c)

La struttura interna della griglia sensibile è formata da celle sensibili. In particolare, ogni cella contiene:

- ID: identificatore cella, è un numero univoco all'interno della griglia;
- $(\text{init}_x, \text{init}_y)$: coordinate dell'angolo in alto a sinistra del rettangolo[11];

[10] Perchè sensibile ai movimenti.
[11] Informazioni utili a renderizzare a video l'area.

- $(\text{final}_x, \text{final}_y)$: coordinate dell'estremo in basso a destra del rettangolo[12];
- $(\text{center}_x, \text{center}_y)$: posizione del centro dell'area.

Occorrenze del movimento (back-end)

Questa struttura dati ha lo scopo di mappare i blob rilevati in movimento, descritti in 4.3.1, al fine di calcolare la stima della posizione della mano. La strutturazione è la seguente:

- matrice binaria di corrispondenza griglia, in corrispondenza uno-a-uno con le aree della griglia sensibile;
- vettore che tiene conto delle aree candidate [13];
- vettore dei candidati con assegnato un identificatore del cluster di appartenenza [14];
- vettore dei centri dei cluster identificati dopo la clusterizzazione presentata nella sezione 4.2.2.

4.3.2 Rilevare movimento

Per riuscire a calcolare un movimento significativo, vi sono dei passaggi intermedi che hanno lo scopo di filtrare ed eliminare movimenti secondari e isolati, al fine di ottenere delle occorrenze effettive del movimento della mano, o delle mani, che si desidera stimare.

Passo 1: cattura di tutti i movimenti

Questo è il primo passaggio, è quello più a grana grossa, il suo compito è quello di rilevare tutti i movimenti compiuti all'interno dell'area sensibile. Una volta ottenuto il foreground che caratterizza il movimento effettuato, calcolato in 4.3.1, ogni area sensibile attua una funzione che è in grado di ottenere il numero di

[12] Idem di sopra.
[13] Discusso nella sezione 4.3.2-Passo2.
[14] Discusso nella sezione 4.3.3.

pixel bianchi occorrenti al suo interno. Si ricorda che se tra $frame_{t-1}$e $frame_t$ non vi è movimento, si ottiene una immagine tutta nera, dovuta al filtro passa-basso, mentre se vi è stato del movimento, questo sarà rilevato dal calcolo della differenza assoluta tra i due frame, ottenendo così delle aree bianche. Nella figura 4.12 si esemplifica la funzione.

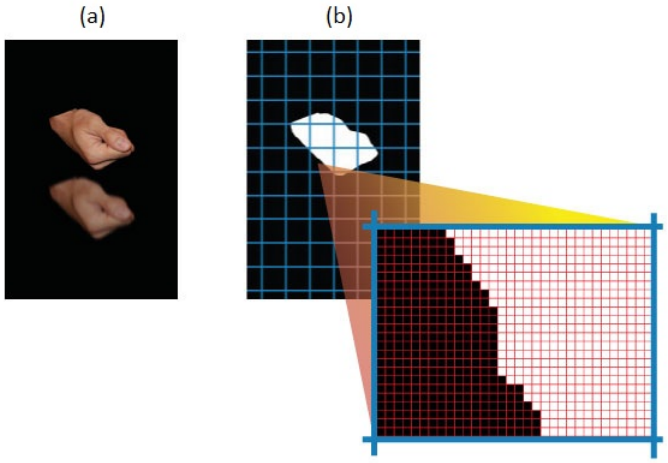


Figura 4.12: Rilevazione e controllo pixel bianchi di un'area sensibile. (a) movimento effettuato,(b) messo in evidenza il foreground rilevato nella griglia sensibile con zoom su di una singola area nella quale si andrà a considerare il numero effettivo di pixel bianchi

Il movimento della mano, effettuato nel primo riquadro, viene estrapolato e identificato nel secondo riquadro. Ad ogni area sensibile viene applicato un controllo nel quale si verifica se quell'area sta ad indicare un movimento oppure no. Nell'ingrandimento dell'immagine si effettua un conteggio dei pixel bianchi, determinanti il movimento, all'interno dell'area. Se il numero di pixel bianchi è maggiore di un certo valore di soglia[15], allora l'area identificata con il suo ID, andrà a mappare, nella matrice binaria di corrispondenza, la rilevazione di movimento. Questo processo si ripete per ogni area della griglia sensibile, ottenendo così, alla fine il popolamento della matrice binaria delle occorrenze. Lo pseudocodice è riportato di seguito:

[15]Impostabile come parametro di configurazione.

```
/*
comparazione dei due frame da oggetto chiamato
frameProcessor che si incarica di gestire i frame
ottenuti dalla webcam
*/
frameProcessor.comparaImmagini(frame1, frame2);

//cicla su tutte le aree
for ( int ID=0; ID < grigliaSensibile.size(); ID++){

        //Render dell'area sull'immagine
        grid.getArea(ID).renderOn(frame_corrente);

        /*si prende il numero di pixel bianchi che
      ci sono dopo la comparazione dell'immagine
      nell'area del quadrato, se sono più di AREA,
      allora il movimento è stato rilevato
    */
    if (numero_pixel_bianchi_in_area > THRESHOLD){
       //setta il valore della cella a 1
       occurrenceMatrix.setCell(ID,1);
    }
}
```

Passo 2: generazione dei candidati di posizione

A questo punto dell'algoritmo, si ha a disposizione una matrice binaria corrispondente al movimento appena rilevato. In figura 4.13, sono riportati degli esempi di movimento visti tramite matrice delle occorrenze creata al 4.3.2-Passo 1.

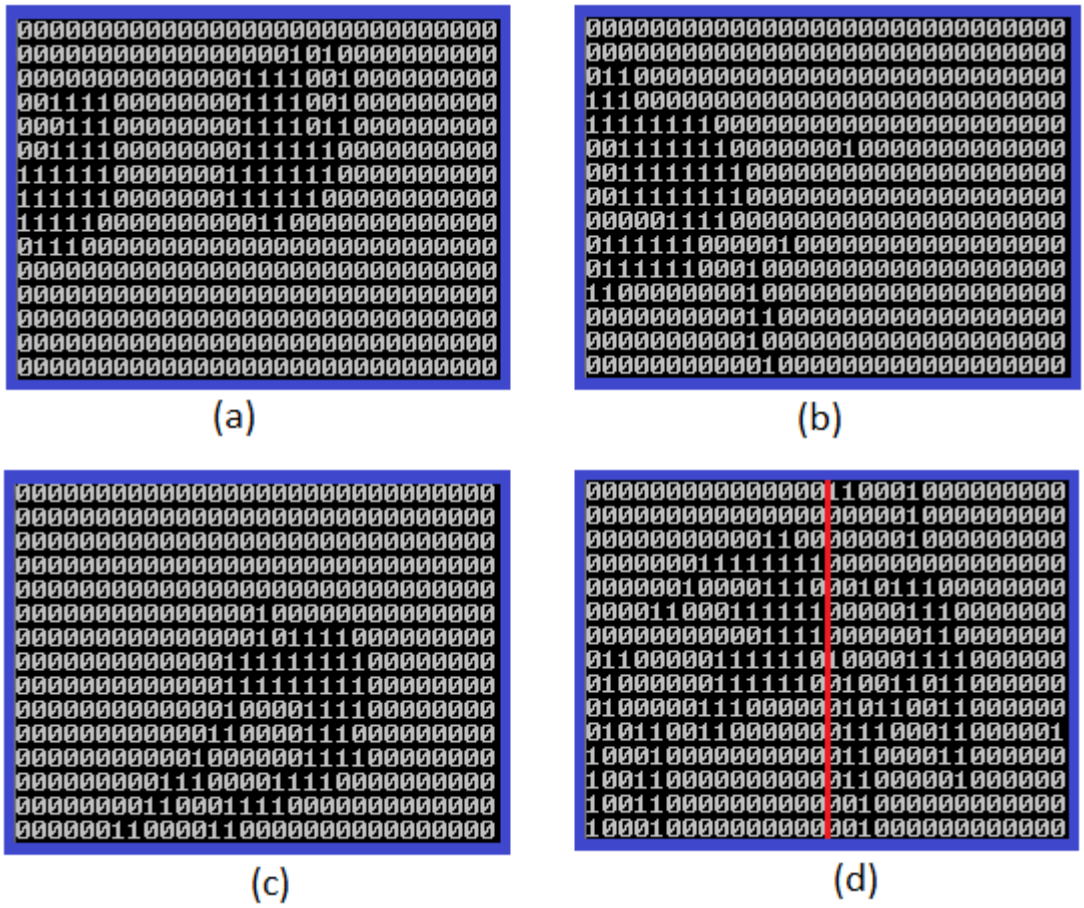


Figura 4.13: (a) movimento a due mani di fronte all'obiettivo; (b) a sinistra una sola mano in movimento da sinistra a destra, gli 1 sparsi singolarmente o molto piccoli, sono rumori; (c) movimento una mano dal basso-centrato vero alto-destra; (d) linea rossa solo immaginaria, a sinistra la mano, a destra rumore

Delle aree settate a 1, sicuramente ci saranno quelle più probabili in cui può essere mappata la posizione della mano, e altre improbabili. Il criterio di filtraggio è il seguente: volendo catturare il movimento maggiore, tra quelli rilevati, si devono considerare solamente le aree la cui occorrenza a 1 non sia isolata. Viene

applicato un filtraggio che elegge ogni area a probabile *posizione candidata* se, quest'ultima, è contenuta interamente tra aree anch'esse settate a 1.

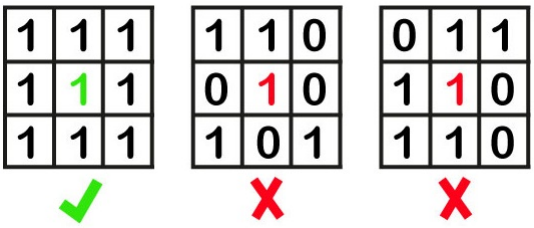


Figura 4.14: Considerando in esame sempre l'area con 1 in rosso, sono rappresentati esempi di accettazione e non, di candidati

Solamente nel caso in cui tutte le aree adiacenti siano a 1, l'area in esame entrerà a far parte del vettore delle aree candidate. Tutte le altre verranno scartate.

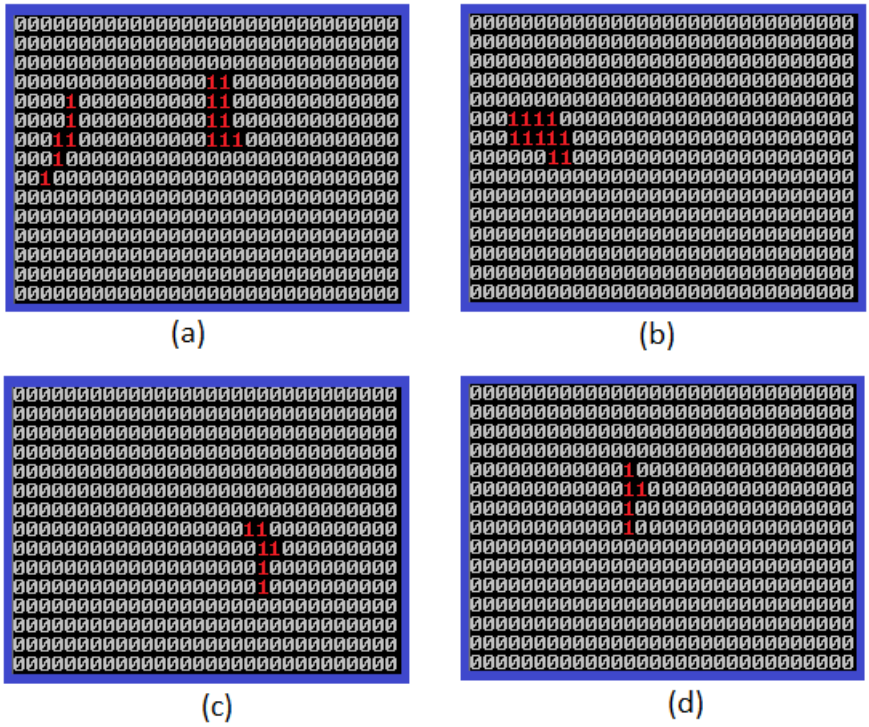


Figura 4.15: Schermate relative a 4.13, dopo aver effettuato il filtraggio. Queste sono le aree candidati ottenute

4.3.3 Cluster e loro centri

Una volta ottenuto il vettore delle aree candidate, calcolate precedentemente, si passa ora alla clusterizzazione di quest'ultime. Utilizzando l'algoritmo K-means, descritto in 4.2.2, con il valore $K = 2$, si vogliono creare due cluster. Una volta definiti, ad ogni area sensibile nel vettore candidati, viene assegnata la propria appartenenza ad un determinato cluster, identificato univocamente da una etichetta. C'è da precisare che, indipendentemente dal movimento di una mano, oppure due, questa clusterizzazione viene sempre eseguita. Successivamente sarà descritta una politica di discriminazione delle casistiche di movimento.

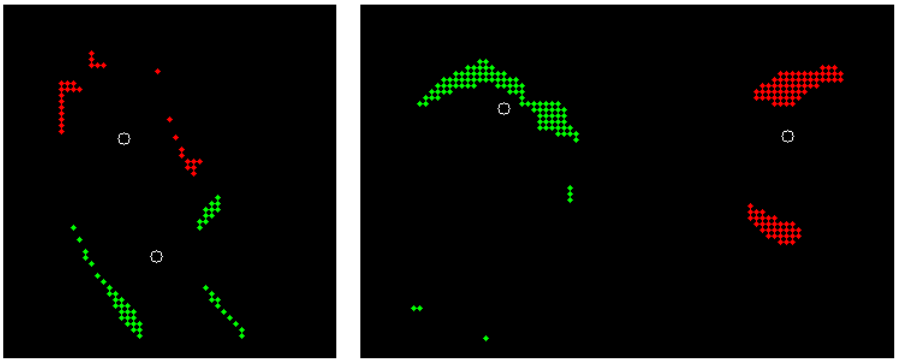


Figura 4.16: Suddivisione in cluster, da sinistra a destra: una mano, due mani chiuse che si allontanano. In bianco i centri dei cluster.

Con il calcolo dei cluster, vengono anche determinati, e memorizzati, i centri. A livello di codice la procedura K-means è così strutturata:

```
/*
Punti - immagine di input ottenuta dalla rappresentazione
        del vettore dei candidati
numeroCluster - numero di cluster
etichette - array che memorizza gli indici del cluster
            per ogni campione
criterioTerm - criterio di terminazione dell'algoritmo,
               in questo caso termina dopo 10-iterazione
               e il valore del parametro epsilon
               raggiunge 1.0
esecuzioni - Flag per specificare il numero di volte che
             viene eseguito l'algoritmo usando differenti
```

```
13              etichettature iniziali.
14 kmeans_pp_centers - Uso inizializzazione di
15                     Arthur and Vassilvitskii
16 centri - Matrice di uscita dei centri di cluster,
17          una riga per ogni centro di cluster
18 */
19
20 kmeans(Punti, numeroCluster, etichette,
21          criterioTerm, esecuzioni,
22          kmeans_pp_centers, centri);
```

4.3.4 Tracciamento e riconoscimento

Si hanno ora a disposizione i due cluster individuati e i relativi centri. Il passaggio che deve essere compiuto ora è quello di determinare se nella scena il movimento è compiuto da una mano, oppure da entrambe. Una metodologia di discriminazione dei due casi prevede l'utilizzo dei centri di cluster. Inizialmente viene calcolata la *distanza euclidea* tra i due centri. A questo punto, bisogna fare una considerazione: generalmente, quando si effettuano dei movimenti molto vicini alla camera (45/50cm), le aree di movimento rilevate e i relativi cluster, hanno dimensioni maggiori rispetto al caso in cui i movimenti siano effettuati ad una distanza maggiore (da 1,20m in poi). Per la determinazione di quante mani siano in movimento sulla scena, si confronta la distanza euclidea tra i due centri di cluster con un preciso valore di soglia. Questo valore però, vista la premessa sulla distanza del movimento rispetto alla camera, è bene che venga calcolato dinamicamente. Più in dettaglio, da parametri ricavati con test "ad hoc", si può affermare che per rilevare:

- due mani con camera vicina[16], la distanza euclidea deve essere maggiore di 35 pixel, con una approssimazione di +5 pixel;
- due mani con camera lontana[17], la distanza euclidea deve essere maggiore di 25 pixel, con una approssimazione di ±5 pixel;

[16] 45/50cm.
[17] Da 1,20 in su.

- una mano con camera vicina, la distanza euclidea deve essere minore di 35 pixel, con una approssimazione di +5 pixel;
- una mano con camera lontana, la distanza euclidea deve essere minore di 35 pixel, con una approssimazione di ±5 pixel;

Settare questo valore come un parametro di configurazione può essere oneroso, e vincolerebbe l'esecutore a non allontanarsi o avvicinarsi, nel corso della cattura delle immagini, dalla posizione iniziale. Per questo motivo il sistema include un calcolo dinamico di questo valore di soglia, in modo tale da fornire libertà di posizionamento a chi si appresta a utilizzare l'applicazione.

```
// ottieni dimensione cluster mano sinistra
dimClusterManoSx = getDimClusterSx();

// ottieni dimensione cluster mano destra
dimClusterManoDx = getDimClusterDx();

//dimensione_media = parametro determinante
                     la dimensione media di un cluster

if( (dimClusterManoSx > dimensione_media)&&
       (dimClusterManoDx > dimensione_media)){
             //probabilmente si è vicini dalla camera
             soglia = 35;
}else{
        //probabilmente si è lontani dalla camera
        soglia=25;
}

//discriminante
if(distanzaEuclidea < soglia){
        return una_mano_rilevata;
}
else{
   return due_mani_rilevate;
}
```

Avendo una previsione del numero di mani sulla scena, è il momento di calcolare la probabile posizione assunta. Supponiamo il caso che sia stata prevista

la presenza di una sola mano. Viene chiamata una funzione che ha il compito di eleggere l'area, riferita alla griglia sensibile di 4.3.1, in cui si trova la mano, appartenente al corrispondente cluster eletto. La politica della funzione prevede che venga restituita l'area più in alto nella griglia, sempre appartenente al cluster identificato come eletto. In figura 4.17 si vedono i movimenti dell'immagine 4.13 con una mano.

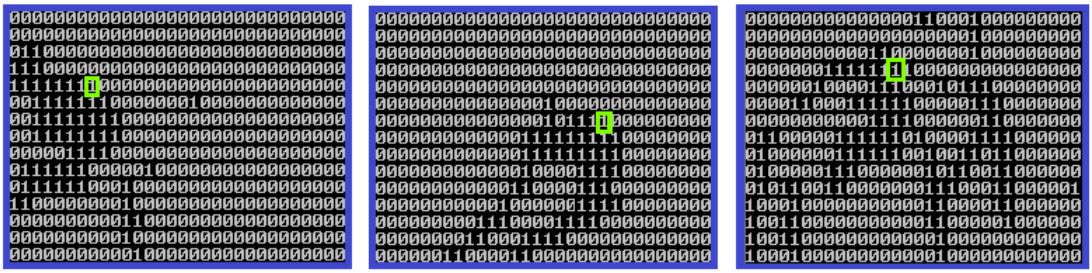


Figura 4.17: Posizioni rilevate riferite ai movimenti in 4.13

Se il discriminante ha previsto la presenza di due mani, per ogni cluster individuato, la funzione corrispondente restituisce l'area sensibile, relativa al cluster, in posizione più alta, nella griglia sensibile.

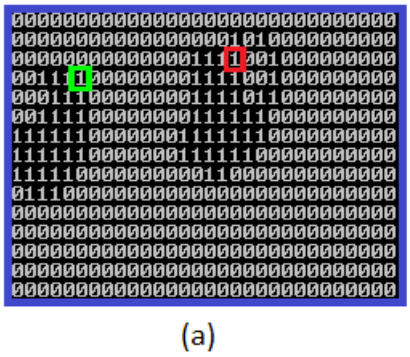


Figura 4.18: Stima posizione delle due mani, in riferimento a 4.13

Indipendentemente dal fatto che ci sia una mano o due, una volta ottenura l'area eletta, si estrapola il valore in pixel delle sue coordinate. Queste coordinate però, non vengono immediatamente visualizzate a video, in quanto, con questa tecnica, emergono delle discontinuità nel tracciare i movimenti tra frame

successivi, e quindi nella stima la giusta posizione; per ovviare a questo disturbo il sistema incorpora un filtro di Kalman, descritto in 4.2.3, che ha il compito di tracciare una traiettoria coerente e fluida lungo la quale è effettuato un particolare movimento.

```
//controllo se si deve catturare una sola mano o entrambe

if(unaMano){
 // ritorna ID dell'area individuata
 posizione = matriceOccorrenze.posizioneMano();

 /*
 dalla posizione ottieni manoX e manoY
 che sono le coordinate
 */
 puntoKalman = kalmanSingolaMano.Filtra(manoX, manoY);

 // memorizza posizione puntoKalman

 /*
 fatto il filtro di Kalman e ottenute
 le coordinate calcolate, queste sono passate
 alla procedura mapHand
 */
 (griglia.Area(posizione)).mappaMano(frame,
                                     puntoKalman.x,
                                     puntoKalman.y);

}else{
 // ritorna X,Y dell'area individuata mano sinistra
 areaManoSinistra = matriceOccorrenze.ottieniManoSx();

 // ritorna X,Y dell'area individuata mano destra
 areaManoDestra = matriceOccorrenze.ottieniManoDx();

 // ottieni posizioni di mano in posSx e posDx

 puntoKalmanSx=kalmanDueMani_sx.Filtra(manoSinistra_X,
                                       manoSinistra_Y);

```

```
36  puntoKalmanDx=kalmanDueMani_dx.Filtra(manoDestra_X,
37                                        manoDestra_Y);
38
39  // mano sinistra
40  (griglia.Area(posSx)).mappa(frame,puntoKalmanSx.x,
41                                    puntoKalmanSx.y);
42
43  // mano destra
44  (griglia.Area(posDx)).mappa(frame,puntoKalmanDx.x,
45                                    puntoKalmanDx.y);
46 }
```

Riconoscimento

Per riconoscere un determinato movimento, è necessario sapere a priori come sarà riproducibile, e soprattutto rilevabile, dal sistema. Si hanno a disposizione le coordinate delle mani in movimento sulla scena e quindi si è in grado di rilevare molte tipologie di movimenti che sono rappresentabili avendo le mani davanti al corpo. In 3.2, sono stati discussi degli esempi e delle casistiche di riconoscimento applicato ad un caso reale. Al fine di riconoscere un movimento, sono disponibili due approcci diversi; a seconda del movimento specifico da effettuare, può risultare più efficace l'uno o l'altro, oppure, in alternativa, utilizzare un approccio ibrido che unisca tutti e due i modelli. La scelta della metodologia di riconoscimento è un blocco separato e tutto quello detto sino ad ora rimane valido. Se nel lavoro futuro, si presentasse la necessità di cambiare il metodo di riconoscimento, questo non andrebbe a modificare in alcun modo l'estrazione delle informazioni di movimento e posizionamento.

Checkpoint di Aree

Questa metodologia si riferisce a quanto descritto in [34], più precisamente al modulo di riconoscimento del gesto. In questo approccio, i gesti vengono rappresentati geometricamente. Caratteristica principale è quella di avere un punto di partenza, una traiettoria (o percorso) da seguire ed un punto di arrivo. Il punto di partenza e di arrivo sono delle porzioni attive nello schermo che vengono

attivate da un componente che si occupa di attivarle o disattivarle, a seconda del movimento da riconoscere. La traiettoria può essere controllata, per esempio, inserendo tra il punto di inizio ed il punto di fine, delle aree di controllo (checkpoint) in cui è previsto il passaggio in un determinato istante. Un altro fattore che viene preso in considerazione è il tempo, questo perchè molti gesti, oltre che alla traiettoria, sono guidati dal tempo: ci sono gesti veloci e gesti lenti.

In questo approccio, non esistono fasi di training, viene richiesta solamente una buona rappresentazione del movimento, in modo tale da gestire il posizionamento di queste aree di checkpoint. Quando la mano arriva al punto finale, il movimento è riconosciuto. Può anche capitare che il gesto sia svolto in modo erroneo; in questo caso, o può essere rilevato prima del punto finale su segnalazione di checkpoint, oppure su scadenza di un tempo tollerabile di esecuzione del gesto. Di seguito è riportato il listato base ripreso da [34]:

```
if(posizioneMano == areaIniziale){
   avvioTimer();
   while(posizioneMano != areaFinale){
      if(posizioneMano != traiettoria o tempoScaduto){
         errore();
      }
   }
   gestoRiconosciuto;
}
```

1$ - Riconoscitore per prototipi interfacce utente

Questo approccio, proposto da [50], usa un riconoscitore che non utilizza training, librerie o toolkit. Si basa anch'esso sulla geometria e trigonometria, e richiede poche linee di codice per definire e riconoscere un gesto a partire da una traiettoria. Include la possibilità di settare e personalizzare i parametri di rotazione, scala e invariante di posizione.

Un gesto, compiuto dall'utente, è visto come una serie di *punti candidati* C; lo scopo è quello di determinare, nell'insieme di *punti template* T_i, quale di questi rappresenta meglio il tracciato di punti in input. I punti candidati vengono campionati ad una frequenza determinata dalla rilevazione hardware e

software, questo fatto indica che C e T_i raramente saranno facili da comparare immediatamente.

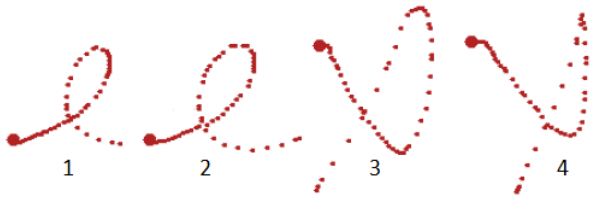


Figura 4.19: I tracciati da 1 a 4, sono parte di un unico gesto da riconoscere, ma utilizzare una comparazione punto-punto non è sufficiente

Le caratteristiche che il riconoscitore deve avere sono le seguenti:

- essere resistente alle variazioni di campionamento causate dalla velocità di movimento o rilevamento;
- supportare la rotazione e la scala;
- non necessitare di tecniche matematiche avanzate (ad esempio, inversioni di matrici, derivate, integrali), in modo tale da essere veloci;
- essere scritto in poche linee di codice;
- essere ragionevolmente veloce per scopi interattivi (senza ritardi);
- consentire agli sviluppatori di inserire nuovi template di gesti, i quali richiedano solamente un esempio di esecuzione;

Esistono però delle limitazioni. Questo sistema si basa sul riconoscimento di template geometrici, il che significa che i punti di traiettoria individuati, vengono confrontati con i template memorizzati precedentemente, e in output si ottiene la corrispondenza più vicina, in due dimensioni, sullo spazio euclideo. $1 non utilizza il tempo, quindi gesti basati sulla velocità non possono essere individuati. Si possono definire nuovi modelli che catturano la variazione di un gesto: utilizzando un nome unico, per esempio "nodo" (fig.4.19), vengono riconosciuti diverse esecuzioni di uno stesso gesto che dovrebbe, appunto, rappresentare un nodo.

Questo approccio di riconoscimento fornisce la rotazione (opzionale), la scala e l'invarianza sulla posizione, offrendo una precisione del 99% con l'ausilio di pochi modelli. Non sono richieste complesse procedure matematiche, dal momento che utilizza un approccio di classificazione statistica.

Capitolo 5

Analisi dei Risultati

I movimenti presi in esame, a titolo dimostrativo, sono relativi ai casi d'uso presenti nel capitolo 3:

- Martellare la punta della scarpa;
- Cucire le pelli;
- Intagliare la suola;
- Annodare, tramite fili, la pelle alla suola.

Queste tipologie di movimento richiedono, alternativamente, l'uso di una o due mani per eseguire il gesto. La strutturazione dell'applicazione prevede che, all'inizio, siano calcolati i due centri dei cluster e, successivamente, venga determinato il numero di mani in movimento sulla scena. Nel discutere i dati ottenuti durante la fase di test, vi sono diversi parametri da riportare e tenere in considerazione:

1. la distanza della camera rispetto all'esecutore;
2. la percentuale di hit[1] della mano, basata sul numero di frame;
3. la distanza euclidea utilizzata dal discriminante per decidere il movimento di una o due mani;

[1]Match corretto della posizione della mano.

4. l'unicità del punto renderizzato a video: se il discriminante sbaglia la previsione verranno erroneamente stimate due posizioni diverse, quando in realtà ne servirebbe solo una, oppure, stimerebbe una sola posizione quando invece ne servirebbero due.

La frequenza di aggiornamento della camera è di 30fps, in ogni prova.

Caso d'Uso 1

Il primo gesto preso in esame consiste nel compiere un colpo di martello, dall'alto verso il basso. In figura 5.1, è riportata un'esecuzione del gesto: la parte sinistra rappresenta l'esito di stima, positivo (hit) o negativo (miss), per ogni frame; la parte destra schematizza il movimento partendo da una posizione iniziale e raggiungendo la posizione finale.

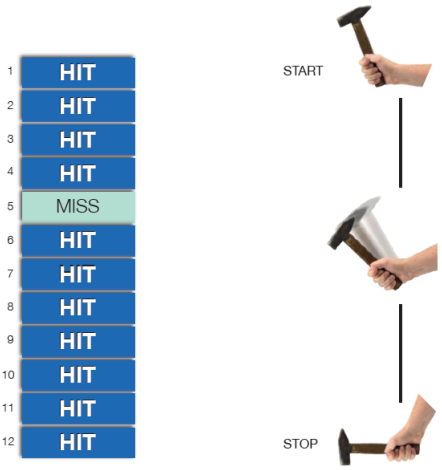


Figura 5.1: Sequenza presa in considerazione nella fase di test

Nella figura 5.2, sono riportati i risultati ottenuti. Ad una particolare distanza della camera è associata una distanza di cluster ottimale per discriminare la presenza di una mano o due. La percentuale di hit corretto indica la precisione di stima dell'algoritmo. Il punto unico è un parametro che stima la necessità di mappare solamente un punto a video e non due, quando si ha a che fare con movimenti principalmente compiuti con una mano. Infine, la stima della distanza

corretta, indica quante volte l'algoritmo è riuscito a stimare la distanza ottimale dai due centri di cluster.

Distanza Euclidea ottimale (pixel)	Distanza camera (cm)	Hit corretto (%)	Punto unico (%)	Stima distanza corretta (%)
35 (± 5)	50	94,34	92,45	90,57
25 (+ 5)	120	92,59	100	98,00

Figura 5.2: Tabella di riepilogo del gesto con martello

Caso d'Uso 2

Il secondo gesto testato prevede il fatto di iniziare da un punto scelto dall'utente, e di proseguire con un andamento circolare sino ad arrivare ad un punto finale che fa terminare il movimento; in questo caso i due punti iniziali e finali sono molto simili, in termini di posizione. La figura 5.3 mostra la stima corretta (hit) o sbagliata (miss) attuata dall'algoritmo per ogni frame; a destra invece, è riporata la schematizzazione del movimento effettuato.

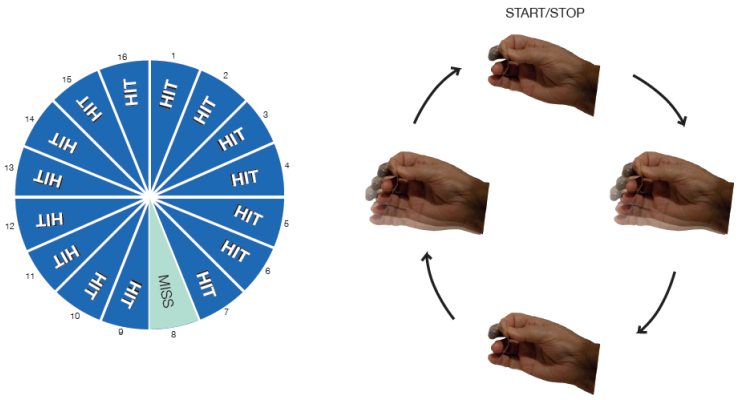


Figura 5.3: Sequenza schematizzata nella fase del cucito

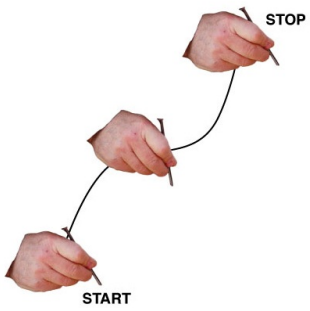
Nella figura 5.4, sono riportati i risultati ottenuti.

Distanza Euclidea ottimale (pixel)	Distanza camera (cm)	Hit corretto (%)	Punto unico (%)	Stima distanza corretta (%)
35 (± 5)	50	96,22	92,45	100
25 (+ 5)	120	98,21	100	98,74

Figura 5.4: Tabella riassuntiva del gesto del cucito

Caso d'Uso 3

Il terzo gesto testato riguarda il tracciamento di una mano intenta ad intagliare, con uno scalpello molto piccolo, la suola di una scarpa. Il movimento parte dal tacco e procede sino alla punta, seguendo la forma della suola. In figura 5.5, nella parte sinistra è rappresentata la schematizzazione del gesto, mentre a destra è riportata una tabella che illustra, per ogni frame, sia la percentuale di successo nello stimare la posizione della mano, sia l'unicità del punto stimato.



N° FRAME	HIT POINT?	UNIQUENESS POINT
1	MATCH	YES
2	MATCH	YES
3	MATCH	YES
4	MATCH	YES
5	MATCH	YES
6	MISS	NO
7	MATCH	YES
8	MATCH	YES
9	MATCH	YES
10	MATCH	YES

Figura 5.5: Schematizzazione del gesto dell'intaglio

Anche in questo caso, si tratta di un movimento principalmente compiuto con una mano. Come negli altri due casi precedenti, la stima della distanza corretta, indica quante volte l'algoritmo è riuscito a stimare la distanza ottimale dai due centri di cluster.

Nella tabella in figura 5.6, sono rappresentati i risultati ottenuti.

Distanza Euclidea ottimale (pixel)	Distanza camera (cm)	Hit corretto (%)	Punto unico (%)	Stima distanza corretta (%)
35 (± 5)	50	100	93,67	93,67
25 (+ 5)	120	95,77	100	100

Figura 5.6: Tabella riassuntiva del gesto dell'intaglio

Caso d'Uso 4

L'ultimo gesto considerato riguarda il movimento delle due mani. L'obiettivo è quello di tracciare contemporaneamente la mano sinistra e la mano destra durante l'operazione di unione della pelle alla suola della scarpa, attraverso nodi e lacci. In figura 5.7 è riportata una tabella centrale che indica se per ogni frame, la stima della posizione di entrambe le mani è avvenuta o meno in maniera corretta [2]. A sinistra e a destra della tabella è schematizzato il movimento compiuto dalle due mani.

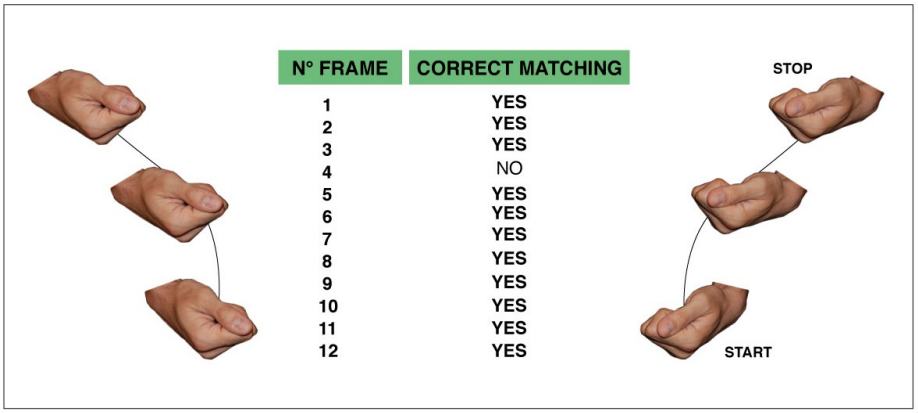


Figura 5.7: Schematizzazione del gesto con due mani

Nella tabella di figura 5.8, questa volta non compare il parametro di unicità del punto in quanto il movimento prevede di mappare due posizioni, le quali, saranno considerate corrette solamente se entrambe sono stimate bene; il risultato è riportato dal parametro hit corretto. La stima di distanza corretta si riferisce

[2]Se fosse stimata una posizione sola con le mani distanti, la stima sarebbe errata.

alla valutazione attuata dinamicamente dall'algoritmo, circa la distanza euclidea usata come valore di soglia.

Distanza Euclidea ottimale (pixel)	Distanza camera (cm)	Hit corretto (%)	Stima distanza corretta (%)
35 ($\pm$ 5)	50	97,22	100
25 (+ 5)	120	100	94,74

Figura 5.8: Tabella relativa al movimento di due mani

Analisi

Questi quattro gesti, non sono che un sottoinsieme molto ristretto di quelli che si potrebbero compiere. La sceltà è stata guidata dai movimenti effettuati durante la lavorazione di scarpe di alta moda, nel constesto descritto nella sezione 3.2. Sono stati presi in considerazioni movimenti compiuti con una o due mani, per fornire una panoramica completa sul discriminante del numero di mani sulla scena. In tutti i casi si è ottenuta una buona percentuale di stima anche a velocità di circa 500ms, in media.

Per quanto riguarda il **caso d'uso 1**, ad una distanza piccola si ottengono buone percentuali di matching e stima di unicità del punto, con una correttezza di scelta del valore di soglia del 90% circa. Ad una distanza più elevata, si nota un lieve calo di hit corretti, restando comunque sopra la soglia del 92%. Da notare come però, a distanza più elevata la previsione del valore di soglia è intorno al 98% con una stima di unicità del punto praticamente esatta.

Nel **caso d'uso 2**, si riscontra una percentuale di hit corretti molto buona; in questo caso la stima da vicino è un po' peggiore di quella da distante, restando comunque ad una differenza del 2% e sopra il 95% di correttezza. La stima del punto unico è buona sia a distanza piccola che grande[3], avendo in entrambi i casi una percentuale di stima della distanza euclidea corretta intorno al 98%.

Nel **caso d'uso 3**, si ritorna ad avere una percentuale di hit corretto maggiore ad una distanza ravvicinata, che si discosta del 5% sulla stima applicata a distanza

[3]Sempre restando nei limiti di inquadratura.

maggiore. Anche in questo frangente però, l'unicità del punto risulta esatta nel secondo caso. Rimangono comunque valori molto buoni in quanto rimangono nell'intorno del 90% di efficacia.

Nell'**ultimo caso d'uso**, persiste sempre un buon matching sia per quanto riguarda le stime di posizione delle due mani in scena, sia per quanto concerne il valore di soglia, stabilito dinamicamente, per classificare la distanza tra i due centri di cluster.

Conclusioni

Con questo elaborato, si è voluto rappresentare un nuovo approccio nel campo del riconoscimento di gesti e movimenti. Le metodologie presentate nel capitolo 2, formano un insieme di approcci base, i quali, proprio come un middleware, sono alla base della creazione di estensioni, customizzazioni o addirittura nuove applicazioni. Si pensi ad esempio alla skin detection, dove per migliorarla si potrebbero inserire contributi provenienti dalla statistica, come le catene di markov nascoste, oppure alla metodologia di optical flow, in cui può essere interessante inserire approcci provenienti dal machine learning. L'intento è quello di portare un contributo a questa struttura di base, in modo da essere utilizzata per creare nuove estensioni. La tipologia di gesti presa in esame è quella dinamica, composta da una serie di posizioni, rappresentate da fotogrammi in successione e scandite dal tempo, mentre lo spazio di movimento è prevalentemente limitato. Il sistema gestisce e codifica gesti fini, non necessariamente ampi, compiuti nelle strette vicinanze del corpo, ad esempio nella zona del torace. Si è cercato di codificare piccoli movimenti, anche molto ridotti, cercando di catturare il movimento "puro", presente nell'esecuzione di un gesto. Un vincolo significativo è stato quello di non adottare l'uso di marcatori o dispositivi indossabili perchè, trattandosi di piccoli movimenti, l'esecutore può riscontrare difficoltà o disturbi nell'eseguire un lavoro con addosso guanti o indumenti invasivi e ingombranti.

Partendo dalla partecipazione al progetto [51, 52], si è voluto creare un nuovo approccio in cui l'interazione invisibile tra l'utente e il sistema riguardasse non più il riconoscimento di gesti ampi svolti all'interno di un ambiente vincolato[4], ma bensì il riconoscimento di piccoli gesti, in un ambiente dove l'illuminazione

[4]Come per esempio un sfondo uniforme o un settaggio di luci predefinito.

può essere variata e non vi è presente uno sfondo predeterminato.

Gli ambiti applicativi sono numerosi, in questo lavoro si è scelto ed adottato l'ambito artigianale perchè il sapere artigiano, frutto di esperienza e tecnica appresa nel corso del tempo, è particolarmente ricco di gestualità, anche meticolosa, molto fine e limitata nello spazio. Testare in questo ambito l'approccio proposto, è sembrato un buon campo di prova. Nello specifico, si è sviluppato un prototipo non invasivo che, solamente con un dispositivo hardware di acquisizione video e particolari algoritmi, riesce a codificare e mantenere la conoscenza dei gesti artigianali. Sono stati considerati quattro casi d'uso che rappresentano dei gesti significativi, coinvolgendo una e due mani nell'esecuzione. In tutti i casi, si è riscontrata una buona percentuale di stima della posizione e una buona resa del tracking della mano, ad una velocità media di circa $500ms$.

Partendo dalla rilevazione di movimento, e successivamente dalla clusterizzazione di tali dati, la determinazione del numero di mani nella scena è determinata dall'algoritmo, il quale, basandosi sulla dimensione dei blob rappresentanti il movimento rilevato, stabilisce se adottare un valore di soglia per la distanza più ampio o più basso, da confrontare poi con i centri dei cluster. Posizionandosi ad una distanza ravvicinata dalla camera (circa $50cm$), da test preliminari è emerso un valore di soglia della distanza euclidea pari a 35 pixel con un'approssimazione di ± 5 pixel. Con questa configurazione si riesce a determinare un matching intorno al 94% con una predizione corretta del numero di mani intorno al 90%. Allontanandosi a circa $120cm$, la determinazione del numero corretto di mani migliora di circa il 5% a scapito però di una piccola riduzione sull'accuratezza della stima di posizionamento.

Dai primi risultati ottenuti, si nota un buon grado di stima sia vicino che distante dalla camera. Ovviamente, allontanandosi troppo, si rischia di non rilevare piccoli movimenti della mano e ridurre in maniera significativa l'accuratezza della stima della posizione.

Con questo lavoro di tesi, si è delineata una base per un nuovo approccio di rilevazione del movimento; il prossimo passo è quello di migliorare ulteriormente il discriminante del valore di soglia della distanza euclidea tra cluster, compiendo ulteriori test in differenti casi d'uso. Allo stato attuale, il sistema stabilisce solamente un punto di posizionamento della mano. Partendo da questo, si può

ottenere la stima della posizione anche delle dita: così facendo la posizione della mano stimata funge da perno per una generazione di ROI in cui è possibile studiare, con appositi algoritmi e tecniche, la stima della posizione delle dita.

Altro aspetto da considerare è la determinazione della posizione nello spazio in tre dimensioni. Risulta interessante, partendo dalla stima 2D restituita dall'algoritmo, ottenere anche la coordinata di profondità con l'ausilio di una seconda camera posta a una determinata distanza dalla prima, in modo da ottenere una visione stereoscopica. Utilizzando algoritmi di corrispondenza stereo, possono essere considerate due (o più) immagini in scala di grigio come input, e quindi può essere prodotta una mappa di disparità[5] $d\,(x, y)$ per ogni pixel, in genere memorizzata come immagine in scala di grigi. Questa è inversamente proporzionale alla profondità, ed è possibile definire una tripla (x, y, d) ottenendo una posizione in tre dimensioni. Se si riesce ad ottenere una buona approssimazione di profondità, è interessante applicare questo algoritmo per rilevare anche movimenti fini.

[5]Il termine di disparità era usato inizialmente per descrivere le differenze, in 2D, stimate dalla camera di destra e quella di sinistra.

Bibliografia

[1] Roccetti, M., Marfia, G. and Zanichelli, M., 2010. *The Art and Craft of Making the Tortellino: Playing with a Digital Gesture Recognizer for Preparing Pasta Culinary Recipes.* ACM Comp. Enter. 8, 4

[2] Mitra, S., Acharya, T., *Gesture Recognition: A Survey.* IEEE Transactions on Systems, Man and Cybernetics - PART C, 2007, 37-3, 311-324

[3] Kakumanu, P., Makrogiannis, S., and Bourbakis, N., *A survey of skin-color modeling and detection methods*, Pattern Recognition, 2007, pp. 1106-1122

[4] Jones, Michael J. and Rehg, James M., *Statistical Color Models with Application to Skin Detection*, Int. J. Computer Vision, January 2002, 46-1, 81-96

[5] Elgammal, A., Muang, C., Hu, D., *Skin Detection - A Short Tutorial,* Encyclopedia of Biometrics R. U. Department of Computer Science, 2009

[6] Yang, M., Ahuja, N.: *Gaussian mixture model for human skin color and its application in image and video databases.* In Proc. of the SPIE: Conference on Storage and Retrieval for Image and Video Databases (SPIE 99). Volume 3656. (1999), 458-466

[7] Horn,B. K. P., and Schunck,B. G., *Determining optical flow*, Artificial Intelligence 17 (1981), 185–203

[8] Barron, J.L., Fleet, D.J., and Beauchemin, S., *Performance of optical flow techniques,* International Journal of Computer Vision, 12(1), 43-77, 1994

[9] Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. , *Pyramid methods in image processing*, RCA Engineer 29 (1984): 33–41

[10] Rosenfeld, A., *Some Uses of Pyramids in Image Processing and Segmentation*, Proceedings of the DARPA Imaging Understanding Workshop (pp. 112–120), 1980

[11] Burt,P. J., and Adelson, E. H., *The Laplacian pyramid as a compact image code*, IEEE Transactions on Communications 31 (1983): 532–540

[12] Pérez, P., Hue, C., Vermaak, J., and Gangnet,M., *Color-based probabilistic tracking*, Springer-Verlag Berlin Heidelberg, pp. 665-670, 2002

[13] Fukunaga,K., *Introduction to Statistical Pattern Recognition*, Boston: Academic Press, 1990

[14] Bradsky, G., Kaehler, A., *Learning OpenCV,* O'Reilly Media, settembre 2008

[15] Bradski, G. R., *Computer Vision Face Tracking For Use in a Perceptual User Interface*, Microcomputer Research Lab, Santa Clara, CA, Intel Corporation, Intel Technology Journal Q2 1998

[16] Juergen,G., Carsten,S., de Aguiar, E., Theobalt, C., Rosenhahn, B., Seidel, H., *Motion Capture using Joint Skeleton Tracking and Surface Estimation*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009

[17] Bouguet, J., *Pyramidal implementation of the lucas kanade feature tracker*, Intel Corporation Microprocessor Research Labs, 2000

[18] Starner, T., Weaver, J., Pentland, A., *Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video*, IEEE Trans. Pattern Anal. Mach. Intell., December 1998, 20-12, pp. 1371-1375

[19] Wang, R.Y. and Popovic, J., 2009. *Real-time hand-tracking with a color glove.* ACM Trans. Graph. 28, 3

[20] Cao,Z., Yin, Q., Tang, X., and Jian, S., *Face recognition with learning-based descriptor*, In Proc. Computer Vision and Pattern Recognition, 2010

[21] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A., *Real-time human pose recognition in parts from single depth images*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011

[22] Deutscher,J., and Reid, I., *Articulated body motion capture by stochastic search*. IJCV, 61(2), 185–205, 2005

[23] Agarwal, A., and Triggs,B., *Recovering 3D human pose from monocular images*. IEEE T-PAMI, 28(1), 44-58, 2006

[24] Grest, D., Woetzel, J., and Koch, R., *Nonlinear body pose estimation from depth images*, In In Proc. DAGM, 2005

[25] Knoop, S., Vacek, S., and Dillmann, R., *Sensor fusion for 3D human body tracking with an articulated 3D body model*, In Proc. ICRA, 2006

[26] Zhu, Y., and Fujimura. K., *Constrained optimization for human pose estimation from depth sequences*, In Proc. ACCV, 2007

[27] Moeslund, T., Hilton, A., and Krüger, V., *A survey of advances in vision-based human motion capture and analysis*, CVIU, 2006

[28] Poppe, R. , *Vision-based human motion analysis: An overview*, CVIU, 108, 2007

[29] Comaniciu, D., and Meer, P., *Mean shift: A robust approach toward feature space analysis*, IEEE Trans. PAMI, 24(5), 2002

[30] Urtasun, R., and Darrell, T., *Local probabilistic regression for activity-independent human pose inference*, In Proc. CVPR, 2008

[31] Doucet, A., De Freitas, N., Gordon, N.J., *Sequential Monte Carlo Methods in Practice*. Springer 2001

[32] Isard, M., and Andrew, B., *CONDENSATION—Conditional Density Propagation for Visual Tracking*, International Journal of Computer Vision 29(1), 5–28 (1998)

[33] DeCarlo, D., Metaxas, D., *Optical Flow Constraints on Deformable Models with Applications to Face Tracking*, International Journal of Computer Vision, 38(2), pp. 99-127, July 2000

[34] Roccetti, M., Marfia, G., *Recognizing Intuitive Pre-defined Gestures for Cultural Specific Interactions: An Image-based Approach*, Proc. 3rd IEEE International Workshop on Digital Entertainment, Networked Virtual Environments, and Creative Technology (DENVECT'11) - 8th IEEE Communications and Networking Conference (CCNC 2011), Las Vegas (USA), IEEE Communications Society, January 2011

[35] Rosner, D.K., and Ryokai, K., 2009, *Reflections on craft: probing the creative process of everyday knitters*, In Proc. 7th ACM Conf. on Creativity and Cognition, Berkeley, 195-204

[36] Niedderer, K., and Townsend, K. *Craft research: Joining emotion and knowledge*, Design and emotion conference 2010: Blatantly blues (Proceedings). Chicago: IIT Institute of Design

[37] Torrey, C., Churchill, E., McDonald, D. W., *Learning How: The Search for Craft Knowledge on the Internet*, CHI 2009, ACM Press, Boston (2009)

[38] Marfia, G., Roccetti, M., Matteucci, G., Marcomini, A., *Technoculture of Handcraft: Fine Gesture Recognition for Haute Couture Skills Preservation and Transfer in Italy*, Proc. 39th ACM International Conference and Exhibition on Computer Graphics and Interactive Techniques Posters, (SIGGRAPH 2012), Los Angeles, August, 2012

[39] Marfia, G., Roccetti, M., Marcomini, A., Bertuccioli, C., Matteucci, G., *'Reframing Haute Couture Handcraftship: How to Preserve Artisans' Abilities with Gesture Recognition,* submitted for publication, July 2012

[40] Intel Research Lab, *Opencv: Open computer vision library*, http://opencv.willowgarage.com/wiki/

[41] Bradski, G., and Kaehler, A., *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, Inc., 2008, ISBN: 978-0-596-51613-0

[42] Laganière, R., *OpenCV 2 Computer Vision Application Programming Cookbook*, Packt Publishing, 2011, ISBN: 978-1-849-51324-1

[43] Sezgin, M., and Sankur, B., *Survey over image thresholding techniques and quantitative performance evaluation*, Journal of Electronic Imaging 13 (2004): 146–165

[44] Vassilvitskii, S., and Arthur, *k-means++: the advantages of careful seeding*, Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007

[45] Kalman, R. E., *A New Approach to Linear Filtering and Prediction Problems*, Transaction of the ASME, Journal of Basic Engineering, 1960, pp. 33-45

[46] Hamerly, G. and Elkan, C., *Alternatives to the k-means algorithm that find better clusterings*, Proceedings of the eleventh international conference on Information and knowledge management (CIKM), 2002

[47] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., Wu, A. Y., *An efficient k-means clustering algorithm: Analysis and implementation*, IEEE Trans. Pattern Analysis and Machine Intelligence 24: 881–892, 2002

[48] Baudel, T., Beaudouin-Lafon, M., Braort, A., and Teil, D., *An interaction model designed for hand gesture input*, L.R.I. CNS URA, Universite de Paris-Sud - France, 1992

[49] Grewal Mohinder, S., Andrews Angus, P., *Kalman filtering Theory and Practice*, Upper Saddle River, New York, Prentice Hall

[50] Wobbrock, J. O., Wilson, A. D., and Li, Y. (2007) , *Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes*, Proceedings of the ACM Symposium on User Interface Software and Tecnology (UIST '07), Newport Rhode Island (October 7-10, 2007), New York: ACM Press, pp. 159-168

[51] Giles, J., *One per Cent: Video Games Teaches You To Make the Perfect Tortellini*, New Scientist, Jan. 2011, accessed online July 4th, 2012, http://www.newscientist.com/blogs/onepercent/2011/01/computer-game-that-teaches-you.html

[52] Mitzman, D., *Italian Hi-Tech Software Teaches Perfect Pasta Skills*, BBC News, June 2011, accessed online July 4th, 2012, http://www.bbc.co.uk/news/world-europe-13856559

Ringraziamenti

A questo punto direi che è ora di ringraziare un po' di persone.

Innanzitutto voglio ringraziare il prof. Marco Roccetti e il dott. Gustavo Marfia per la disponibilità e la possibilità datami nello sviluppare una tesi che al momento mi sta dando un bel po' di soddisfazioni, come la pubblicazione di un poster e la sottomissione di un articolo: cose che sicuramente non capitano tutti i giorni a chi deve ancora laurearsi.

Un grazie va a tutti gli amici dell'università per le serate fatte insieme, anche fino all'alba, parlando di massimi sistemi e vaccate allucinanti, per poi finire ad essere "in confidenza" con "la lavatrice".

Un grazie ai miei coinqui reali e "adottati", che mi hanno fatto stare bene sin dal mio arrivo "in quel" di Bologna.

Un saluto ai "The Threemons" con i quali ho affrontato, partita dopo partita, un entusiasmante campionato di volley CUSB ottenendo bellissimi risultati, non solo sul campo ma anche al pub nel post partita all'insegna di hamburger!

Un ringraziamento speciale va alla mia personale Art Director, la mia sorellona Lisa che mi ha dato un'enorme mano nella realizzazione della grafica e immagini presenti in poster, articolo e tesi, il tutto sviluppato in tempo record.

Un enorme grazie va a mamma e papà che mi hanno sempre incoraggiato in tutto e consigliato al meglio, lasciandomi anche carta bianca circa il proseguimento degli studi.

Un ringraziamento a Fernanda, la mia ragazza, per la pazienza, l'amore e la sconfinata comprensione...anche se devo dire che come correttore della tesi, è stata severissima e intransigente, una vera rompiballe[6]!

[6]In senso positivo, si intende :)