Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

TECNICHE PER LA SEPARAZIONE LIVE DI PIANI DI PROFONDITÀ IN UN FLUSSO VIDEO MEDIANTE KINECT E UNA APPLICAZIONE NELL'INTRATTENIMENTO DIGITALE

Tesi di Laurea in Multimedia e Tecnologie Creative

Relatore:
Chiar.mo Prof.
Marco Roccetti

Presentata da:
Luca Serravalli

Correlatore:
Dott. Gustavo Marfia

Sessione 1
Anno Accademico 2013 / 2014

Per te,
che questa volta non potrai esserci . . .

Indice

Capitolo 1

Introduzione

I primi esempi di applicazione di ciò che oggi ricade sotto la classificazione di *effetti speciali* risalgono già alla prima epoca del cinema analogico a pellicola: autori come *Georges Méliès* hanno investigato sin dall'inizio le possibilità offerte dal cinema, che per la prima volta consentiva in maniera efficace di mostrare anche ciò che comunemente esula dalla realtà e dalla concretezza della vita di ogni giorno. La composizione, e quindi l'estrazione dalla pellicola, è stata una delle prime tecniche utilizzate per ottenere quell'effetto di stupore e meraviglia tanto ricercato.

Il passaggio al digitale ha ampliato ancora di più le possibilità per realizzare questa tipologia di effetti. Sono state sviluppate diverse tecniche che consentono l'estrazione di oggetti dalla scena digitale alla quale appartengono, tra le quali la più famosa è chiamata *chroma keying*: tale tecnica viene ampiamente utilizzata in numerosi contesti, ma possiede una serie di limiti e di requisiti da soddisfare per poterla rendere applicabile. Un aspetto interessante è quindi investigare quali siano le possibilità alternative, e possibilmente con meno problematiche di applicazione, per effettuare keying.

Nell'ambito del corso di Multimedia e Tecnologie Creative, e in seguito a una serie di incontri e approfondimenti avuti con il Signor Pietro Babina circa il possibile ruolo delle tecnologie multimediali in un contesto teatrale, assieme ai colleghi Alkida Balliu, Andrea Monzali e Dennis Olivetti è stato

sviluppato un sistema che si basa sulla profondità per effettuare l'estrazione, utilizzando un dispositivo digitale per l'home gaming chiamato Kinect. Tale sistema sviluppato è l'oggetto di questo lavoro di tesi.

Nel prossimo capitolo verrà introdotto il problema generale nel quale ci si è mossi durante lo sviluppo del sistema: *keying* e *composizione*. Verrà mostrato come i primi esperimenti al riguardo risalgano alla produzione cinematografica a pellicola, e come tuttora si tratti di una tipologia di effetti molto comune e utilizzata anche nel mondo digitale. Si parlerà inoltre dello specifico problema dell'utilizzare la profondità come chiave di estrazione, cioè di separazione in piani: si introdurrà quindi il concetto di *depth map*. Verranno spese alcune parole sullo stato dell'arte delle tecniche di keying esistenti, ed in particolare del chroma keying; successivamente si citeranno le principali tecnologie utilizzabili per il calcolo di depth map, e alcune loro implementazioni che sono rappresentative dello stato dell'arte. Infine, verrà anticipato il senso dei risultati ottenuti.

Nel capitolo successivo verranno mostrate le principali tecniche esistenti per effettuare il cosiddetto keying. Si parlerà in particolare della tecnica chiamata *chroma keying*, dalla quale si è partiti per sviluppare il progetto presentato, e di quali siano i suoi principali svantaggi e limiti.

Nel quarto capitolo si mostrerà una tecnica di keying alternativa a quella basata sul colore dei pixel, cioè l'estrazione tramite profondità. Si introdurrà quindi il concetto di *depth map* e verranno mostrate diverse tecniche e tecnologie in grado di calcolarle. Infine, sulla base di una serie di requisiti che il progetto sviluppato voleva soddisfare, verrà motivata la scelta tecnologica adottata, e verranno illustrate le caratteristiche dello specifico strumento utilizzato: il *Kinect* prodotto da Microsoft.

Il quinto capitolo sarà invece interamente dedicato alla presentazione del software realizzato. Si parlerà delle librerie di sviluppo utilizzate, in particolar modo della libreria *OpenCV*, e verranno illustrate le principali fasi che compongono il flusso di lavoro del sistema sviluppato: *inizializzazione* e *acquisizione* dei dati dal Kinect, *miglioramento* della depth map e sua

applicazione per estrarre la parte di scena desiderata, *composizione* in un unico flusso finale dei diversi flussi video e *miglioramento* dei bordi sul risultato finale. In particolare verrà dedicata particolare attenzione alla fase di miglioramento della depth map, nella quale verranno mostrati i primi due approcci tentati e la soluzione adottata infine.

Nel sesto capitolo verrà mostrata una possibile applicazione nel campo dell'intrattenimento digitale: l'utilizzo nel teatro multimediale proposto dal Signor Pietro Babina.

Nell'ultimo capitolo, infine, si valuterà quale sia il risultato raggiunto dal sistema realizzato, in particolare dal punto di vista del numero di *frame per secondo* prodotti e della qualità dell'estrazione digitale; verranno mostrati una serie di test realizzati e si parlerà dei limiti e delle problematiche rilevate.

Capitolo 2

Problema, stato dell'arte e anticipazione dei risultati ottenuti

Il desiderio di stupire è sempre stato una costante presente nell'arte, in qualsiasi sua forma. L'invenzione del cinematografo ha dato in questo senso una fondamentale scossa alle possibilità espressive degli artisti, e il cinema è divenuto la forma d'arte più utilizzata per mostrare anche ciò che l'uomo riesce a vedere soltanto nel sogno e non nella realtà.

Le prime applicazioni di quelli che oggi vengono comunemente definiti *effetti speciali* risalgono alla fine del 1800, dovuti in gran parte all'inventiva di *Georges Méliès*, precursore dell'utilizzo di tecniche quali la mascherina e la sovrimpressione, e comunemente riconosciuto come inventore del concetto stesso di *montaggio* cinematografico. Le sue opere hanno infatti gettato le basi per lo sviluppo di forme di *composizione*, cioè la combinazione di elementi provenienti da immagini diverse all'interno di una nuova unica immagine, utilizzate ancora oggi nel campo dell'intrattenimento video digitale.

2.1 Il problema affrontato: estrazione e composizione

Estrazione e composizione, nella loro accezione più generica, sono le problematiche indirizzate dal lavoro presentato in questa tesi. Si tratta in generale di processi molto comuni e popolari ad esempio nella produzione cinematografica e televisiva: in particolare è molto frequente il caso nel quale viene estratto un soggetto in primo piano, che viene poi composto con uno sfondo proveniente da una scena diversa.

Il processo di separazione di un soggetto dal resto dell'immagine viene chiamato *keying* o *matting*: partendo dall'immagine originale dalla quale si desidera estrarre un soggetto, viene creato un oggetto chiamato *maschera*; tale maschera identifica quali zone dell'immagine debbano essere estratte e quali no, applicandola all'immagine originale è possibile pertanto estrarre il soggetto voluto. La composizione coinvolge quindi almeno tre sequenze di dati distinte: uno sfondo, un primo piano ed una maschera; le prime due sono codificate nello stesso spazio di colori, mentre la terza è un'informazione codificata in un unico canale e non corrisponde ad alcun colore visibile nelle immagini di partenza.

In generale, un processo di composizione può quindi essere descritto come la ricerca del valore C di un pixel, il quale viene calcolato a partire da un certo colore di primo piano F, un colore di sfondo B e i rispettivi valori di trasparenza α, determinati dalla maschera. Poiché qualsiasi tipo di immagine digitale viene normalmente salvata e processata come immagine a tre canali (ad esempio con il modello RGB), la composizione può essere formalizzata nella seguente equazione in 10 variabili.

$$\begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} = \alpha \begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} + (1 - \alpha) \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} \tag{2.1}$$

E' possibile notare come il valore α determini la provenienza del pixel:

con $\alpha = 0$ viene restituito il pixel proveniente dallo sfondo, con $\alpha = 1$ viene invece restituito il pixel proveniente dal primo piano; per valori intermedi di α, viene restituita una combinazione dei colori dei due pixel.

Tale formula può essere ricondotta all'equazione che descrive un processo di keying: se supponiamo che la variabile C - dove C rappresenta l'immagine su cui eseguire keying - sia nota, si ottiene un sistema lineare sotto determinato, in 3 equazioni e 7 incognite [8]. Per ridurre la complessità intrinseca di questo problema, la maggior parte dei metodi di risoluzione assume che siano presenti requisiti aggiuntivi: se lo sfondo B è noto, ad esempio, il numero di incognite scende a 4.

2.2 La profondità come chiave di estrazione

Una fotografia è sostanzialmente la registrazione statica delle emanazioni luminose degli oggetti che si stanno immortalando: non a caso la parola *fotografia* è composta dai termini foto (dal greco φῶς, -φωτός, *luce*) e -grafia (dal greco γραφία, *disegno*). Quindi, dal momento che le immagini contengono unicamente dati relativi alla luce di una certa scena fisica, qualsiasi tecnica di keying che utilizzi, come punto di partenza, soltanto delle immagini, è naturale che sia costretta a basarsi solamente su valutazione di proprietà come luminosità e colore di un certo pixel.

Tuttavia, il mondo reale è un sistema multidimensionale che evolve nel tempo, del quale la fotografia riesce a catturare soltanto una visione limitata e semplificata; ad esempio, dal punto di vista di un osservatore che sta per scattare una foto, ogni oggetto ha una ben precisa collocazione nello spazio, determinata anche dalla sua *profondità* rispetto alla posizione di chi osserva: nel passaggio da scena reale a fotografia, questa informazione viene però persa.

E' quindi necessario creare un oggetto chiamato *depth map* della scena, ossia una immagine nella quale sono espresse le informazioni relative alla distanza di ogni superficie dall'osservatore. In una depth map, ogni pixel

possiede una tinta di grigio più o meno scuro a seconda della sua distanza: nell'esempio in figura 2.1, più l'oggetto è vicino nella realtà, più nella depth map possiede un colore chiaro; più è lontano, più possiede un colore scuro.

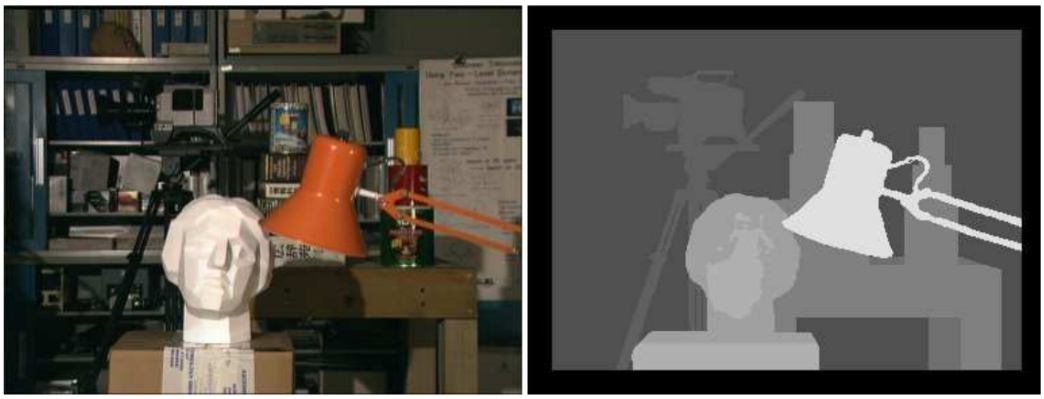


Figura 2.1: A sinistra la scena reale, a destra la depth map ground truth ricavata [31]

Depth map ed estrazione con profondità da essa ricavata sono proprio le specifiche problematiche che verranno affrontate nel corso di questa tesi. Più precisamente ancora, si affronterà il problema dell'estrazione con profondità operante in *tempo reale*, cioè con l'applicazione dell'effetto durante l'esecuzione stessa della ripresa, utilizzando inoltre il *Kinect*, un dispositivo elettronico creato per un contesto video ludico, come mezzo per ottenere la profondità di una determinata scena.

2.3 Keying: stato dell'arte

Le tecniche di keying più utilizzate sono sostanzialmente basate sull'utilizzo di informazioni relative al colore o alla luminosità del pixel; nel prossimo capitolo verranno illustrate più dettagliatamente le principali esistenti. Tra le varie tecniche, la più utilizzata tuttora con successo è il *chroma keying*, chiamata anche *green* o *blue matting* [34] a seconda del colore di sfondo utilizzato come chiave.

Esistono diverse implementazioni, sia dal punto di vista hardware sia software, che consentono di ottenere un effetto di chroma keying di buona qualità. Nella maggior parte dei casi, in particolare nella produzione cinematografica che per sua natura non è un'operazione in tempo reale, vengono spesso adottati approcci che operano appunto in post produzione e non in tempo reale: dai vari software come *Final Cut Pro X* [2], *Adobe Premiere Pro CS6* [36], fino a strumentazione specifica per la fase di registrazione [26].Esistono implementazioni che lavorano invece in tempo reale, ottenendo diversi livelli di qualità: soluzioni pensate per operare su semplici computer [25] [18], ma anche una architettura VLSI specificatamente realizzata [1].

Tutte le soluzioni basate sull'utilizzo di un colore come chiave presentano una serie di inconvenienti e di limiti di applicabilità; verranno anch'essi discussi nel prossimo capitolo.

2.4 Depth map e profondità: stato dell'arte

Nel campo della computer vision e della elaborazione di immagini/video, stimare una depth map è tutt'ora ambito di ricerca molto attivo. Esistono sostanzialmente due principali metodologie di acquisizione di depth map: un approccio *passivo* e un approccio *attivo*. Nel primo caso vengono solitamente utilizzati algoritmi di *stereo vision*, in grado di ricavare la profondità a partire da immagini multiple della stessa scena; nel secondo caso si sfruttano invece particolari apparecchi, spesso chiamati *RGB-D camera* per sottolineare una quarta componente relativa alla profondità, dotati di sensori in grado di ricavare la profondità agendo attivamente sull'illuminazione della scena.

Diverse tecnologie sono state sviluppate per calcolare depth map accurate: sistemi di *stereo vision* [9] [37], dei quali non sono però presenti soluzioni operanti con un numero di frame per secondo sufficiente; telecamere di tipo *Time-of-Flight* [17] [12], tipicamente particolarmente costose, e tecnologia a *luce strutturata* [28] [27]. Tali approcci verranno esaminati dettagliatamente nel relativo capitolo.

2.5 Risultati ottenuti

Una selezione basata su profondità presenta un livello di flessibilità maggiore della selezione basata su chroma keying, come avviene per i colori. Consente infatti di eliminare, o perlomeno largamente ridimensionare, molti dei vincoli sulla preparazione del set e sul posizionamento delle luci che esistono invece nelle tecniche di estrazione basate su colore; non si è inoltre vincolati ad effettuare una selezione obbligata con due piani singoli (primo piano e sfondo) come avviene nel chroma keying, ma è possibile ipotizzare selezioni multiple di piani di profondità (ad esempio selezionare ciò che si trova in un intervallo specifico di distanze).

Queste e altre considerazioni giustificano l'interesse che ha portato alla realizzazione del sistema. Verrà infatti mostrato un approccio, funzionante, in grado di selezionare un determinato piano di profondità attraverso la depth map prodotta dal Kinect. In particolare, volendo in primo luogo eseguire un effetto simile a quello ottenuto con chroma keying e green screen, verranno elencati una serie di miglioramenti apportati alla qualità della depth map, in particolare lungo i bordi. Si vedrà che il sistema realizzato è in grado di operare in tempo reale, producendo in media tra i 20 e 25 frame per secondo, con una qualità visiva accettabile.

Capitolo 3

Tecniche di keying

Inizialmente, quando la produzione cinematografica era basata esclusivamente su sistemi analogici, l'estrazione veniva effettuata tagliando fisicamente a mano la pellicola oppure con particolari processi fotochimici; la maschera poteva ad esempio essere una semplice striscia di pellicola monocromatica sovrapposta alla pellicola a colori, in modo che soltanto le parti desiderate rimanessero visibili. L'introduzione di sistemi di registrazione e riproduzione digitali ha poi portato alla realizzazione di metodi di keying implementati via software, nei quali le operazioni vengono effettuate a livello di pixel. Nei seguenti paragrafi verranno presentati alcuni di questi metodi.

3.1 Luma keying

Dal momento che la maggior parte degli standard di codifica video a colori, come ad esempio *PAL* e *NTSC*, forniscono informazioni sulla *luminanza* in modo indipendente dalla *crominanza* (fu una scelta progettuale motivata dal voler comunque mantenere la retro compatibilità con i sistemi già esistenti che trasmettevano soltanto in bianco e nero), è semplice ottenere una maschera utilizzando direttamente i dati sulla luminanza.

In figura 3.1 si vede come sia possibile selezionare il soggetto desiderato applicando una soglia: i pixel che hanno luminanza maggiore o uguale di tale

soglia vengono settati a completamenti bianchi, quelli che hanno luminanza inferiore invece diventano completamenti neri; in realtà, spesso i luma-keyers sono dotati di due soglie, in modo da poter sfumare il passaggio tra pixel bianchi e pixel neri, rendendo i bordi del soggetto più morbidi. Lo svantaggio di tale approccio consiste nel fatto che funziona bene soltanto quando il soggetto che si vuole ritagliare è molto più luminoso dello sfondo sottostante (in questo senso è necessario un requisito aggiuntivo sullo sfondo, come anticipato prima); un esempio di applicazione nel quale questa tecnica rende al meglio è quello di un testo bianco su sfondo nero: in questo caso le lettere vengono estratte molto agevolmente.

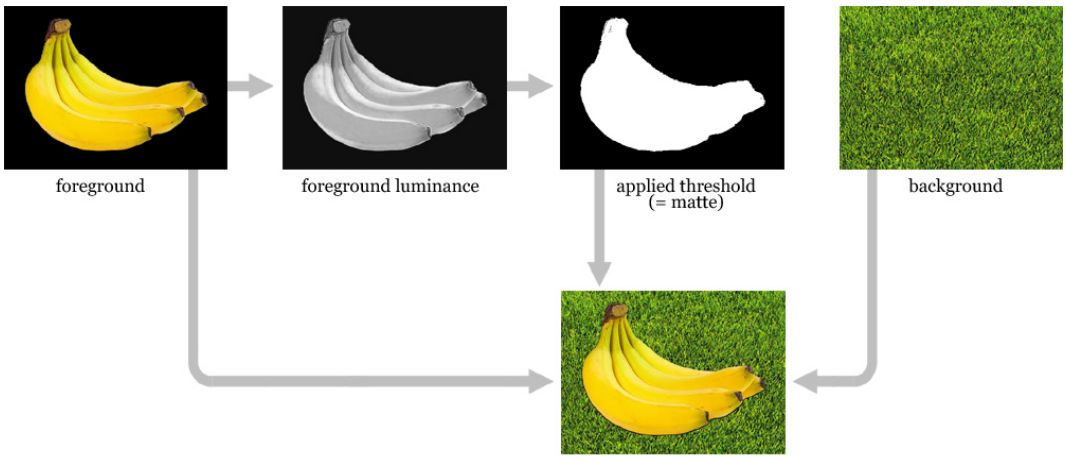


Figura 3.1: Un esempio di applicazione di luma keying (fonte: [33]).

3.2 Difference keying

Un difference-keyer lavora identificando le differenze tra due diverse immagini: la prima contenente uno sfondo e il soggetto da estrarre, la seconda contenente soltanto lo sfondo (ad esempio, una scena con l'attore che recita e la stessa identica scena priva di attore). La differenza in valore assoluto tra

i pixel delle due immagini determina la maschera che consente di estrarre il soggetto, figura 3.2.

Figura 3.2: (a) immagine con target, (b) sfondo privo di target, (c) differenza (fonte: [8]).

Il vantaggio di tale approccio risiede nella possibilità di usare uno sfondo qualsiasi, mentre l'evidente svantaggio è la necessità di riprendere la scena due volte, in modo che lo sfondo sia esattamente lo stesso: questo è un grosso limite, in quanto anche i più piccoli cambiamenti di luce e prospettiva dello strumento di ripresa possono rendere lo sfondo totalmente diverso, rendendo inapplicabile tale metodo.

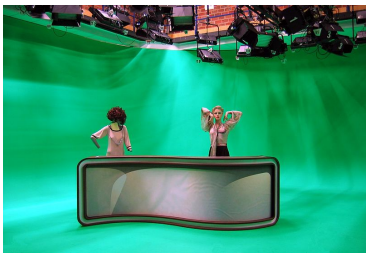
3.3 Chroma keying

Come suggerisce il nome, tale tecnica prende in considerazione la componente cromatica dell'immagine: riprendendo un soggetto su di uno sfondo uniforme di uno specifico colore, è possibile andare a sostituire quel colore con una seconda immagine. In generale, un chroma keyer procede attraverso i seguenti passi:

1. Converte l'immagine RGB in input verso una rappresentazione interna HSV

2. Per ogni pixel, viene calcolata la differenza tra il canale H ed il colore RGB scelto come sfondo

(a) set con sfondo blu



(b) set con sfondo verde

Figura 3.3: Esempio di utilizzo di blue e green back

3. I pixel che superano una certa soglia vengono scartati, al loro posto viene mostrato il pixel proveniente dalla seconda immagine

La conversione da RGB in HSV è necessaria in quanto un buon chroma keyer deve comunque esser in grado di poter distinguere livelli diversi di saturazione e valore, attributi che nel primo modello non sono presenti mentre nel secondo si: la gestione di questi canali si rende necessaria per poter variare con più precisione la tolleranza dei pixel filtrati.

Per quanto riguarda la scelta del colore, qualsiasi tinta può essere usata come chiave per filtrare i pixel di un'immagine. In realtà, storicamente sono stati usati sostanzialmente soltanto due colori: inizialmente il blu e in seguito il verde; è infatti uso comune riferirsi alla tecnica chroma key direttamente con i termini *blueback* (figura 3.3a) oppure *greenback* (figura 3.3b), a seconda del colore del telone di sfondo utilizzato. Il colore blu venne inizialmente scelto in quanto complementare al colore della pelle umana, agevolando l'estrazione di un attore in scena; in più, nelle pellicole tradizionali tale colore aveva una resa migliore del rosso e del verde, ad esempio. Con l'utilizzo di sistemi digitali si è invece preferito passare al colore verde, in quanto il sensore digitale è più sensibile alla tinta verde piuttosto che a quella blu; inoltre, il verde è di per sé più luminoso del blu, facilitando il posizionamento delle luci sul set.

Dal momento che l'intera tecnica si basa sul riconoscimento di un certo colore, è evidente che il soggetto che si vuole estrarre non potrà possedere

nulla del colore scelto, poiché altrimenti verrebbe anch'esso filtrato; fa eccezione il caso in cui si desideri sfruttare il colore proprio per *cancellare* parte del soggetto, ad esempio mascherando parte del corpo di un attore oppure rendendo "invisibili" eventuali operatori sul set (figura 3.4e). Per quanto riguarda invece la qualità del risultato, la principale fonte di problemi è da imputare all'allestimento del set, in particolare alla sua illuminazione. Pertanto, per la buona riuscita del processo, è necessario prestare attenzione alle seguenti considerazioni:

- lo sfondo e il soggetto devono essere illuminati in modo indipendente, con luci il più possibile regolari
- il soggetto deve essere illuminato in modo da non creare ombre che vadano a finire sulla porzione di telone inquadrata: in caso contrario, le ombre potrebbero essere rilevate come tonalità più scure del colore scelto e quindi non essere filtrate
- il materiale con cui è realizzato il telone di sfondo influisce sul risultato: i materiali opachi sono da preferire a quelli lucidi, in quanto questi ultimi riflettono maggiormente la luce e creano quindi zone di colore più scuro e più pallido, che di nuovo potrebbero essere rilevate come tonalità più scure o più chiare del colore e quindi non essere filtrate
- una sorgente di controluce sul soggetto può essere utile per semplificare la procedura di scontornamento dei bordi del soggetto
- è necessario evitare che si verifichi il cosiddetto *colour spill*: la luce riflessa dal telone di sfondo rimbalza sul soggetto e lascia una leggera tinta colorata sui bordi ed in certe aree (in particolare nei capelli)

Riassumendo, la tecnica di keying sopra presentata ha in generale le seguenti criticità: il colore usato come sfondo non può essere presente sul soggetto; è necessaria la realizzazione di un set complesso, cosa che la rende poco pratica per l'utilizzo in scenari non precedentemente preparati; il risultato finale dipende soprattutto dalla qualità dell'illuminazione, che deve essere

organizzata accuratamente. Ciò nonostante, il chroma keying è tutt'ora la tecnica di keying più utilizzata sia in ambito televisivo sia nella produzione cinematografica (alcuni dei tanti esempi di utilizzo in figura 3.4), settori nei quali è possibile una adeguata preparazione del set e soprattutto un intervento in *post-produzione*: questo significa che eventuali errori rilevati durante la registrazione possono poi essere corretti con adeguati software di computer grafica (ad esempio problemi di colour spill o bordi non perfettamente scontornati). Inoltre, sopratutto nella produzione cinematografica, l'intero processo di estrazione e seguente composizione solitamente viene effettuato in un momento successivo alla effettiva registrazione delle scene: in sostanza, nella maggior parte dei casi non sono richiesti algoritmi in grado di lavorare in tempo reale.

Partendo da questi presupposti, ci si è chiesto se fosse possibile realizzare una tecnica alternativa al chroma keying, in grado di lavorare senza la necessità di utilizzare un telone di sfondo da cui filtrare il colore, con meno vincoli sull'illuminazione utilizzata e, soprattutto, in grado di operare in tempo reale, producendo cioè l'effetto di composizione contemporaneamente alla ripresa.

(a) previsioni meteorologiche

(b) studio virtuale

(c) sfondo virtuale

(d) scenografia in film

(e) mascheramento di attori

Figura 3.4: Esempi di utilizzo di blue e green back in ambito televisivo e cinematografico

Capitolo 4

Tecniche per il calcolo di depth map

La realizzazione del sistema presentato in questo elaborato ha richiesto, come punto di partenza, l'individuazione di una tecnica per il calcolo di depth map che riuscisse a soddisfare i seguenti requisiti:

- produrre una mappa di qualità sufficiente da poter essere utilizzata per l'estrazione di soggetti

- effettuare la computazione in un tempo abbastanza rapido da potere generare fotogrammi ad una velocità vicina a 30fps, condizione necessaria per rispondere al requisito di esecuzione in tempo reale che era stato posto sin dall'inizio

Come si vedrà, rispondere contemporaneamente ad entrambi i due requisiti si rivela essere un compito non alla portata delle principali tecnologie esistenti, se non accettando compromessi in termini di qualità. In particolare, nei prossimi paragrafi verranno brevemente introdotte le principali opportunità che lo stato dell'arte offre, quindi ci si soffermerà sulla tecnica che è stata scelta per realizzare il sistema presentato in questo elaborato.

4.1 Stereo vision e disparity map

La prima tecnica presentata fa parte dell'approccio di tipo passivo, nel quale si risale alle profondità degli oggetti semplicemente sulla base di osservazioni della scena, senza agire attivamente su di essa.

Partendo da una singola immagine non è possibile risalire alla struttura tridimensionale di una scena, in quanto nella proiezione prospettica da spazio 3D a spazio 2D è implicitamente presente la perdita di tali informazioni. Se si hanno invece a disposizione più immagini, cioè più punti di vista della scena, è possibile calcolare la profondità sulla base delle differenze tra le varie immagini. In particolare, in modo analogo a quanto avviene nella visione binoculare del sistema visivo umano, è sufficiente una coppia di immagini raffiguranti la stessa scena, ripresa da due punti di vista vicini e sullo stesso asse orizzontale: sfruttando le relazione geometriche che intercorrono tra i punti 3D reali e le loro proiezioni sui piani 2D (*geometria epipolare*), si può calcolare una *mappa delle disparità* della scena, ricavando poi la profondità attraverso un processo di *triangolazione*.

In figura 4.1 sono indicate le relazioni per due punti di vista O_l e O_r rispetto al punto P del quale si vuole calcolare la profondità: in questo caso la disparità, cioè la differenza tra le ascisse di due punti corrispondenti, è data da $x^l - x^r$.

In generale, un sistema di stereo vision opera attraverso i seguenti passi.

1. *Calibrazione*: vengono rilevate le caratteristiche delle due telecamere di ripresa (ad esempio: lunghezza focale, distorsione della lente)

2. *Rettificazione*: utilizzando i parametri ricavati con la calibrazione, si rimuove la distorsione e si rendono le immagini epipolari

3. *Corrispondenza*: nella coppia di immagini stereo vengono identificati i punti corrispondenti, creando la mappa delle disparità. Questo è tipicamente il passaggio più difficile e oneroso in termini di computazione, in quanto un punto p_l dell'immagine di sinistra potrebbe teoricamente

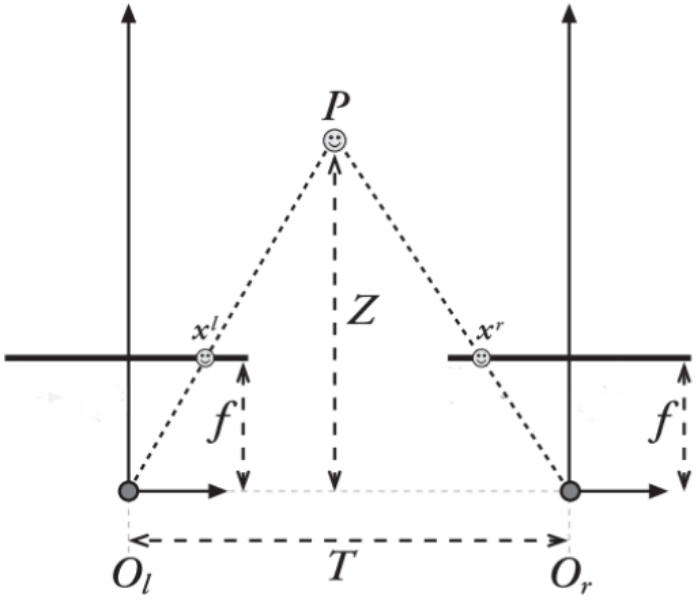


Figura 4.1: Relazioni geometriche epipolari in un sistema di stereo vision

avere corrispondenza con qualsiasi punto p_r che giace sulla retta epipolare associata a p_l

4. *Triangolazione*: sulla base della mappa di disparità, viene calcolata la posizione nello spazio 3D di ogni punto del piano

Verranno ora mostrate due tecniche facenti parte dell'approccio attivo, nel quale cioè vengono utilizzate particolari apparecchiature che intervengono direttamente sulla scena emettendo un segnale luminoso.

4.2 Telecamere Time-of-Flight

Le telecamere di tipo *Time-of-Flight* sono una tipologia di *Lidar*, una tecnologia di telerilevamento che calcola la distanza di un oggetto illuminandolo con un laser, ed analizzandone la luce riflessa. Si tratta inoltre di Lidar di tipo *scannerless*, in grado cioè di rilevare l'intera scena con un'unica misurazione.

Questo tipo di telecamera utilizza quindi un emettitore per mandare un segnale di luce, tipicamente infrarosso, verso l'oggetto del quale si vuole calcolare la profondità. Esistono poi diverse modalità per analizzare il segnale che, rimbalzando, torna indietro verso l'emettitore:

- sfruttando il fatto che la velocità della luce è una costante nota, una ToF di tipo *trigger mode* calcola direttamente la distanza in funzione del tempo necessario all'impulso luminoso per raggiungere l'oggetto e tornare indietro (figura 4.2)

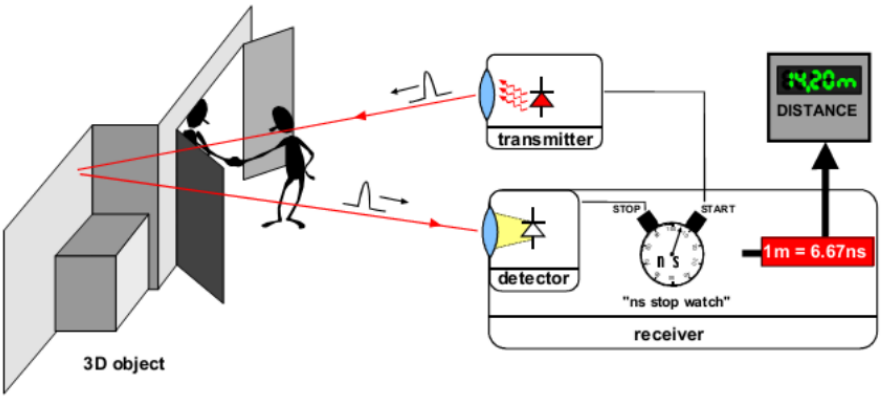


Figura 4.2: Schema di funzionamento di una ToF trigger mode

- le *range gated* possiedono invece un otturatore che si apre e si chiude alla stessa frequenza con la quale vengono emessi i segnali luminosi: la distanza viene quindi calcolata in funzione della quantità di luce che, rientrando verso la telecamera, viene persa in quanto bloccata dall'otturatore

- infine, le telecamere *RF-modulated wave*, al posto di inviare brevi impulsi come nei casi precedenti, emettono segnali luminosi continui sotto forma di onda: la profondità viene calcolata in funzione della differenza di fase con la quale l'onda riflessa torna verso l'apparecchio (figura 4.3)

In commercio esistono diversi modelli di telecamere ToF [14][15][24], studiate sia per operare sulla breve distanza (ad esempio per effettuare riconosci-

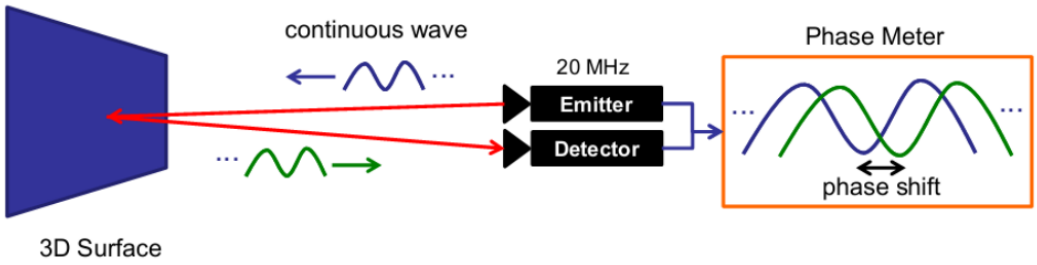
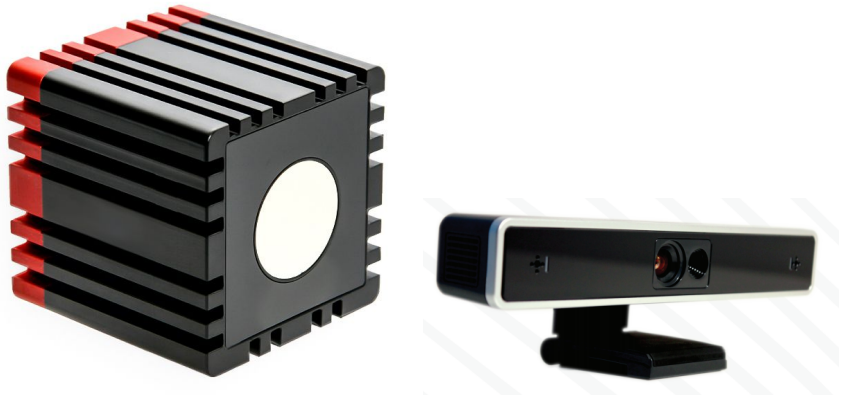


Figura 4.3: Schema di funzionamento di una ToF a *continuous wave modulation*

mento di gesti) sia sulla media distanza (per effettuare riconoscimento dell'intero corpo umano). A titolo esemplificativo, la tabella 4.1 mostra le principali caratteristiche tecniche di alcuni dei modelli attualmente più utilizzati.

	DS311	*SR 4000*	*CamBoard nano*
Produttore	SoftKinetic	Mesa Imaging	PMD
Campo di rilevamento	0.15-1.0 m, 1.5-4.5 m	0.1-5.0 m/0.1-10.0 m	0-2 m
Frame rate	25fps-60fps	50fps massimi	90fps massimi
Risoluzione depth map	160x120 (QQVGA)	176x144 (QCIF)	160x120
Risoluzione RGB	640x480 (VGA)	-	-
Illuminazione usata	Infrarossi	Infrarossi	Infrarossi
Tipologia	Trigger mode	Wave modulation	Wave modulation
Costo indicativo	$299.00	$4,295.00	$490

Tabella 4.1: Caratteristiche principali di alcuni modelli di telecamera ToF



(a) SR 4000 prodotto da Mesa Imaging

(b) DS 311 prodotto da SoftKinetic

Figura 4.4: Alcuni modelli in commercio di telecamere ToF

4.3 Luce strutturata

Con il termine *luce strutturata* si intende la proiezione sulla scena di un certo pattern di pixel, noto a priori. L'idea è semplificare il problema della corrispondenza visto nell'approccio con immagini stereo: al posto della seconda telecamera può essere utilizzato un proiettore laser (a patto che venga effettuata una calibrazione con la prima telecamera), e la triangolazione può quindi essere effettuata tra il pattern "virtuale" e quello percepito sull'oggetto.

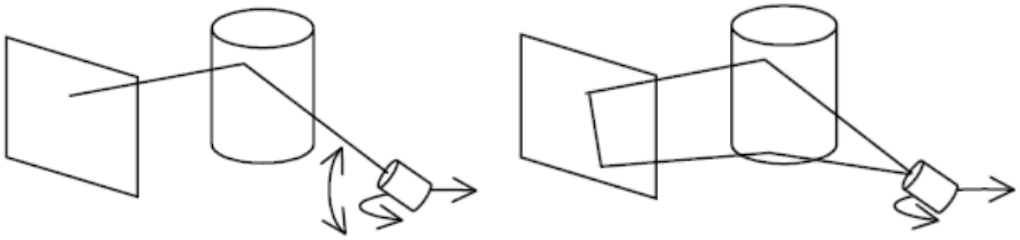
Per quanto riguarda il pattern da proiettare esistono diverse possibilità, di seguito sono mostrati alcuni esempi.

- *singolo punto*: viene proiettato un unico punto per volta, che può essere quindi individuato senza ambiguità (figura 4.5a). In questo caso l'accuratezza è elevata, ma la scansione può essere molto lenta in quanto bisogna coprire tutta la scena, spostando volta per volta il punto proiettato sia sull'asse x sia sull'asse y

- *striscia*: proiettando una singola striscia si aumenta la velocità di scansione, poiché tutti i punti sulla linea vengono rilevati in una volta sola

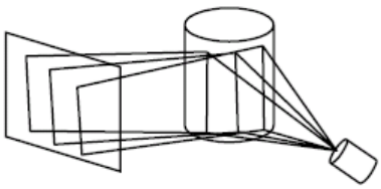
ed è quindi sufficiente la scansione lungo un unico asse; allo stesso tempo la corrispondenza è ancora univoca, in quanto la striscia proiettata interseca ogni linea epipolare in un punto unico (figura 4.5b)

- *strisce multiple/griglia*: con un pattern di questo tipo l'intera scena viene coperta in un colpo solo, evitando quindi la fase di scanning necessaria nei casi precedenti (figure 4.5c e 4.5d). Viene però persa la condizione di univocità, in quanto non è più possibile stabilire a priori quale delle n strisce sia quella corrispondente; si ripresenta quindi il problema iniziale della corrispondenza, seppur in forma semplificata. E' comunque possibile utilizzare delle forme di *encoding* per evitare la perdita di ambiguità, ad esempio adottando colorazioni diverse per le strisce, oppure codificando in modo diverso sulla base del tempo o dello spazio [23][29][4]

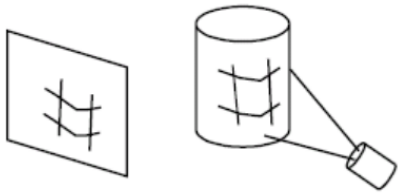


(a) pattern a punto singolo

(b) pattern a striscia



(c) pattern a strisce multiple



(d) pattern a griglia

Figura 4.5: Esempi di pattern proiettati da dispositivi a luce strutturata (fonte [10])

L'approccio a luce strutturata viene impiegato soprattutto sotto forma di *scanner laser 3D* utilizzati in svariati settori, da quello industriale a quello medico, e in generale in tutti quei contesti nei quali è necessario poter disporre del modello 3D di un dato oggetto, per poterlo poi elaborare con software di grafica 3D; in figura 5.6 alcuni esempi di scanner 3D professionali basati su luce strutturata. Questa tecnologia viene anche impiegata in soluzioni ibride, nelle quali la luce strutturata viene combinata a tecniche di stereo vision [32].

(a) HDI Advance Scanner prodotto da LMI Technologies

(b) HDI 120 prodotto da LMI Technologies

(c) eQT Rexcan CS+ prodotto da eQuality Tech Inc

Figura 4.6: Alcuni modelli in commercio di scanner laser 3D a luce strutturata

4.4 Confronto tra le tecniche e scelta effettuata

Ognuno dei metodi mostrati in precedenza ha i propri pregi e difetti. Calcolare una depth map tramite coppia di immagini stereo ha il vantaggio di non richiedere attrezzature specifiche, in quanto sono sufficienti due normali telecamere e non è necessario intervenire attivamente sulla scena: questo si traduce nel fatto che, in generale, si tratta di una tecnica che è possibile applicare sia in ambienti con illuminazione artificiale sia con illumi-

nazione naturale. Al contrario, le soluzioni che ad esempio adoperano laser ad infrarossi subiscono l'interferenza della luce solare diretta.

Allo stesso tempo, dal momento che il calcolo della distanza è basato su analisi di immagini, il matching corretto dei punti corrispondenti dipende sostanzialmente dal colore e dalla texture delle superfici fotografate. Si pensi ad esempio al caso nel quale si vuole calcolare quanto disti un muro di colore uniforme: la mancanza di variazioni sulla superficie del soggetto crea problemi nella fase di ricerca dei punti corrispondenti tra le due immagini stereo; questo problema non si pone invece nelle altre due tecnologie mostrate.

Le telecamere di tipo ToF, rispetto all'approccio basato esclusivamente su stereo vision, hanno il vantaggio di operare senza alcun tipo di setup iniziale (calibrazione e rettificazione) in quanto è sufficiente un'unica telecamera e non due. Analogamente alle telecamere a luce strutturata, riescono inoltre a produrre un numero molto elevato di frame per secondo; da questo punto di vista, tra le telecamere ToF e quelle a luce strutturata, le seconde ottengono in media prestazioni inferiori, poiché necessitano di proiezioni multiple (scanner 3D) oppure di forme di encoding computazionalmente costose.

Nella scelta della tecnologia più idonea allo sviluppo del progetto, la tecnologia ToF è stata automaticamente esclusa a causa del suo costo elevato e dell'impossibilità di reperire una telecamera di quel tipo. Sono state quindi valutate, in particolare dal punto di vista del requisito di real timeness, le possibilità attualmente offerte dalle due rimanenti tecnologie.

Il Middlebury College ha compiuto una classificazione dei principali algoritmi di stereo vision [31], ideando anche un sistema di valutazione online al quale gli sviluppatori possono sottoporre i propri algoritmi, per testarne l'efficacia rispetto le tecniche già esistenti: i risultati di queste valutazioni sono disponibili online, con una classifica in termini di qualità della mappa di disparità realizzata [30]. Da tale classifica emerge come ad una elevata qualità della depth map corrisponda anche un elevato tempo di calcolo, tale da non poter essere utilizzabile in una situazione real time. L'unico algoritmo in grado di coniugare qualità discreta a tempi di calcolo ridotti risulta essere

ADCensus [19], che nella sua implementazione per CPU richiede comunque 2.5 secondi per calcolare la depth map del dataset di esempio Tsukuba; esiste anche una implementazione per GPU che nel caso Tsukuba risulta decisamente più veloce (0.016 secondi), ma che se applicata a una sequenza video 320x240 produce un output a non più di 10 fps, risultando quindi non adatto al contesto considerato.

L'approccio stereo vision è stato quindi scartato e sono state considerate le alternative disponibili basate su tecnologia a luce strutturata. In particolare, tra i vari dispositivi esistenti, è stato scelto un apparecchio nato in realtà per essere utilizzato nel settore dell'*home gaming*, ma che per le sue caratteristiche si avvicinava più di tutti al soddisfare i requisiti imposti: il Kinect per Xbox 360 [22].

4.5 Il Kinect

Prodotto da Microsoft e commercializzato nel Novembre 2010 come accessorio per la console Xbox360, il Kinect è stato il primo device disponibile per il grande pubblico, ad un prezzo accessibile, a contenere una tecnologia capace di effettuare operazioni che prima venivano svolte soltanto da apparecchi professionali.

Nonostante il Kinect sia stato ideato per consentire all'utente di interagire con l'Xbox senza alcun tipo di controller ed è quindi principalmente pensato per il riconoscimento di gesti e del corpo umano, le sue caratteristiche tecniche hanno destato l'interesse di studiosi e sviluppatori, che l'hanno utilizzato anche in altri contesti: dalla modellazione 3D di ambienti interni [13], all'utilizzo come sistema di navigazione in robot mobili [3] o come ausilio a sistemi di riabilitazione fisica [7].

4.5.1 Caratteristiche generali

Il Kinect, in riferimento ai diversi approcci mostrati precedentemente, si tratta di una soluzione ibrida: combina infatti una tecnologia basata su

luce strutturata, sviluppata e brevettata [11] dall'azienda *PrimeSense*, alle classiche tecniche di stereo vision. L'apparecchio è dotato di un proiettore di luce infrarossa, una telecamera a infrarossi ed una telecamera RGB (figura 4.7), ed è in grado di produrre contemporaneamente uno stream a colori e uno stream relativo alla depth map ciascuno di 640x480 pixel, con una velocità massima di 30fps [21].

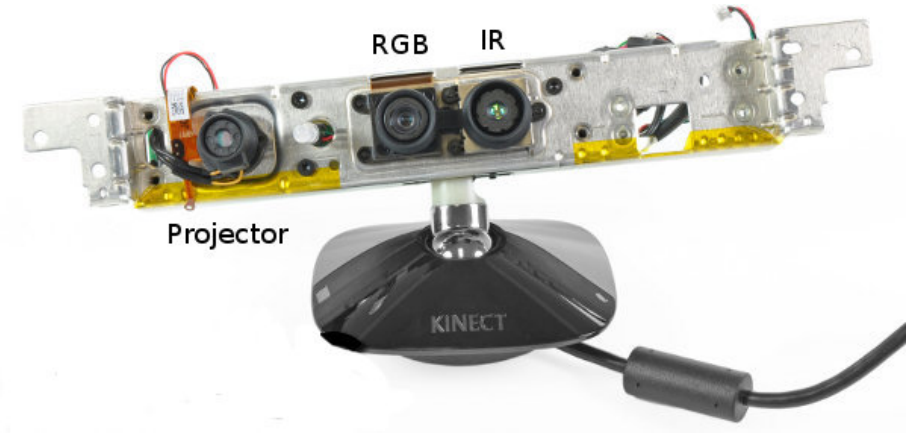


Figura 4.7: Vista del Kinect con la cover superiore rimossa

Il processo di creazione della depth map non è noto in tutti i suoi dettagli, in quanto si tratta come già detto di tecnologia brevettata, ma agisce sostanzialmente attraverso i seguenti passi:

- il sensore IR proietta un pattern noto di tipo *speckle* (cioè a macchiolina), figura 4.8. Si tratta di un pattern ottenuto con un unico impulso di luce che passa attraverso un reticolo di diffrazione, venendo quindi diviso in una serie di puntini che coprono l'intera scena

- la telecamera IR rileva il pattern sulla scena e lo confronta con un set di pattern che sono stati precedentemente proiettati a distanze note e memorizzati sul device; il proiettore IR e la telecamera IR agiscono quindi come una coppia di classiche telecamere stereo, mentre la telecamera RGB non partecipa alla realizzazione della depth map.

Attraverso l'analisi degli spostamenti del pattern proiettato rispetto quello di riferimento, viene quindi realizzata una mappa delle disparità

- utilizzando la mappa delle disparità, per ogni pixel viene ricavata la sua relativa profondità

A questo punto la depth map viene correlata al frame proveniente dalla telecamera RGB e vengono mostrati in output i due stream video.

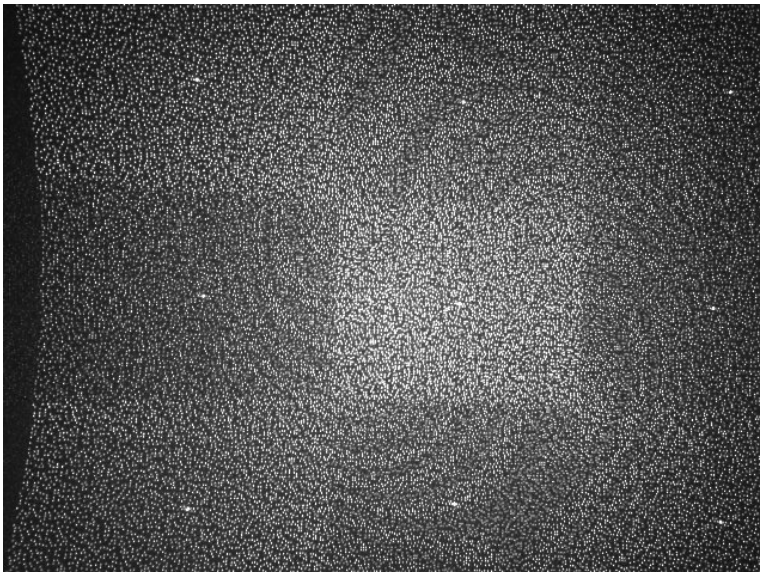


Figura 4.8: Pattern a macchia proiettato dal Kinect

4.5.2 Qualità della depth map

Gli errori che vanno a deteriorare la qualità della depth map calcolata dal Kinect derivano principalmente dalle condizioni di illuminazione, dalla geometria degli oggetti in scena e dalle proprietà delle superfici degli oggetti.

I problemi legati alle condizioni di illuminazione derivano sostanzialmente dall'eventuale presenza di forte luce, in particolare del sole, diretta sulla scena: in questa situazione il pattern IR viene percepito con poco contrasto dalla telecamera IR, e questo può portare a errori di misurazione. Anche

il posizionamento geometrico degli oggetti influisce in modo molto evidente: all'aumentare della distanza dal sensore aumenta anche l'errore casuale relativo alla misurazione [16] (si noti che l'intervallo di operatività del sensore è compreso tra circa 0.5 m e 5.0 m). Inoltre, a seconda di come è orientato l'oggetto, può ad esempio capitare che un lato venga illuminato dal proiettore ma non possa esser visto dalla telecamera, o viceversa: questo fenomeno è chiamato *occlusione*. Infine, anche le superfici particolarmente lucide possono impedire la corretta misurazione. Esempi di questi problemi sono osservabili in figura 4.9.

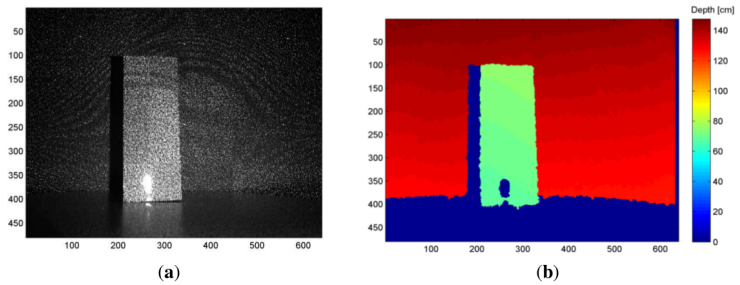


Figura 4.9: (a) Immagine infrarossi del pattern proiettato (b) la relativa depth map. Si possono notare una occlusione sul lato sinistro dell'oggetto e una errata misurazione dovuta ad una piccola area sovraesposta.

Per quanto riguarda l'accuratezza, cioè un indice di quanto il valore misurato si discosti dal valore reale, è stato calcolato [16] che l'errore casuale varia da pochi millimetri quando la distanza è vicina ai 0.5 m, fino a 4 cm quando la distanza è intorno ai 5 m. Nello stesso studio è stato inoltre confrontato il Kinect con il FARO LS 880, uno scanner laser professionale (il suo costo si aggira sui 5000$): l'84% dei punti della *point cloud* del Kinect sono risultati essere distanti meno di 3 cm dai rispettivi punti della point cloud del laser FARO.

Capitolo 5

Il sistema realizzato

Il progetto esposto in questo elaborato ha come obbiettivo principale implementare un effetto di blue back, solitamente realizzato tramite chroma keying, utilizzando invece la profondità come chiave di selezione e il Kinect come mezzo per calcolare la profondità.

Il flusso di lavoro del sistema si articola nei seguenti passi principali:

Acquisizione dati viene instaurata la connessione con il Kinect e vengono recuperati i flussi RGB e della depth map

Taglio e miglioramento della depth map viene selezionata la parte di depth map relativa alla profondità desiderata, quindi vengono eseguiti una serie di algoritmi atti a migliorarne la qualità, per poter poi procedere all'estrazione

Composizione il flusso RGB selezionato viene composto con il flusso RGB proveniente da una seconda sorgente

Blurring viene applicato un miglioramento ai bordi del frame finale composto

Il software è stato realizzato con il linguaggio di programmazione *C++* in ambiente di sviluppo *Visual Studio*, su sistema operativo *Windows 7*. Per la gestione del Kinect sono state utilizzate le librerie fornite nel SDK v1.7

rilasciato da Microsoft [20], del quale sono state utilizzate le API in C++. E' stata inoltre utilizzata la libreria OpenCV, attualmente lo strumento di sviluppo più utilizzato nel campo della computer vision, sulla quale verranno ora spese alcune parole. Successivamente verranno illustrate più nel dettaglio le varie fasi nelle quali si articola il sistema.

5.1 La libreria OpenCV

Open source Computer Vision è una libreria multi piattaforma, rilasciata con licenza BSD, che contiene più di 2500 algoritmi relativi a computer vision e machine learning [5]. E' composta da 4 moduli principali, mostrati in figura 5.1: CV contiene le funzioni per l'elaborazione di immagini, MLL contiene classificatori e algoritmi per reti neurali (questo modulo non è stato utilizzato per il progetto), HighGUI contiene le funzioni per gestire I/O e infine in CXCore sono implementate le strutture dati di base.

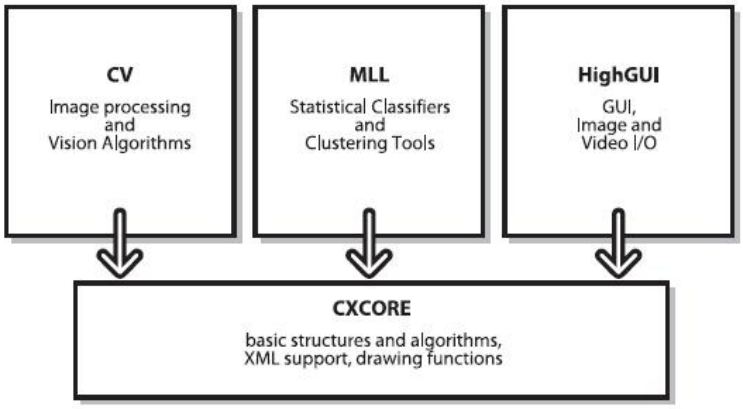


Figura 5.1: I moduli principali che compongono OpenCV

Le due strutture dati più utilizzate per memorizzare matrici, e quindi immagini, sono gli oggetti `Mat` e `IplImage`. `IplImage` è in realtà il formato originale utilizzato nelle prime versione di OpenCV, mentre `Mat` è stato introdotto successivamente: il vantaggio di usare quest'ultimo è che non ne-

cessita di gestione manuale della memoria, in quanto il *garbage collector* dell'interfaccia C++ si occupa del lavoro. Alcune funzioni delle API del Kinect utilizzano ancora il vecchio formato, tuttavia quando possibile si è preferito adoperare le `Mat`. Un aspetto da sottolineare è che OpenCV memorizza i canali dei colori nell'ordine *BGR* invece che nell'usuale *RGB*.

5.2 Acquisizione dati dal Kinect

Il primo passo effettuato consiste nell'inizializzazione del Kinect, effettuata utilizzando le apposite API messe a disposizione dal SDK del dispositivo. In particolare si richiede al Kinect di prepararsi a fornire lo stream relativo alla depth map, alla telecamera RGB e al rilevamento delle persone. Quest'ultimo flusso di dati verrà utilizzato più avanti nella fase di gestione della depth map.

```
1 NuiInitialize(NUI_INITIALIZE_FLAG_USES_DEPTH |
     NUI_INITIALIZE_FLAG_USES_COLOR |
     NUI_INITIALIZE_FLAG_USES_SKELETON);
```

Le chiamate `NuiImageStreamOpen`, utilizzate per inizializzare i flussi dati desiderati, sono basate su un meccanismo *a maniglia*: vengono utilizzati una serie di oggetti `HANDLE` ai quali fare successivamente riferimento per reperire i dati.

Dopo la fase di inizializzazione, che viene effettuata ovviamente soltanto una prima volta, possono essere via via reperiti i frame prodotti dal dispositivo. Viene utilizzato un modello a eventi: quando un nuovo frame è disponibile per essere processato, viene inviato un segnale sulla rispettiva maniglia. Il thread in attesa sulla `HANDLE` può quindi eseguire la chiamata `NuiImageStreamGetNextFrame` e recuperare i dati.

Per accedere fisicamente ai dati viene utilizzato un sistema basato su *lock*, per evitare che questi vengano corrotti durante il processo di recupero: un

oggetto di tipo `NUI_LOCKED_RECT` contiene un puntatore al buffer che contiene i dati, i quali vengono acceduti soltanto dopo aver effettuato una chiamata `LockRect`. Tale funzione blocca il buffer per l'accesso in lettura e scrittura. Dopo aver copiato i dati, il frame del Kinect viene rilasciato con la funzione `NuiImageStreamReleaseFrame`. Tale sequenza di istruzioni viene ripetuta per ogni stream al quale si desidera accedere.

L'ultimo passaggio di questa fase è relativo all'allineamento: infatti dal momento che i frame della depth map e quelli della telecamera RGB provengono da due diverse sorgenti, come in qualsiasi sistema stereo, non inquadrano esattamente la stessa porzione di scena. Viene quindi utilizzata una funzione che adatta il frame della depth map a quello RGB, rendendo i due perfettamente sovrapponibili.

5.3 Miglioramento della depth map

Una prima analisi della depth map prodotta dal Kinect ha messo in luce come abbia una qualità non sufficiente per il contesto di applicazione, cioè l'estrazione di una persona umana dalla scena nella quale si trova. Infatti, dal momento che nel contesto dell'home gaming non è necessario un livello di dettaglio elevato, le depth map del Kinect sono particolarmente carenti sui bordi degli oggetti e nelle aree come dita e testa (figura 5.2).

Il punto di partenza è stato quindi tentare di migliorare la qualità della depth map nelle zone vicino ai bordi: sono stati sviluppati e valutati tre diversi algoritmi, che verranno presentati nei prossimi paragrafi, basati sull'utilizzo di tecniche di Edge Detection come ausilio per il completamento dei bordi errati presenti nella depth map.

D'ora in avanti si noti che, se non diversamente indicato, quando verrà utilizzato il termine depth map si intenderà sempre una depth map alla quale è già stata applicata la selezione in base alla profondità richiesta (figura 5.3): la selezione viene graficamente effettuata colorando di nero tutti i pixel ad una profondità maggiore da quella desiderata.

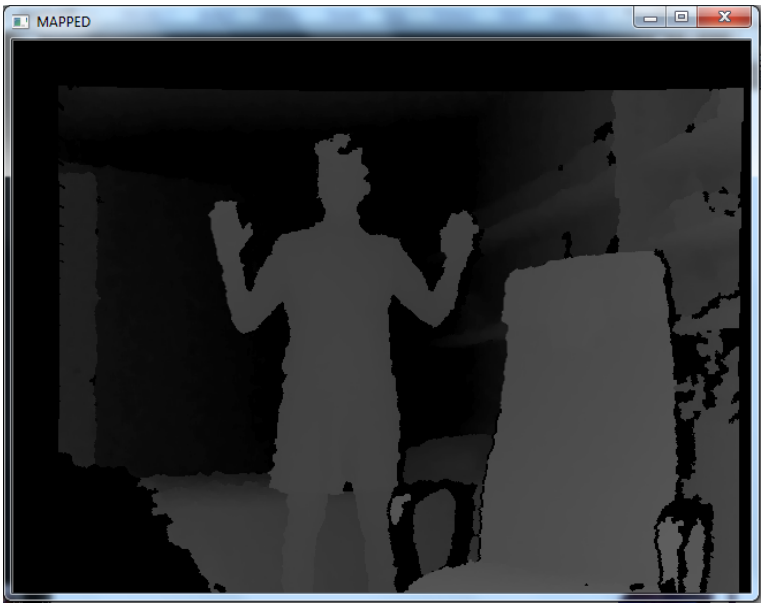


Figura 5.2: Livello di dettaglio della depth map prodotta dal Kinect

Algoritmo 5.1 mascheramento dei pixel. cutdepth rappresenta il valore numerico entro il quale mascherare i pixel

```
1  Mat depthcut = depthmap.clone();
2
3  for(int i=0;i<frame.rows;i++){
4     for(int j=0;j<frame.cols;j++){
5        if( depthmap.at<Vec3b>(i,j)[0] < cutdepth ){
6              depthcut.at<Vec3b>(i,j)[0] = 0;
7              depthcut.at<Vec3b>(i,j)[1] = 0;
8              depthcut.at<Vec3b>(i,j)[2] = 0;
9        }
10    }
11 }
```

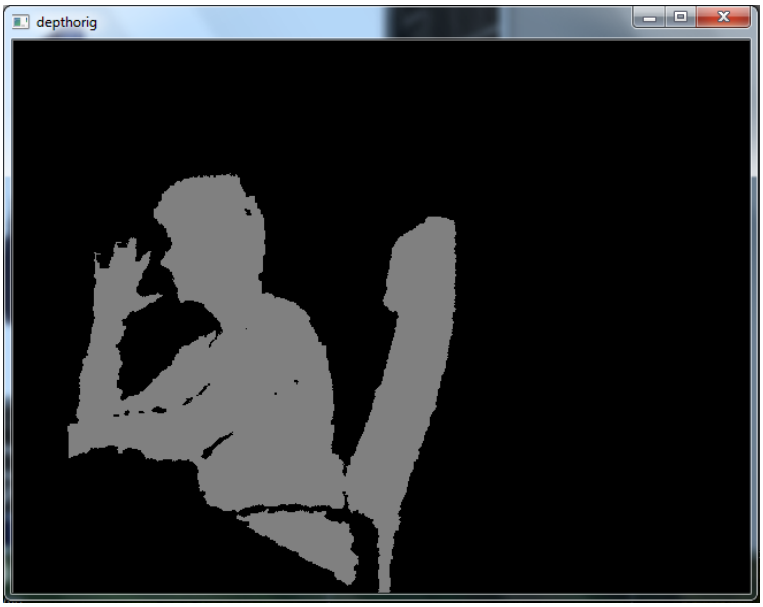


Figura 5.3: Esempio di depth map nella quale è stata mascherata la parte oltre la profondità richiesta

5.3.1 Edge Detection

Il rilevamento dei bordi è una delle operazioni più frequentemente utilizzate nel campo dell'analisi delle immagini. Ogni oggetto è infatti definito anche da un insieme di bordi: oltre a definire graficamente la sua sagoma, il bordo rappresenta anche il limite che separa l'oggetto dallo sfondo, e più in generale da qualsiasi altro elemento visivamente sovrapposto.

Identificare correttamente tutti i bordi in una immagine si rivela quindi una operazione estremamente utile, in quanto attraverso i bordi è possibile risalire a numerose proprietà degli oggetti, quali ad esempio l'area che occupano, il perimetro e la loro forma. Dal momento che la computer vision riguarda anche l'identificazione e classificazione degli oggetti contenuti in una immagine, il problema del rilevamento dei bordi è stato sin dall'inizio campo di ricerca particolarmente fertile [38].

Più specificatamente, un bordo può essere considerato come un cambiamento locale significativo all'interno dell'immagine, che può essere ad es-

empio dovuto a una discontinuità nella profondità, nel colore oppure nell'illuminazione (figura 5.4). Identificare un bordo consiste quindi nel ricercare repentini cambiamenti di intensità, operazione che viene effettuata utilizzando lo strumento matematico della *derivata*. In particolare esistono due principali strategie: il calcolo del massimo sulla derivata prima e la ricerca del cambiamento di segno nella derivata seconda (figura 5.5).

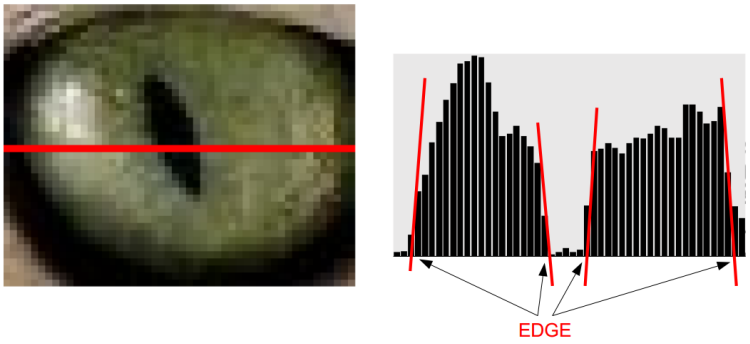


Figura 5.4: Esempio di bordo visto come cambiamento di intensità nell'immagine

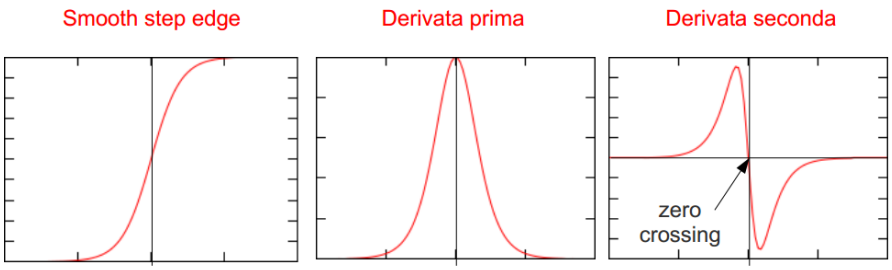


Figura 5.5: Strategie per la ricerca di un bordo, basate sul calcolo della derivata prima e seconda dell'intensità

Nel prossimo paragrafo verrà mostrato nel dettaglio il primo approccio adoperato per il miglioramento dei bordi.

5.3.2 Approccio 1: Algoritmo di Canny ed estensione con direzione

Dal momento che spesso la depth map non segue con precisione quelli che sono i bordi reali degli oggetti, si è pensato di utilizzare un edge detector come guida per l'estensione dell'area coperta dalla depth map. L'edge detector utilizzato è quello basato sull'algoritmo di Canny [6], il quale sfrutta la derivata prima di una gaussiana per la ricerca dei contorni. In particolare, la funzione *Canny* disponibile nella libreria OpenCV ha la seguente intestazione

```
1 void Canny(InputArray image, OutputArray edges, double threshold1,
     double threshold2, int apertureSize=3, bool L2gradient=false )
```

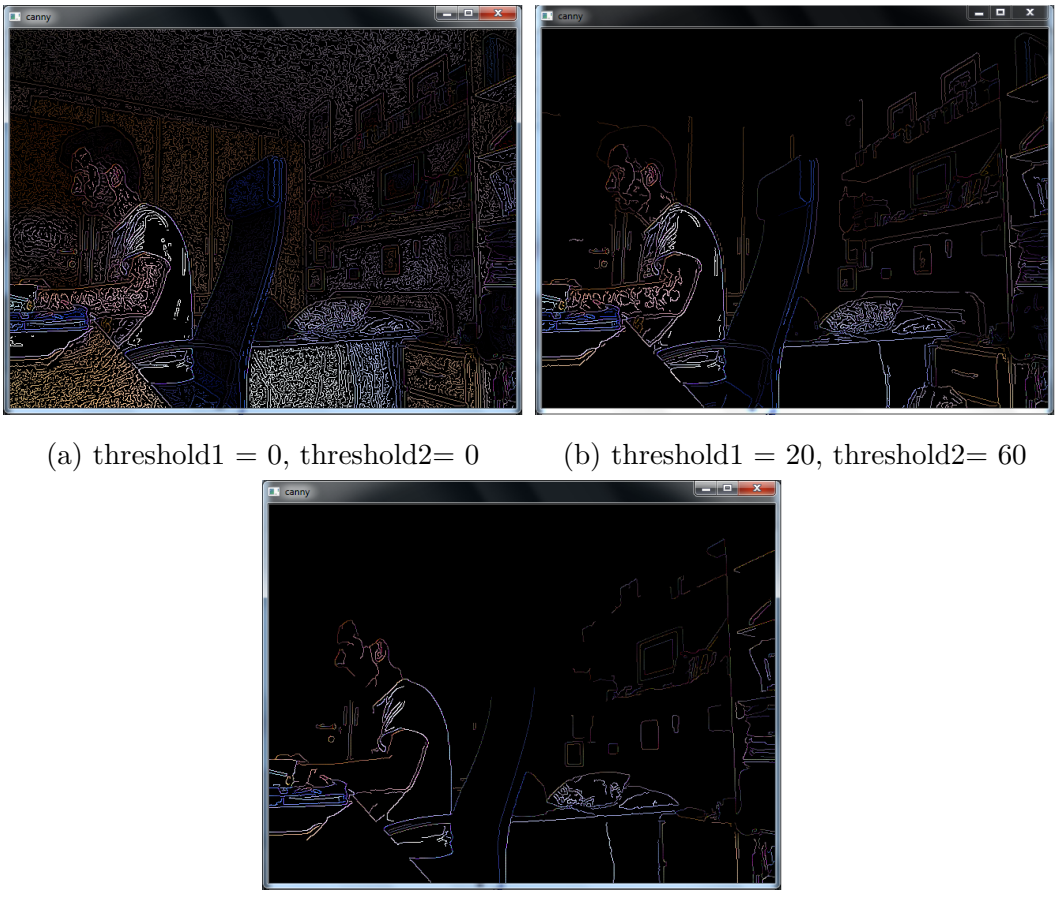
e un esempio della sua applicazione viene mostrato nel seguente frammento di codice

```
1 Mat edgeCanny(Mat img, int tsh){
2   Mat imggray;
3   Mat edges;
4
5   cvtColor( img, imggray, CV_BGR2GRAY );
6
7   blur( imggray, edges, Size(3,3) );
8   Canny( edges, edges, tsh, tsh*3, 3 );
9 }
```

Alcuni aspetti da sottolineare: l'algoritmo viene eseguito su una versione in scala di grigi dell'immagine originale, in quanto la funzione prevede in input una immagine con un singolo canale, sulla quale viene inoltre precedentemente applicata una sfumatura. La sfumatura è necessaria per ridurre il rumore digitale ma ha anche un impatto diretto sull'esecuzione di Canny: una sfocatura effettuata con filtri piccoli consente di riconoscere contorni

più netti, mentre una sfocatura con filtri maggiori è indicata per riconoscere contorni ampi e sfumati.

Per determinare se estrarre o meno un contorno sono inoltre utilizzate due soglie: se il gradiente calcolato in ciascun punto è inferiore alla prima soglia, allora il punto viene scartato; se è superiore alla seconda soglia, viene invece accettato come parte di un contorno; infine, se il valore è compreso tra le due soglie, il punto viene accettato soltanto se contiguo ad un punto già selezionato come facente parte del contorno.



(a) threshold1 = 0, threshold2= 0

(b) threshold1 = 20, threshold2= 60

(c) threshold1 = 60, threshold2= 120

Figura 5.6: Esempio di applicazione dell'edge detector Canny con diverse soglie

Definito il tipo di edge detector utilizzato, è possibile illustrare il funzionamento del primo approccio. Vengono presi in considerazione i bordi della depth map e quelli individuati dall'edge detector: se il bordo della prima dista al massimo una certa distanza dal bordo del secondo, allora si suppone che la depth map sia imprecisa e si procede quindi ad estenderla fino ad arrivare al bordo dell'edge detector.

In particolare, la procedura utilizzata è la seguente:

1. viene applicato l'edge detector per rilevare i bordi
2. i bordi individuati vengono spezzati in frammenti aventi la stessa direzione
3. per ogni frammento viene calcolata, nei due versi ortogonali alla direzione del frammento, la distanza dalla depth map
4. se la distanza è inferiore alla soglia, si estende il bordo della depth map fino a quello rilevato dall'edge detector

Questo primo approccio ha prodotto risultati migliori soltanto per soggetti fissi ma non per quelli in movimento: questo perché anche la più piccola variazione nella posizione del soggetto può cambiare la direzione media del bordo della depth map, rendendola non più allineabile con il bordo individuato da Canny. Questo effetto si manifestava in un costante *lampeggiamento* visibile su soggetti in movimento.

5.3.3 Approccio 2: Algoritmo di Canny ed estensione con BFS

Poiché calcolare la distanza ortogonalmente alla direzione del bordo non si è dimostrata una soluzione adeguata alla situazione, è stato valutato un differente algoritmo di estensione della depth map. La correzione avviene nel seguente modo:

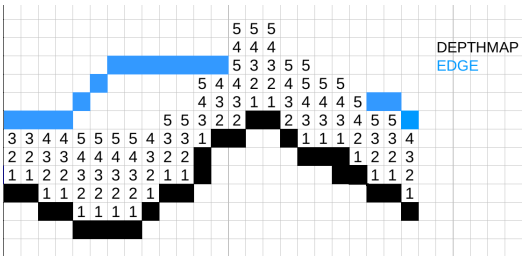
- partendo dai punti di bordo della depth map viene effettuata una visita BFS con distanza limite, con l'intento di raggiungere il bordo più vicino trovato da Canny. Siano T la lista dei punti visitati, cioè i possibili candidati a diventare punti facenti parte della depth map, A la lista degli eventuali punti raggiunti e contigui ad un bordo, B la lista dei punti finali visitati non contigui ad alcun bordo. In figura 5.7a si può vedere un esempio di visita con distanza massima pari a 5, e in figura 5.7b le relative composizioni delle liste di punti

- per quanto riguarda i punti memorizzati in B, sono possibili due casi: il bordo è presente ma troppo distante per essere raggiunto dalla visita con la distanza inizialmente assegnata, oppure il bordo è assente. Per individuare i punti del primo caso viene effettuata una ulteriore BFS, partendo dai punti della lista B ed espandendosi solo verso punti non in T, non facenti parte dei bordi dati da Canny e non facenti parte della depth map. Le nuove aree rilevate più piccole di una certa soglia vengono inserite in T, i punti di partenza rimossi da B e i punti di arrivo sul bordo aggiunti in A (figura 5.7c)

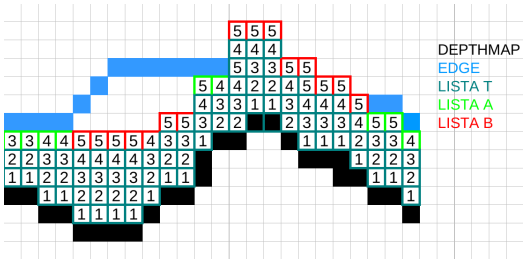
- a questo punto, i punti rimanenti in B sono quelli per i quali non esiste un bordo trovato da Canny: nello spazio compreso tra essi e il bordo della depth map, quest'ultima non deve essere estesa. Tali punti vengono quindi selezionati ed eliminati, sfruttando i punti di A come nuovo confine oltre il quale non eliminare (figura 5.7d)

- i punti compresi tra B e il nuovo confine delineato al passo precedente non vanno considerati: viene quindi effettuata una BFS partendo da B e utilizzando come limite i punti di confine precedentemente calcolati più i bordi della depth map. I punti visitati in tale modo vengono rimossi da T (figura 5.7e)

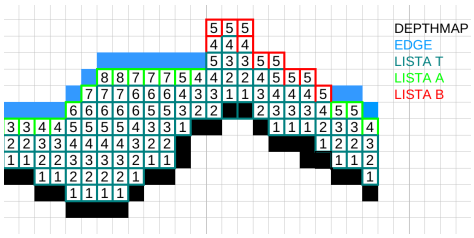
- infine, la depth map viene estesa su tutti i punti di T (figura 5.7f)



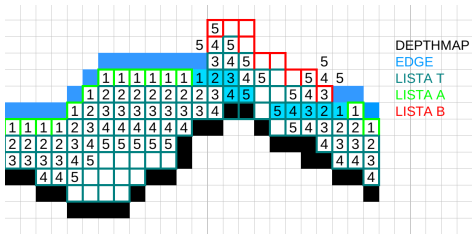
(a) Punti visitati con distanza massima pari a 5



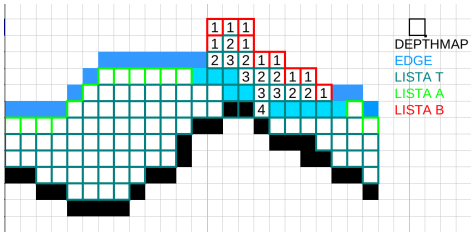
(b) Contenuto delle liste T, A e B



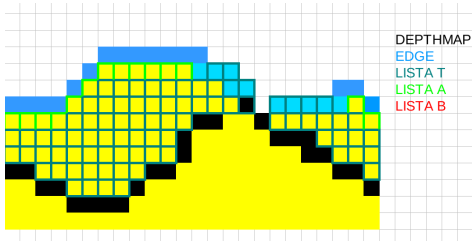
(c) Stato dopo la seconda visita partendo dai punti in B



(d) Eliminazione dei punti tra B e il bordo della depth map



(e) Rimozione dei punti superflui tra B e il nuovo confine



(f) Stato al termine dell'algoritmo, in giallo la depth map corretta

Figura 5.7: Esempio di applicazione dell'algoritmo di correzione mostrato nel secondo approccio

Il procedimento appena esposto funziona correttamente soltanto se il bordo trovato da Canny si trova all'esterno della depth map, ma non se si trova invece all'interno. Poiché in generale non è detto che la depth map sia sbagliata sempre in eccesso piuttosto che sempre in difetto, si applica nuovamente l'algoritmo invertendo i ruoli dell'area di sfondo e della depth map: in questo modo gli eventuali bordi trovati da Canny si trovano nuovamente all'esterno della depth map. Al termine del processo la depth map viene invertita nuovamente, in modo che i punti che l'algoritmo ha deciso di aggiungere nell'ultimo passaggio diventino punti eliminati nella depth map originale.

La modalità di correzione vista fin'ora dipende sostanzialmente dalla capacità di Canny di trovare o meno dei bordi corretti. Come anticipato in precedenza, il risultato prodotto da Canny è influenzato dalla soglia scelta: eseguendo l'algoritmo con una soglia bassa vengono trovati numerosi bordi che spesso sono però errati, mentre l'esecuzione con una soglia alta produce invece pochi bordi, che tendenzialmente sono però più affidabili.

Per questo motivo la correzione sopra citata viene in realtà reiterata più volte: in partenza si esegue Canny con una soglia alta e si effettua la correzione utilizzando i bordi trovati, quindi si procede abbassando la soglia e rieseguendo la correzione sugli eventuali punti rimasti non ottimizzati al passo precedente. Questo metodo conservativo è giustificato dal fatto che è ragionevole tentare inizialmente di correggere utilizzando bordi considerati sicuri, e solo successivamente accettare di correggere con possibilità di errori in quelle aree che comunque al passo precedente non erano state corrette.

Algoritmo 5.2 Esecuzione reiterata della correzione. Soglia iniziale pari a 100 e valore calato di 10 ad ogni step

```
1 Mat edgefirst;
2 vector<pair<int,int> > startpoints;
3 int starttsh = 100;
4 for(int tsh=starttsh;tsh>0;tsh-=10){
5   Mat bordi = edgeCanny(img,imggray,tsh);
```

```
6    pair<Mat,vector<pair<int,int> > > resulttemp = correggi(depth,
         frame, bordi,false,tsh!=starttsh,startpoints);
7    pair<Mat,vector<pair<int,int> > > resulttemp2 =
         correggi(resulttemp.first,frame,bordi,true,tsh!=starttsh,resulttemp.second);
8    if( tsh == starttsh){
9       edgefirst = bordi;
10   }
11   depth = resulttemp2.first;
12   startpoints = resulttemp2.second;
13   if( startpoints.size() == 0 )break;
14 }
```

Il procedimento presentato in questo paragrafo ha mostrato miglioramenti rispetto il primo approccio proposto. Tuttavia, il risultato mostra ancora dei limiti in particolare dal punto di vista della stabilità, cioè in certe situazioni si verifica un effetto di lampeggiamento della depth map. In particolare si consideri la seguente situazione: supponiamo sia presente un oggetto A, i cui bordi vengono sempre rilevati, ed un oggetto B, i cui bordi vengono rilevati con valori di soglia molto vicina a quella impostata inizialmente, per cui nel tempo l'algoritmo Canny a volte rileva i suoi bordi e a volte no. Nel caso i bordi di A si intersechino con B e A si trovi posizionato dietro B, l'algoritmo a volte ottimizza verso i bordi di A e a volte verso i bordi di B: questo determina quell'effetto di lampeggiamento citato prima.

Questo comportamento è determinato dal risultato dell'edge detector Canny, il quale nonostante sia comunemente riconosciuto come miglior algoritmo di edge detection, in questo contesto di applicazione si dimostra troppo rigido nella distinzione tra bordo rilevato o non rilevato, risultando non stabile tra un frame ed il successivo. E' stato quindi sviluppato un terzo approccio, basato su un differente algoritmo di edge detection, che verrà illustrato nel prossimo paragrafo.

5.3.4 Approccio 3: Operatore Sobel e Algoritmo Canny

Sobel [35] è un operatore differenziale utilizzabile in algoritmi di edge detection. Tale operatore calcola, per ogni pixel, il gradiente della luminosità dell'immagine, indicando la direzione lungo la quale è presente il massimo incremento possibile da chiaro a scuro, e la velocità del cambiamento. In altre parole, Sobel calcola per ogni punto una stima di quanto repentinamente o gradualmente l'immagine stia cambiando, quindi in sostanza indica la probabilità che quella parte di immagine rappresenti un contorno o meno. Più specificatamente, l'operatore calcola i valori approssimati delle derivate, in direzione orizzontale e verticale, applicando due matrici di convoluzione di dimensione 3x3. Nell'equazione sottostante sono indicate le matrici utilizzate da Sobel per calcolare i valori approssimati delle derivate orizzontali e verticali: I è l'immagine originale e $*$ è l'operatore di convoluzione bidimensionale.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \tag{5.1}$$

Algoritmo 5.3 Applicazione della funzione Sobel di OpenCV nella direzione orizzontale e verticale

```
1 Sobel( src_gray, grad_x, ddepth, 1, 0, 3, scale, delta,
     BORDER_DEFAULT );
2 Sobel( src_gray, grad_y, ddepth, 0, 1, 3, scale, delta,
     BORDER_DEFAULT );
```

Nell'algoritmo sviluppato, si è scelto di eseguire l'operatore Sobel sulla singola componente RGB del pixel considerato: la probabilità finale del pixel viene quindi scelta come il valore massimo tra le probabilità nei canali

Red, *Green* e *Blue*. Nel codice sottostante vengono mostrate le funzioni che applicano Sobel in tal modo, e in figura 5.8 un esempio di risultato ottenuto.

Algoritmo 5.4 Funzioni relative all'esecuzione dell'operatore di edge detection Sobel

```
1  Mat sobel2(Mat src){
2     Mat src_gray, Mat grad;
3     int scale = 1, int delta = 0, int ddepth = CV_16S;
4     GaussianBlur( src, src, Size(11,11), 0, 0, BORDER_DEFAULT );
5     cvtColor( src, src_gray, CV_RGB2GRAY );
6     Mat grad_x, grad_y;
7     Mat abs_grad_x, abs_grad_y;
8     Sobel( src_gray, grad_x, ddepth, 1, 0, 3, scale, delta,
          BORDER_DEFAULT );
9     convertScaleAbs( grad_x, abs_grad_x );
10    Sobel( src_gray, grad_y, ddepth, 0, 1, 3, scale, delta,
          BORDER_DEFAULT );
11    convertScaleAbs( grad_y, abs_grad_y );
12    addWeighted( abs_grad_x, 0.5, abs_grad_y, 0.5, 0, grad );
13    Mat ret = Mat::zeros(grad.rows,grad.cols,CV_8UC3);
14    for(int i=0;i<grad.rows;i++){
15       for(int j=0;j<grad.cols;j++){
16          if( grad.at<unsigned char>(i,j) ){
17             ret.at<Vec3b>(i,j)[0] = grad.at<unsigned char>(i,j);
18             ret.at<Vec3b>(i,j)[1] = grad.at<unsigned char>(i,j);
19             ret.at<Vec3b>(i,j)[2] = grad.at<unsigned char>(i,j);
20          }
21       }
22    }
23        return ret;
24 }
25
26 Mat sobel(Mat img){
```

```
27    Mat R = Mat::zeros(img.rows,img.cols,CV_8UC3);
28    Mat G = Mat::zeros(img.rows,img.cols,CV_8UC3);
29    Mat B = Mat::zeros(img.rows,img.cols,CV_8UC3);
30    for(int i=0;i<img.rows;i++){
31       for(int j=0;j<img.cols;j++){
32          R.at<Vec3b>(i,j)[0] = R.at<Vec3b>(i,j)[1] =
                R.at<Vec3b>(i,j)[2] = img.at<Vec3b>(i,j)[2];
33          G.at<Vec3b>(i,j)[0] = G.at<Vec3b>(i,j)[1] =
                G.at<Vec3b>(i,j)[2] = img.at<Vec3b>(i,j)[1];
34          B.at<Vec3b>(i,j)[0] = B.at<Vec3b>(i,j)[1] =
                B.at<Vec3b>(i,j)[2] = img.at<Vec3b>(i,j)[0];
35       }
36    }
37    Mat s1 = sobel2(R);
38    Mat s2 = sobel2(G);
39    Mat s3 = sobel2(B);
40    Mat ret = Mat::zeros(img.rows,img.cols,CV_8UC3);
41    for(int i=0;i<img.rows;i++){
42       for(int j=0;j<img.cols;j++){
43          int a = s1.at<Vec3b>(i,j)[0];
44          int b = s2.at<Vec3b>(i,j)[0];
45          int c = s3.at<Vec3b>(i,j)[0];
46          ret.at<Vec3b>(i,j)[0] = ret.at<Vec3b>(i,j)[1]
                =ret.at<Vec3b>(i,j)[2] = (a>b&&a>c)? a : (b>c?b:c);
47       }
48    }
49    return ret;
50 }
```

L'algoritmo che effettua la correzione della depth map si occupa di spostarne i bordi verso i punti di massimo locale del gradiente calcolato da Sobel. La correzione avviene in tre fasi:

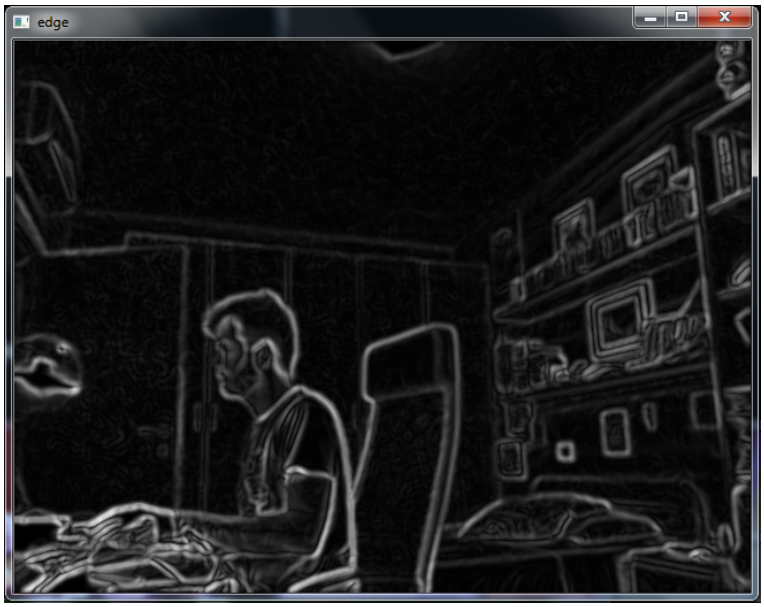


Figura 5.8: Un esempio di applicazione di Sobel

1. vengono corretti i punti per i quali esisteva un bordo esterno alla depth map

2. vengono riempiti i buchi rimasti nella depth map aventi un'area inferiore ad una certa soglia

3. vengono corretti i punti per i quali esisteva un bordo interno alla depth map

Algoritmo 5.5 Pseudocodice dell'algoritmo di correzione dei bordi della depth map

```
1 Mat depthMap
2 Mat edgeSobel = sobel(img);
3
4 T = puntiSulBordo(depthMap);
5 while ( !T.empty() ){
6   p = T.pop();
```

```
7     Per ogni vicino v di p {
8        if v not in depthMap && edgeSobel.at(v) > edgeSobel.at(p) {
9           depthMap.add(v);
10          T.add(v);
11       }
12    }
13 }
14
15 inpainting(depthMap, tshHoles);
16
17 T = puntiSulBordo(depthMap);
18 while ( !T.empty() ){
19    p = T.pop();
20    Per ogni vicino v di p {
21       if v not in depthMap && edgeSobel.at(v) < edgeSobel.at(p) {
22          depthMap.remove(p);
23          T.add(v);
24       }
25    }
26 }
```

L'algoritmo appena presentato ottiene in generale risultati migliori di quello basato sull'utilizzo di Canny, tranne in alcune aree specifiche come ad esempio la testa delle persone. I capelli infatti creano problemi di riflessione e interpretazione del pattern IR, quindi nella zona attorno alla testa la qualità della depth map è in media inferiore al resto della scena.

Per questo motivo è stata aggiunta una seconda fase di correzione, specifica per l'area relativa alla testa delle eventuali persone presenti in scena. In questa fase viene invece utilizzato l'algoritmo di Canny, che per sua natura riesce a delineare in modo più efficace la zona dei capelli. Per individuare la posizione della testa viene utilizzato lo stream di dati di tipo *Skeleton* fornito dal Kinect, quindi l'applicazione di questa ulteriore correzione è subordinata al fatto che il Kinect abbia effettivamente riconosciuto la presenza di una o

più persone sulla scena.

Nel caso siano presenti i dati Skeleton, viene eseguito l'algoritmo Canny sull'immagine originale in scala di grigi, quindi si iniziano a valutare i punti sul bordo della depth map corretta nella fase precedente: nel caso il punto si trovi ad una distanza inferiore ad una certa soglia dai punti che il Kinect ha identificato come testa, si procede a calcolare quanti sono i pixel compresi tra la depth map e il bordo individuato da Canny. Se il numero di pixel individuati è inferiore ad una data soglia, allora si estende la depth map anche in tali punti.

Per riassumere, il seguente frammento di codice (semplificato di alcune parti) mostra l'algoritmo di miglioramento della depth map utilizzato e spiegato nel corso di questo paragrafo. In figura 5.9 viene invece mostrato il risultato della sua applicazione sui bordi di una depth map di esempio.

Algoritmo 5.6 Algoritmo di miglioramento della depth map utilizzato (Terzo approccio)

```
1  Mat depth = {depth map fornita dal Kinect};
2  Mat img = {frame RGB fornito dal Kinect}
3  Mat depthcut = depth.clone();
4  Mat edge = sobel(img);
5
6  // seleziona la parte di depth map desiderata
7  for(int i=0;i<frame.rows;i++){
8    for(int j=0;j<frame.cols;j++){
9      if( mapped.at<Vec3b>(i,j)[0] < cutdepth ){
10           depthcut.at<Vec3b>(i,j)[0] = 0;
11           depthcut.at<Vec3b>(i,j)[1] = 0;
12           depthcut.at<Vec3b>(i,j)[2] = 0;
13     }
14   }
15 }
16
```

```
17 Mat depthintermedia = algoritmoCorrezione(depthcut,edge);
18
19 Mat ec ;
20 if( ret->numSkel > 0 ){
21    Mat gray;
22    cvtColor( frame, gray, CV_BGR2GRAY );
23    ec = edgeCanny(img,gray,THScanny);
24    depthintermedia =
          algoritmoCorrezioneTeste(depthintermedia,ec,{punti relativi
          allo scheletro},{numero di scheletri individuati});
25 } else ec = Mat::zeros(depth.rows,depth.cols,CV_8UC3);
```

(a) La depth map originale priva di correzioni

(b) La depth map dopo l'algoritmo di correzione lungo i bordi

(c) In bianco è evidenziata la correzione effettuata dall'algoritmo

Figura 5.9: Risultato dell'algoritmo di correzione sui bordi della depth map

5.4 Composizione dei flussi video

La depth map migliorata può essere ora utilizzata per selezionare parte del contenuto dello stream proveniente dal Kinect, e fonderlo assieme ad un secondo flusso. Nel codice sviluppato, il secondo flusso in questione consiste in un video MP4 salvato in memoria, ma possono essere ugualmente usati i frame provenienti ad esempio da una seconda telecamera. In OpenCV è infatti presente una funzione `VideoCapture` che consente di catturare frame sia da videocamera sia da file. Chiaramente l'area del secondo flusso viene ridotta in modo che sia concorde alla dimensione 640x480 utilizzata dal Kinect.

Il secondo flusso assume quindi il ruolo di *sfondo*, mentre il frame RGB del Kinect assume il ruolo di *primo piano*: utilizzando la depth map come maschera di selezione viene creata una nuova `Mat`, nella quale per ogni pixel viene decisa la sua provenienza in accordo con la maschera di selezione.

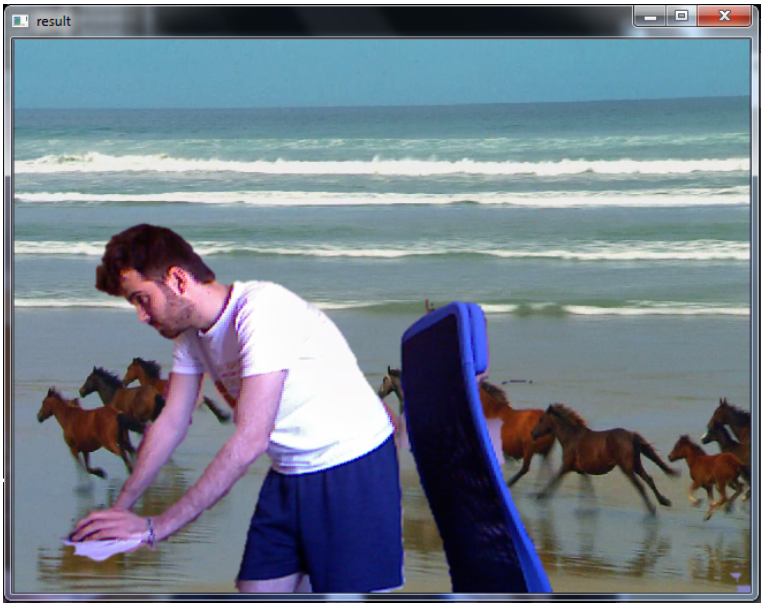


Figura 5.10: Composizione di parte del flusso video proveniente dalla telecamera RGB del Kinect (primo piano) con un video in memoria (sfondo)

5.5 Miglioramento del risultato

L'ultimo passaggio ha invece a che fare con l'applicazione di una leggera sfocatura lungo i bordi del soggetto estratto, per rendere il passaggio tra primo piano e nuovo sfondo più graduale. La sfocatura applicata è di tipo gaussiano:

```
1 GaussianBlur( merged, cutblur, Size( 2*nblur-1, 2*nblur-1 ), 0, 0
    );
```

L'entità dell'applicazione del filtro è determinata da un valore `nblur` regolabile dall'utente.

5.6 Dettagli implementativi

Il sistema è stato sviluppato con un'architettura multithread: il calcolo di differenti frame viene distribuito su differenti core del computer. In particolare esiste una distinzione tra il thread che si occupa dell'acquisizione dei dati dal Kinect e i thread che effettuano invece i calcoli presentati.

Vengono utilizzate due strutture dati `workParams` e `workResult`, che rappresentano rispettivamente i dati recuperati dal Kinect relativi al singolo frame, sui quali vanno eseguiti i calcoli, e il risultato dell'applicazione degli algoritmi. Vengono mantenute due liste di tali strutture: `list<workParams> workToDo` e `list<workResult> workResults`; il thread di acquisizione dati aggiunge continuamente elementi (cioè frame) alla lista `workToDo`, dalla quale i thread di calcolo vanno a recuperare i dati ed effettuare i calcoli. E' presente un sistema di gestione della *critical section* che protegge l'accesso alle due liste di dati. Inoltre i thread di calcolo vengono addormentati finché la lista del lavoro da svolgere è vuota, e vengono svegliati dal thread di acquisizione non appena quest'ultimo ha aggiunto nuovi frame alla lista.

Infine, per evitare che tempi di computazione diversi per set di frame possano invertire l'ordine di visualizzazione finale (cioè ad esempio due frame arrivati originariamente nell'ordine A-B vengono invece mostrati nell'ordine

B-A, poiché A richiede più tempo di computazione di B), ogni elemento delle due liste viene marcato con un identificatore. Nella fase di visualizzazione del risultato, per impedire evidenti artefatti video, gli eventuali frame ordinati in modo errato vengono quindi scartati.

Capitolo 6

Una applicazione nel teatro multimediale

Il progetto presentato in questo elaborato fa parte di una serie di riflessioni nate durante il corso di Multimedia e Tecnologie Creative, su spunto del Signor Pietro Babina, un regista teatrale molto interessato all'utilizzo delle nuove tecnologie nel contesto della produzione teatrale. Il Signor Babina ci ha esposto la sua visione sulla grande opportunità che i mezzi tecnologici attuali, le tematiche tipiche del mondo della computer vision, della realtà aumentata ed in generale della multimedialità, possano offrire nell'ambito dello spettacolo dal vivo.

In particolare, è stato illustrato uno dei principali progetti al qualeBabina, assieme al suo staff, sta lavorando da diversi anni: realizzare uno spettacolo teatrale nel quale lo spettatore è parte attiva durante la presentazione dello spettacolo, avendo la possibilità di compiere, in una certa misura, scelte autonome che influenzino l'esperienza personale di fruizione dello spettacolo.

Una delle prime tipologie di interazione ipotizzata da Babina riguarda l'impianto scenografico, o per meglio dire l'ambiente in cui gli attori compiono le azioni. L'obbiettivo ricercato è fornire allo spettatore una duplice fruizione dello spettacolo teatrale: quella relativa al palco, sul quale gli attori recitano dal vivo, e quella fruita attraverso il dispositivo, la quale non

deve necessariamente essere congruente alla precedente e attraverso la quale diventa possibile, per lo spettatore, interagire. In sostanza si concede la possibilità allo spettatore, per mezzo dell'utilizzo di dispositivi personali quali *smartphone* o *tablet*, di cambiare, all'interno del dispositivo, l'ambientazione dell'azione scegliendola tra una serie di proposte significative dal punto di vista della narrazione, studiate durante fase di produzione dello spettacolo.

L'effetto che si vuole perseguire è quello di una direzione teatrale volutamente *anticomunicativa*, che lasci la possibilità allo spettatore di penetrare nello svolgimento dello spettacolo e di crearsi così una propria prospettiva di visione assolutamente soggettiva.

6.1 Architettura proposta

Dal punto di vista tecnologico, il mezzo iniziale ipotizzato per permettere un cambio di scenografia alle spalle degli attori era chiaramente basato su chroma keying e green/blue back, con conseguente presenza del fondale sul set. Partendo da questa considerazione, è stato pensato e sviluppato il progetto mostrato in questa tesi, che nel contesto teatrale trova una sua applicazione specifica.

E' stata infatti ipotizzata la realizzazione di un *modulo di calcolo*, cioè un sistema formato da una serie di apparecchi tecnologici hardware e software, che implementano l'estrazione digitale tramite profondità. Tale sistema sarebbe formato da:

Kinect posizionato in prossimità del palco, che diventa un punto di ripresa dello spettacolo teatrale

Tablet di controllo relativo al Kinect, che mostra all'operatore la scena attualmente ripresa dal dispositivo

Mini PC/sistema single board che si occupa dell'impacchettamento su Ethernet dello stream proveniente da Kinect, e della sua trasmissione via cavo all'unità di elaborazione

Unità di elaborazione sulla quale vengono eseguiti gli algoritmi di miglioramento della depth map e la composizione dei flussi

L'unità di elaborazione del modulo di calcolo verrebbe poi collegata via Ethernet al *computer di regia*, nel quale l'operatore avrebbe la possibilità di selezionare quali flussi video trasmettere ai dispositivi degli spettatori, impostare la profondità con la quale estrarre dalla scena, e altre funzionalità. L'idea che sta alla base di una struttura di questo tipo è la *modularità*, cioè la possibilità di poter aumentare a piacere il numero di moduli di calcolo, e quindi di possibilità di ripresa e visione da fornire durante lo spettacolo. La figura 6.1 schematizza graficamente la struttura esposta.

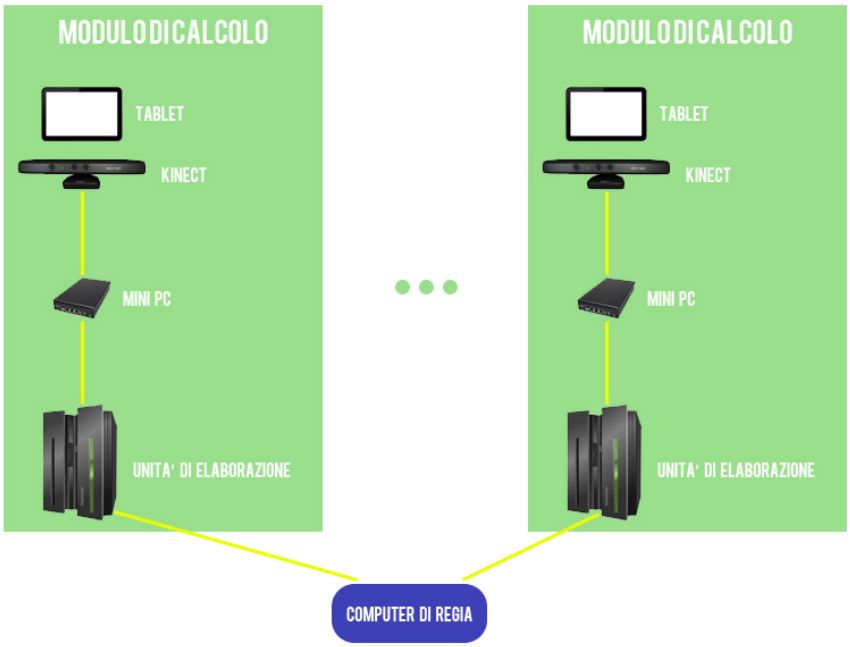


Figura 6.1: Componenti del modulo di calcolo utilizzabile in un contesto teatrale

Dal computer di regia verrebbe poi eseguito lo streaming dei flussi video verso i dispositivi presenti in sala: sulla realizzabilità dello streaming e la relativa applicazione per smartphone/tablet, per poter visualizzare il flusso

video, si è occupato invece un secondo gruppo di lavoro, quindi questo aspetto non verrà discusso in questo elaborato.

Capitolo 7

Test effettuati e risultati

Per valutare quanto l'approccio proposto in questo lavoro di tesi fosse efficace, si è in primo luogo fatto riferimento ai requisiti che sin dall'inizio ci si era imposti di soddisfare: in particolare, la capacità di operare in tempo reale e una qualità sufficiente per ottenere un effetto paragonabile al classico chroma keying, avendo però meno vincoli di installazione del set ed eventualmente una flessibilità maggiore.

Il codice sviluppato è stato eseguito su tre diverse configurazioni hardware, le cui caratteristiche principali sono riportate in tabella 7.1.

	Macchina 1	Macchina 2	Macchina 3
Processore	Intel® Core™ i7-4800MQ cPU @ 3.70GHz	Intel® Core™ i7-2600K CPU @ 3.40GHz	Intel® Core™ 2 Duo CPU @ 2.50GHz
Ram	16GB DDR3, 1600Mhz	8GB DDR3, 1600Mhz	4GB
Scheda grafica	NVIDIA® GeForce® GTX 780M	NVIDIA® GeForce® GeForce GTX 570	NVIDIA® GeForce® 8400M GS

Tabella 7.1: Principali caratteristiche delle macchine sulle quali è stato eseguito il codice

7.1 Numero di frame per secondo

Per quanto riguarda la condizione di tempo reale, va valutato l'impatto che hanno gli algoritmi sviluppati sul numero di frame per secondo prodotti. La complessità asintotica temporale di tutti gli algoritmi è in generale *lineare nel numero di pixel*, in quanto la fase dominante consiste sempre nello scorrimento di una matrice di $640 * 480 = 307200$ pixel. Per quanto riguarda invece gli algoritmi già implementati in OpenCV, ed in particolare Canny e Sobel, non è chiaramente esposto nella documentazione quale sia la loro complessità, ma verosimilmente dovrebbe trattarsi in entrambi i casi di una complessità pari a *O(n log n)* con n pari al numero di pixel, in quanto sono eseguite operazioni di convoluzione tra matrici e filtri di dimensioni fissata, che se implementate con *FastFourierTransform* hanno appunto tale complessità.

La seguente tabella 7.2 riporta il valore medio, calcolato su un minuto di esecuzione nelle tre diverse configurazioni hardware, del numero di fps prodotti, con relativo numero di thread di calcolo utilizzati.

Computer	**# di fps medio su 60s**	**# thread di calcolo**
Macchina 1	20	2
Macchina 2	23	4
Macchina 3	4	2

Tabella 7.2: Risultati relativi al numero di fps su diverse configurazione hardware

Variando il numero di thread su *Macchina 2*, la quale consiste in un desktop quad-core, è stato in particolare notato che 4 thread di calcolo risulta essere la configurazione ottimale (7.3).

Dall'analisi sul numero di fps emerge quindi che con una configurazione hardware recente e piuttosto performante, quali ad esempio *Macchina 1* e *Macchina 2*, è possibile soddisfare il requisito di operatività in tempo reale, producendo un numero di frame tutto sommato sufficiente per poter essere percepito in modo fluido dall'occhio umano.

# thread di calcolo	# di fps medio su 60s
2	15
3	19
4	23
8	23

Tabella 7.3: Effetto del numero di thread sugli fps, Macchina 2

Tale risultato è soddisfacente soprattutto se valutato in confronto al numero di fps prodotti nativamente dal Kinect, che si ricorda essere di 30fps.

7.2 Qualità del risultato

Il secondo aspetto determinante per poter dare un giudizio sul lavoro svolto è ovviamente la qualità visiva del risultato finale. Come si è visto, la depth map iniziale prodotta dal Kinect presentava notevoli difetti e imprecisioni in particolare lungo i bordi, rendendola inapplicabile per l'estrazione di soggetti dalla scena, e sono quindi stati sviluppati una serie di algoritmi con l'obbiettivo di migliorarla.

Il modo più logico per valutare la loro efficacia sarebbe stato confrontare la depth map del Kinect, dopo la loro esecuzione, con una ipotetica depth map che si sa essere perfetta, o perlomeno avente un livello di precisione tale da renderla utilizzabile come metro di paragone: una depth map di questo tipo viene genericamente chiamata *ground truth*. Nel campo della stereo vision è pratica comune testare i diversi algoritmi per il calcolo di disparity map eseguendoli su determinati set di immagini, come il già citato in precedenza *Tsukuba*. Tali set sono in sostanza formati da varie immagini della scena, ripresa da punti di vista differenti, e dalla disparity map realizzata con accurati scanner, con una precisione tale da poter essere considerate come disparity ground truth. Gli sviluppatori possono quindi eseguire i propri algoritmi sulle immagini fornite, e confrontare la disparity map calcolata dal

proprio algoritmo con quella ground truth presente nel set: in questo modo è possibile stimare, eventualmente anche pixel per pixel, quale sia il livello di qualità raggiunto.

Nel caso del Kinect, invece, dal momento che è il dispositivo stesso a inquadrare la scena dalla quale ricavare la depth map e sulla quale vengono poi applicati gli algoritmi, non è possibile utilizzare il metodo spiegato sopra. Si è quindi deciso di mettere direttamente a confronto il risultato finale dell'estrazione con profondità, con la "classica" tecnica del chroma keying.

Sul mercato sono presenti diversi dispositivi, sia hardware che software, che realizzano un effetto di green screen molto realistico e di alta qualità, ma si tratta in generale di prodotti commerciali piuttosto costosi oppure che lavorano in post produzione, cioè sulle registrazioni effettuate, e non in tempo reale come invece si desidera. Inoltre, tali prodotti di terze parti utilizzano sorgenti diverse dal Kinect per il flusso video, quindi tentare di confrontare due risultati in quelle condizioni avrebbe introdotto una serie di problematiche da superare, quali ad esempio: risolvere il problema della diversa prospettiva e porzione di scena inquadrata, rimuovere la distorsione indotta dalle diverse lenti utilizzate, gestire una possibile resa differente dei colori.

Dal momento che durante lo studio del progetto era stato sviluppato del codice di test che implementa un effetto di chroma key utilizzando come sorgente una semplice WebCam USB, si è pensato di riutilizzare direttamente parte di quel codice: è stata effettuata una modifica per utilizzare come sorgente video il Kinect piuttosto che una WebCam esterna, e il codice è stato integrato nel sistema realizzato. In questo modo, il Kinect mostra contemporaneamente in output sia il risultato dell'estrazione tramite profondità, sia il risultato del chroma keying; inoltre, le immagini mostrate hanno esattamente le stesse caratteristiche, in quanto provengono dalla stessa telecamera RGB.

L'algoritmo di chroma keying introdotto consente di selezionare il colore che si desidera filtrare cliccando direttamente sul frame mostrato, oppure regolando manualmente i canali R, G e B; una volta selezionato il colore,

è possibile inoltre agire sulla tolleranza dei canali *Hue*, *Saturation* e *Value* (internamente all'algoritmo, lo spazio di colori utilizzato è infatti HSV), in modo da poter consentire una selezione più o meno precisa, a seconda della situazione. Sono inoltre presenti alcuni controlli aggiuntivi, analogamente a quanto avviene nella selezione con profondità, per applicare una sfocatura sui bordi oppure un loro restringimento.

7.2.1 Confronto con un chroma keyer implementato

Il primo aspetto che è emerso durante la realizzazione dei confronti, come si era già evidenziato dal punto di vista teorico, è la difficoltà di organizzare un set adeguato all'applicazione del chroma keying, in particolare se effettuato non in uno studio adeguato ma in un contesto domestico generico, che è il contesto nel quale si è dovuto operare.

Sono stati effettuati una serie di confronti, nei quali sono stati principalmente variati:

- tipo di illuminazione adoperata (*artificiale* oppure *naturale*, con diversi livelli di intensità)
- tipologia di scena inquadrata: oggetti statici oppure una persona in movimento

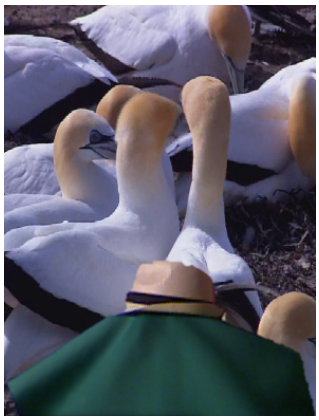
Oggetti statici con illuminazione artificiale

Il primo set molto elementare è stato realizzato con un lenzuolo bianco posizionato alle spalle di alcuni oggetti. Per illuminare tutta l'area dello sfondo è stato posizionato un proiettore alogeno da 400W, cercando di fare in modo che si venissero a creare poche ombre e che l'illuminazione fosse il più uniforme possibile.

In figura 7.1a viene mostrato il risultato dell'applicazione del chroma keyer: nonostante si sia cercato, variando opportunamente le tolleranze dei canali, di coprire tutta l'area del lenzuolo, rimangono comunque alcune zone di ombra che non vengono filtrate dall'algoritmo. A fianco, in figura 7.1b,

viene invece mostrato il risultato dell'estrazione con profondità. Dal punto di vista dei vincoli sul set, l'approccio con profondità risulta ovviamente più convincente, poiché non è necessario coprire l'intera area con il fondale e non si hanno problemi di ombre e illuminazione uniforme relativamente al fondale. In figura 7.1c si notano in particolare le ombre e le piccole aree non filtrate nel caso chroma keying.

(a) Risultato chroma keying con illuminazione artificiale



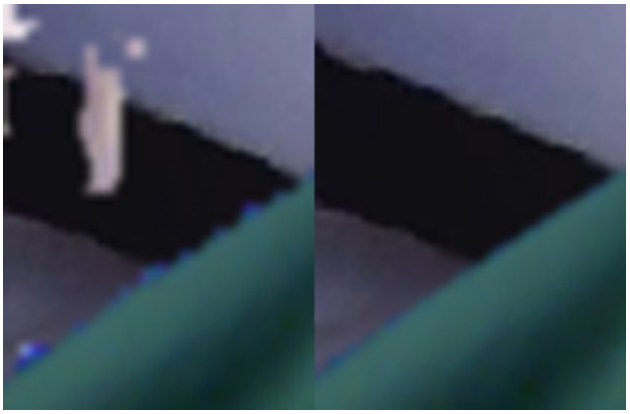
(b) Risultato estrazione profondità con illuminazione artificiale

(c) Differenza tra i due risultati: in bianco i pixel uguali, in nero quelli diversi

Figura 7.1: Confronto chroma keying e selezione con prondità: oggetti, illuminazione artificiale

Di seguito invece due ingrandimenti di alcuni dettagli relativi ai bordi degli oggetti, con a sinistra chroma keying e a destra l'estrazione con profondità: in figura 7.2a quest'ultima pare in generale ottenere bordi meno frastagliati, ma in quel particolare frame ha selezionato in modo errato una parte finale del cappello; in figura 7.2b si nota invece come il bordo ottenuto con gli algoritmi sviluppati sia di qualità decisamente migliore del rispettivo con chroma keying.

(a) Confronto tra chroma keying ed estrazione con profondità, dettaglio di un primo oggetto



(b) Confronto tra chroma keying ed estrazione con profondità, dettaglio del bordo di un secondo oggetto

Figura 7.2: Ingrandimento dei dettagli su alcuni oggetti

Oggetti statici con illuminazione ambientale

Nella seguente serie di confronti è stata invece valutata la capacità di adattamento del sistema alla quantità di luce disponibile. In altre parole, l'intento è cercare di capire quanto l'estrazione con profondità sia indipendente dall'illuminazione presente sulla scena.

Il proiettore è stato inizialmente posizionato sulla scena, e i controlli del chroma keyer e del sistema sviluppato sono stati adattati il più possibile alla condizione di illuminazione presente (figura 7.3). Mantenendo fisse tali impostazioni, è stato quindi spento il proiettore e aperta l'unica finestra presente nella stanza (la luce proviene dal lato destro delle immagini proposte), figura 7.4. Nella figura 7.5 la finestra è stata parzialmente chiusa, riducendo la quantità di luce presente, e infine è stata chiusa quasi completamente (figura 7.6).

Figura 7.3: Risultati ottenuti con luce artificiale

Figura 7.4: Risultati ottenuti con luce proveniente da una finestra completamente aperta

Figura 7.5: Risultati ottenuti con luce proveniente da una finestra parzialmente aperta

Figura 7.6: Risultati ottenuti con luce proveniente da una finestra quasi del tutto chiusa

Come ci si aspettava, conseguentemente ai vincoli necessari per il funzionamento di un chroma keyer, già nel passaggio da luce artificiale ad ambientale si ha un calo drastico della capacità di ottenere un buon risultato: del lenzuolo utilizzato come sfondo, nel secondo caso ne viene filtrata a mala pena la metà. L'estrazione con profondità non pare invece particolarmente influenzata da questo passaggio, che mantiene sostanzialmente lo stesso livello di qualità, lungo i bordi, del caso precedente.

Questo andamento viene confermato anche nei passaggi successivi, fino ad arrivare al caso limite con quantità di luce ridotta: nonostante, per ovvi motivi, la qualità dell'immagine sia pessima, con evidente presenza di notevole *rumore digitale*, l'estrazione con profondità continua a funzionare e selezionare correttamente la parte di immagine richiesta. Un sistema basato su chroma keying avrebbe invece richiesto, ad ogni piccolo cambiamento di luce, una nuova regolazione delle impostazioni; in più, in ogni caso, in mancanza di sufficiente illuminazione non avrebbe prodotto i risultati aspettati.

Figura umana con illuminazione mista

Nelle seguenti immagini viene invece proposta una scena nella quale è presente una persona in movimento, con illuminazione in parte composta da un lampadario a soffitto e in parte da luce che entra dalla finestra. L'estrazione con profondità ottiene in media dei bordi più "lisci" e naturali del chroma keying; tuttavia, in alcune parti gli algoritmi non riescono a correggere efficacemente il bordo, in particolare nelle zone dove due bordi distinti si trovano a distanza molto ravvicinata, come un braccio molto vicino al resto del corpo.

Figura 7.7: Confronto tra chroma keying ed estrazione con profondità, persona in movimento 1

Figura 7.8: Confronto tra chroma keying ed estrazione con profondità, persona in movimento 2

Figura 7.9: Confronto tra chroma keying ed estrazione con profondità, persona in movimento 3

7.2.2 Confronto con estrazione tramite depth map presente nel SDK 1.7

Tra i vari esempi di applicazioni realizzabili con il Kinect, nella versione 1.7 del SDK è incluso il codice che implementa un effetto di rimozione dello sfondo. L'estrazione del soggetto dallo sfondo avviene in realtà con modalità diverse dalla pura selezione per profondità: nell'esempio fornito nel SDK viene infatti specificatamente rimossa soltanto la persona inquadrata, e non un generico piano di profondità come avviene invece nel progetto sviluppato.

In figura 7.10 viene mostrato il risultato dell'esecuzione dell'esempio fornito nel SDK, mentre in figura 7.11 il risultato utilizzando il sistema sviluppato. Per rendere il confronto apprezzabile, come secondo flusso è stata utilizzata la stessa immagine adoperata nell'esempio. I due frame sono stati catturati in esecuzioni diverse ma nella medesima scena e condizione di illuminazione (nessun tipo di fondale e illuminazione naturale).

Figura 7.10: Esecuzione dell'esempio presentato nel SDK ufficiale, versione 1.7

Figura 7.11: Esecuzione del sistema sviluppato, utilizzando le stesse condizioni di esecuzione dell'esempio nel SDK

7.2.3 Indipendenza dalla tipologia di sfondo

Uno dei principali motivi per cui risulta interessante cercare di ottenere un buon risultato con la selezione per profondità, consiste nel fatto che non si è obbligati al montaggio di un set apposito, come nel caso del chroma keying. Diversamente dai test mostrati in precedenza, nelle immagini seguenti viene mostrato il risultato che si ottiene senza alcun tipo di preparazione della scena: i frame sono stati catturati nel soggiorno di una abitazione, senza alcun tipo di fondale posizionato alle spalle (figura 7.12).

I frame seguenti (figura 7.13) sono stati catturati nella scena mostrata sopra. La qualità del risultato è comprensibilmente peggiorata rispetto all'utilizzo di un fondale alle spalle, ma nel complesso l'estrazione avviene. In questo caso, la qualità dei bordi estratti dipende molto dalla quantità di contorni che l'edge detector rileva alle spalle del soggetto.

Figura 7.12: Lo sfondo utilizzato per il test senza alcun tipo di set preparato.

(a) Estrazione su sfondo eterogeneo, frame 1

(b) Estrazione su sfondo eterogeneo, frame 2

Figura 7.13: Esempio di esecuzione senza set preparato precedentemente

7.3 Test Mean Opinion Score

Il risultato ottenuto dal sistema è stato sottoposto ad un test di tipo *MOS*, nel quale una serie di parametri selezionati vengono proposti al giudizio di un campione di persone. In particolare il test è stato effettuato con un insieme di 7 persone, estranee allo sviluppo del progetto, alle quali è stato mostrato il sistema in esecuzione per un periodo di 20-30 secondi. E' stato chiesto loro di esprimere un giudizio relativamente ai seguenti tre parametri individuati:

qualità generale del risultato cioè quanto sia ritenuto convincente l'effetto di estrazione ottenuto tramite il Kinect

qualità specifica lungo i bordi cioè quanto sia ritenuto preciso lo scontornamento lungo i bordi del soggetto selezionato

stabilità nel tempo cioè con quale entità si ritiene che il risultato cambi durante la successione dei frame

La scala di valutazione che è stato chiesto loro di rispettare è riportata in tabella 7.4.

MOS	Stabilità nel tempo	Qualità generale	Qualità lungo i bordi
5	Sempre stabile	Ottima	Ottima
4	Decisamente stabile	Buona	Buona
3	Sufficientemente stabile	Discreta	Discreta
2	Decisamente instabile	Scarsa	Scarsa
1	Totalmente instabile	Pessima	Pessima

Tabella 7.4: Scala di valutazione dei parametri individuati per il test MOS

7.3.1 Valutazione del sistema realizzato

Il grafico 7.14 riporta la media aritmetica dei valori MOS ottenuti nel caso dell'estrazione di una persona in movimento; l'esecuzione è avvenuta con l'ausilio di un fondale posizionato alle spalle del soggetto.

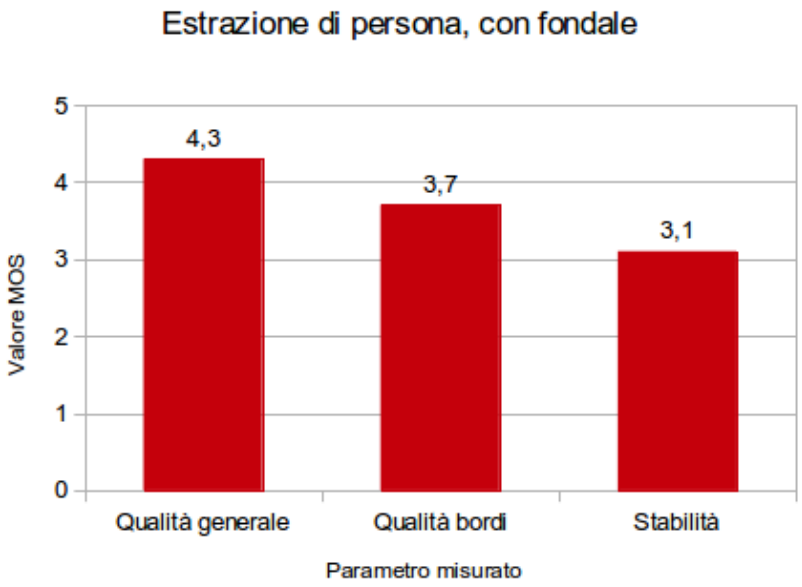


Figura 7.14: Risultato del test MOS relativo all'estrazione di una persona in movimento; esecuzione con l'ausilio di un fondale

Nel grafico 7.15 viene invece riportato il risultato nel caso dell'esecuzione senza alcun tipo di fondale alle spalle del soggetto.

Il giudizio delle persone intervistate conferma in sostanza che il sistema realizzato ottiene una qualità migliore nel caso venga aiutato da uno sfondo privo di molti oggetti (come ad esempio un fondale), in quanto questo semplifica il lavoro dell'edge detector; in uno scenario con sfondo eterogeneo, il sistema ottiene invece un punteggio inferiore, ma in media raccoglie comunque un generale giudizio di sufficienza.

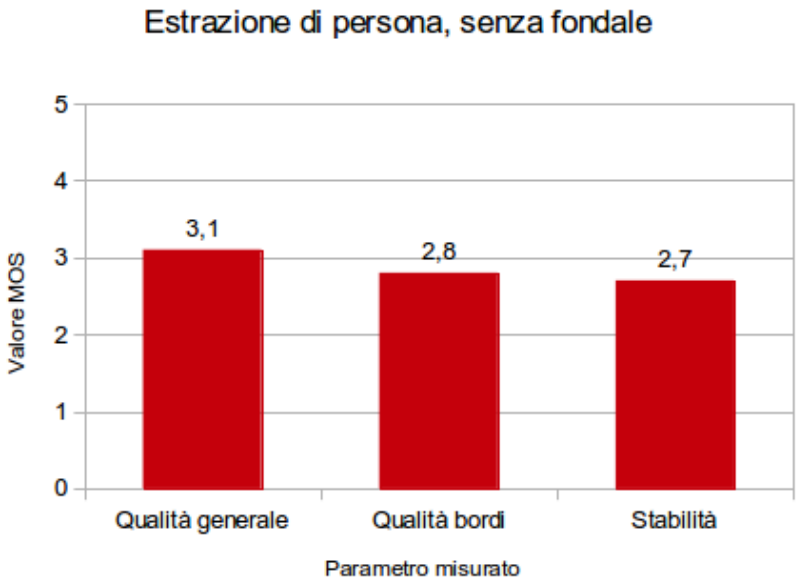


Figura 7.15: Risultato del test MOS relativo all'estrazione di una persona in movimento; esecuzione con sfondo eterogeneo

7.4 Le problematiche emerse

Il principale problema riscontrato nell'utilizzo dell'approccio con Kinect consiste nella bassa qualità, per il tipo di operazione che si desidera eseguire, della depth map fornita dall'apparecchio. Gli algoritmi sviluppati migliorano sensibilmente il risultato, che dipende però ancora sostanzialmente dal successo o meno dell'edge detector nel rilevare correttamente, e soprattutto con continuità, i contorni realmente presenti nell'immagine. Esistono quindi scenari, o per meglio dire sfondi alle spalle dei soggetti da estrarre, che in pratica funzionano meglio di altri: più lo sfondo è caotico, cioè pieno di oggetti e texture diverse, più l'edge detector ha difficoltà a rilevare correttamente i bordi, ed è quindi più probabile notare sbavature ed imprecisioni nell'estrazione operata.

Il secondo importante problema è la qualità del flusso RGB utilizzato: la telecamera RGB del Kinect registra soltanto ad una risoluzione pari a

640x480 pixel, che è verosimilmente troppo bassa per poter essere davvero utilizzata con successo in un contesto reale. L'alternativa sarebbe stata l'utilizzo di una sorgente diversa per il flusso RGB, quale ad esempio una seconda telecamera di qualità elevata: questa possibilità è stata inizialmente considerata, ma avrebbe significato aggiungere inevitabilmente una importante fase di calibrazione tra la telecamera IR del Kinect e l'ipotetica telecamera ad alta qualità, con il rischio che una calibrazione non estremamente precisa avrebbe vanificato tutto il lavoro degli algoritmi (si ricordi che, per l'applicazione della depth map al flusso RGB, è necessaria una corrispondenza a livello del singolo pixel). Lavorare inoltre con frame di risoluzione molto maggiore a 640x480 pixel, come ad esempio una risoluzione HD, avrebbe verosimilmente impedito l'operatività in tempo reale del codice attualmente sviluppato.

Conclusioni e sviluppi futuri

In questo lavoro di tesi è stato mostrato il progetto, realizzato con alcuni colleghi del corso di Multimedia e Tecnologie Creative, e i risultati ottenuti dalla sua applicazione. E' stato mostrato come sia possibile, utilizzando il dispositivo Kinect, effettuare una estrazione di soggetti usando come chiave di selezione la profondità: sono stati evidenziati i limiti impliciti della depth map prodotta dal Kinect ed è stato proposto un approccio completo, dalla fase di acquisizione iniziale dei dati fino alla composizione finale, che permette di migliorare sensibilmente la qualità lungo i bordi della depth map e ottenere quindi un effetto di keying paragonabile a quello realizzato con un chroma keyer; per certi aspetti, in particolare vincoli di realizzazione del set e illuminazione, è stato anche mostrato come tale approccio presenti dei vantaggi.

Permangono una serie di difetti, che sono principalmente da imputare a limiti dello strumento Kinect, che rendono un sistema di questo tipo ancora difficilmente utilizzabile in uno scenario di applicazione reale. Allo stesso tempo, gli incoraggianti risultati ottenuti già in questo iniziale livello di studio rendono cautamente ottimisti sulla possibilità di migliorare ancora il prodotto finale. In particolare, questo potrebbe avvenire seguendo due principali vie:

- una ottimizzazione generale del codice, che potrebbe portare ad un incremento nel numero di fps prodotti; tale incremento potrebbe eventualmente essere speso in ulteriori affinamenti dei bordi, permanendo comunque in un contesto di esecuzione in tempo reale. A questo proposito andrebbe valutata anche la possibilità di parallelizzare su GPU parte

del codice prodotto.

- l'utilizzo del *Kinect One*, la nuova versione annunciata durante lo sviluppo del progetto, attualmente non ancora disponibile per l'utilizzo su personal computer, che dovrebbe essere in grado di produrre una depth map decisamente migliore di quella calcolata con il modello precedente; la telecamera del dispositivo produce inoltre frame con risoluzione Full HD, un ulteriore vantaggio rispetto alla situazione attuale.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] P. Kalaiarasi Anamika Tiwari, Geetha Lakshmi Basker. Fpga implementation for real time chroma-key effect using adaptive filter. In *Proceedings of 4th IRF International Conference*, 2014.

[2] Apple. Final cut pro x: Use chroma keys.

[3] P. Benavidez and M. Jamshidi. Mobile robot navigation and target tracking system. In *System of Systems Engineering (SoSE), 2011 6th International Conference on*, pages 299–304, June 2011.

[4] PaulJ. Besl. Active, optical range imaging sensors. *Machine Vision and Applications*, 1(2):127–152, 1988.

[5] Adrian Bradski. *Learning OpenCV, [Computer Vision with OpenCV Library ; software that sees]*. O'Reilly Media, 1. ed. edition, 2008. Gary Bradski and Adrian Kaehler.

[6] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov 1986.

[7] Yao-Jen Chang, Shu-Fang Chen, and Jun-Da Huang. A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in Developmental Disabilities*, 32(6):2566 – 2570, 2011.

[8] Yung-Yu Chuang. *New Models and Methods for Matting and Compositing*. PhD thesis, University of Washington, 2004.

[9] Klaus Denker and Georg Umlauf. Accurate real-time multi-camera stereo-matching on the gpu for 3d reconstruction. *Journal of WSCG*, 19(1):9–16, 2011.

[10] D. Fofi, T. Sliwa, and Y. Voisin. A comparative survey on invisible structured light. In J. R. Price and F. Meriaudeau, editors, *Machine Vision Applications in Industrial Inspection XII*, volume 5303 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 90–98, May 2004.

[11] J. Garcia and Z. Zalevsky. Range mapping using speckle decorrelation, September 20 2007. US Patent App. 11/712,932.

[12] S. Burak Gokturk, Hakan Yalcin, and Cyrus Bamji. A time-of-flight depth sensor - system description, issues and solutions. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 3 - Volume 03*, CVPRW '04, pages 35–, Washington, DC, USA, 2004. IEEE Computer Society.

[13] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *Int. J. Rob. Res.*, 31(5):647–663, April 2012.

[14] Mesa Imaging. Sr4000 data sheet, 2011.

[15] Mesa Imaging. Sr4500 data sheet, 2013.

[16] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[17] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-flight sensors in computer graphics (state-of-the-art report). In *Proceedings of Eurographics 2009 - State of the Art Reports*, pages 119–134. The Eurographics Association, 2009.

[18] Medialooks. Chroma key.

[19] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. In *ICCV Workshops*, pages 467–474, 2011.

[20] Microsoft. Kinect for windows sdk v1.7.

[21] Microsoft. Kinect for windows sensor components and specifications.

[22] Microsoft. Kinect per xbox 360.

[23] Raymond A. Morano, Cengizhan Ozturk, Robert Conn, Stephen Dubin, Stanley Zietz, and Jonathan Nissanov. Structured light using pseudorandom codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):322–327, March 1998.

[24] PMD. Pmd photonics 19k-s3 preliminary datasheet, 2012.

[25] Zach Poff. Chroma key live.

[26] Reflecmedia. Chromatte and lite ring.

[27] M. Ribo and M. Brandner. State of the art on vision-based structured light systems for 3d measurements. In *Robotic Sensors: Robotic and Sensor Environments, 2005. International Workshop on*, pages 2–6, Sept 2005.

[28] Joaquim Salvi, Sergio Fernandez, Tomislav Pribanic, and Xavier Llado. A state of the art in structured light patterns for surface profilometry. *Pattern Recognition*, 43(8):2666 – 2680, 2010.

[29] Joaquim Salvi, Jordi Pagès, and Joan Batlle. Pattern codification strategies in structured light systems. *PATTERN RECOGNITION*, 37:827–849, 2004.

[30] D. Scharstein and R. Szeliski. Middlebury stereo evaluation.

[31] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, April-June 2002.

[32] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195–I–202 vol.1, June 2003.

[33] Christopher Schultz. Digital keying methods. 2006.

[34] Alvy Ray Smith and James F. Blinn. Blue screen matting. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 259–268, New York, NY, USA, 1996. ACM.

[35] I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. Never published but presented at a talk at the Stanford Artificial Project, 1968.

[36] Adobe Creative Team. *Adobe Creative Suite 6 Production Premium Classroom in a Book*. Adobe Press, 1st edition, 2012.

[37] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda. Near real-time stereo based on effective cost aggregation. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, Dec 2008.

[38] Djemel Ziou and Salvatore Tabbone. Edge detection techniques - an overview. *INTERNATIONAL JOURNAL OF PATTERN RECOGNITION AND IMAGE ANALYSIS*, 8:537–559, 1998.

Ringraziamenti

In primo luogo un ringraziamento alle persone che sono state coinvolte direttamente in questo lavoro di tesi: un grazie ad Alkida, Dennis e Andrea, amici e colleghi di Multimedia e Tecnologie Creative con i quali è stato sviluppato il progetto; grazie anche al Professor Marco Roccetti, che per primo ha proposto di realizzare questa tesi e mi ha aiutato con preziosi suggerimenti, e al signor Pietro Babina, che ha fornito lo spunto iniziale e al quale auguro di poter realizzare ogni progetto professionale.

Ringrazio inoltre tutta la mia famiglia: mi avete sempre supportato, so che quando serve siete dalla mia parte, e non è poco. Grazie a Rita, perché i caffè al bar non sono mai stati così interessanti: ne abbiamo davanti tanti altri. Grazie anche a tutti gli amici, quelli conosciuti durante questi cinque e più anni in Università, e quelli che da sempre ci sono. Un saluto a tutti voi.