Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

STUDIO ED IMPLEMENTAZIONE DI ALGORITMI DI STEREO VISION IN AMBITO MOBILE: L'APPLICAZIONE "SUPERSTEREO"

Tesi di Laurea in Multimedia e Tecnologie Creative

Relatore:
Chiar.mo Prof.
Marco Roccetti

Presentata da:
Alessandro Francesconi

Correlatore:
Dott. Gustavo Marfia

Sessione 3
Anno Accademico 2012 / 2013

A Luca e Piero, fedeli compagni in questa avventura
Alla mia Famiglia, per il supporto e la pazienza
A Francesca, per tutto il resto

"I'm personally convinced that computer science has a lot in common with physics. Both are about how the world works at a rather fundamental level. The difference, of course, is that while in physics you're supposed to figure out how the world is made up, in computer science you create the world. Within the confines of the computer, you're the creator. You get to ultimately control everything that happens.
If you're good enough, you can be God. On a small scale."

Linus Torvalds
(dal libro "Just for Fun", di L. Torvalds e D. Diamond)

Introduzione

La fotografia, inventata nei primi anni dell'Ottocento, è considerata una delle pietre miliari nello sviluppo della nostra società. La possibilità di racchiudere in un'immagine una rappresentazione realistica di ciò che ci circonda ha dato vita a fenomeni sociali, comunicativi, tecnologici ed artistici con i quali, ormai, interagiamo anche inconsapevolmente. Conseguentemente alla sua nascita, il progresso ha portato all'ideazione di strumenti fotografici sempre più evoluti. Pensati non solamente per migliorare la qualità dei colori e dei dettagli acquisiti, nuovi approcci hanno introdotto la possibilità di *analizzare* il contenuto di una fotografia, oltre che di apprezzarne l'aspetto estetico.

L'analisi di un'immagine con strumenti automatici si è dunque sviluppata in quella che oggi viene chiamata *computer vision*, la materia di studio proveniente dal mondo informatico che si occupa, letteralmente, di "vedere oltre", di estrarre da una figura una serie di aspetti strutturali, sotto forma di dati numerici.

Tra le tante aree di ricerca che ne derivano, una in particolare è dedicata alla comprensione di un dettaglio estremamente interessante, che si presta ad applicazioni di molteplici tipologie: la *profondità*. L'idea di poter recuperare ciò che, apparentemente, si era perso fermando una scena ed imprimendone l'istante in un piano a due dimensioni poteva sembrare, fino a non troppi anni fa, qualcosa di impossibile. Grazie alla cosiddetta *visione stereo*[1], invece, oggi possiamo godere della "terza dimensione" in diversi ambiti, legati ad attività

[1] *Stereo*, in Greco, significa "solido"

professionali piuttosto che di svago.

Inoltre, la percezione della profondità si presta ad utilizzi ancora più interessanti quando gli strumenti possono vantare caratteristiche tecniche accessibili, come dimensioni ridotte e facilità d'uso. Proprio quest'ultimo aspetto ha catturato l'attenzione di un gruppo di lavoro. In concomitanza con il corso di Multimedia e Tecnologie Creative del Dipartimento di Informatica di Bologna, è nata l'idea di sviluppare una soluzione che permettesse la stereo vision usando uno strumento estremamente diffuso del mercato tecnologico globale: uno *smartphone* e, più in generale, qualsiasi dispositivo mobile appartenente a questa categoria.

Con questa tesi si vuole, prima di tutto, raccontare il processo di analisi iniziale e successivo sviluppo di tale opera, partendo dai Capitoli 1 e 2, che rispettivamente trattano la storia della visione stereo e lo stato dell'arte per quanto concerne le sue applicazioni in ambito mobile. Successivamente, il lavoro svolto è trattato nel Capitolo 3, con la descrizione degli algoritmi dal punto di vista principalmente teorico. Gli aspetti e le problematiche scaturite durante la scrittura vera e propria dell'applicazione, difatti, vengono esposti nel successivo Capitolo 4. L'elaborato si conclude con una visione d'insieme del lavoro svolto e pone le basi per gli sviluppi successivi.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

La stereo vision

Tra gli innumerevoli ambiti di ricerca che compongono l'universo informatico ed ingegneristico, si sta velocemente affermando il particolare settore della *computer vision*, del quale fanno parte le tecniche dedicate all'acquisizione, analisi e manipolazione di immagini da parte di un elaboratore. In quest'area tutto è focalizzato sull'emulazione delle capacità percettive del sistema visivo umano, tramite l'unione di due grandi materie di studio: l'analisi delle immagini e l'intelligenza artificiale. Tali studi si concretizzano in numerose applicazioni, come il riconoscimento di gesti [1] o di volti ed espressioni facciali [2][3], l'automazione di veicoli stradali [4] o la cosiddetta *realtà aumentata*, per aggiungere elementi creati artificialmente in una scena reale, osservata per mezzo in un display [5].

Tra tutte le discipline che ne derivano, una in particolare è la cosiddetta *stereo vision* o visione stereoscopica. Con questo termine si indica l'insieme dei principi che, applicati a sistemi non necessariamente di un unico tipo, permettono di comprendere la profondità di una scena, sia statica che in movimento. Un'affascinante possibilità che già da diversi anni ha aperto le porte a molteplici utilizzi in campo scientifico ed in settori più *consumer*, come l'intrattenimento.

Del resto, un concetto che immediatamente viene associato alla parola "profondità" è senza dubbio "tridimensionalità", "terza dimensione", "3D".

Sono ormai termini di uso comune, sfruttati in svariate circostanze, a volte non senza una certa esagerazione. Si pensi alla progressiva diffusione di schermi appositamente progettati per dare all'utente una percezione sempre più naturale della profondità di un'immagine, di una scena in un film oppure in un videogame. Tali tecnologie sono certamente vettori di nuovi trend di mercato e stanno permettendo la nascita di nuove modalità di interazione con l'ambiente che ci circonda, ma non bisogna dimenticare che sono prima di tutto il risultato di una lunga fase di ricerca che fonda le sue origini nella fisiologia stessa dell'uomo.

1.1 La visione binoculare nel sistema visivo umano

La visione umana e di molti altri animali è caratterizzata dalla presenza di due occhi, opportunamente distanti tra loro, in modo da poter fornire al sistema cerebrale una doppia sorgente di informazione della medesima scena. E' la struttura base necessaria al processo, inequivocabile e non acquisibile con l'esperienza, chiamato *stereopsi* o visione binoculare [6].

Per ogni singolo istante durante il quale usiamo il nostro sistema visivo, tra l'occhio ed il cervello passano una quantità quasi incalcolabile di segnali che si traducono in informazioni sull'ambiente circostante. La luce viene catturata dalla superficie sensibile dell'occhio (Figura 1.1/1), costituita da due tipi di *fotoricettori*: i *coni* (sensibili al colore ed attivati in condizioni di alta luminosità) e i *bastoncelli* (attivati in maggiori condizioni di buio). Già in questa fase avviene il primo processo di trasformazione dell'energia entrante in attività elettriche da parte della *retina*. I due canali detti *nervi ottici* (1.1/2) che, partendo da ciascun occhio, dirigono il flusso verso il cervello, si congiungono a metà del percorso in una zona detta *chiasma* (1.1/3). In questo punto la coincidenza delle due informazioni produce la sensazione di profondità spaziale.

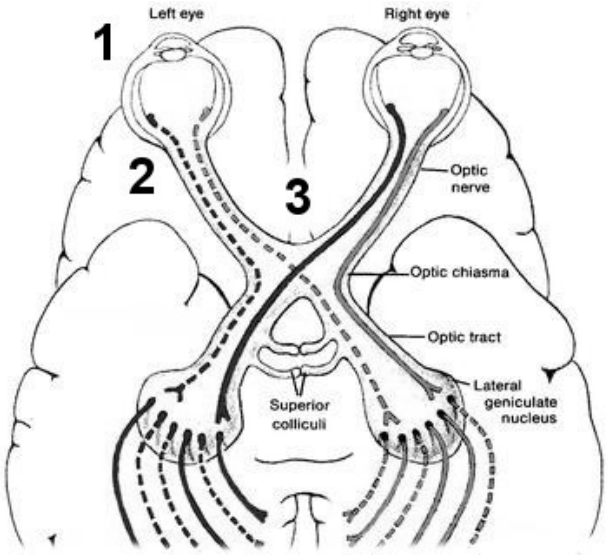

Figura 1.1: Il sistema visivo umano

I due punti di vista generati si distinguono per un certo grado di disparità. Banalmente, se tali viste si potessero sovrapporre si noterebbe che gli oggetti in primo piano nell'immagine di sinistra assumono una posizione che non combacia con quella corrispondente nell'immagine di destra. Cosa diversa, invece, avviene nei livelli più in profondità, dove le immagini tendono a combaciare man mano che si raggiunge lo sfondo. Ciò è fondamentale per far sì che il cervello possa comprendere le varie distanze alle quali gli oggetti sono dislocati rispetto all'osservatore.

Si potrebbe pensare, allora, che in assenza di una doppia visione perfettamente funzionante l'analisi della profondità di una scena verrebbe irrimediabilmente compromessa. In realtà, nei mammiferi cosiddetti "superiori", la semplice vista monoculare può essere abbastanza evoluta da emulare la stereopsi, senza però eguagliarne l'efficacia. Basta guardare una semplice fotografia per accorgersene: non stiamo effettivamente osservando una scena reale, ma il nostro cervello ha comunque le capacità per comprendere (sommariamente) le varie distanze. Questo avviene grazie ad indizi quali la prospettiva, la differenza di colori e dettagli percepiti da un oggetto e dall'altro o l'effetto di parallasse se la scena è in movimento. Sono tutti concetti

sfruttati da individui che, per natura o deficit, non impiegano la visione binoculare e, al tempo stesso, sono lo strumento principale per lo sviluppo delle famose illusioni ottiche.

E' da tali nozioni biologiche che nasce la teoria della stereo vision in campo matematico, ingegneristico ed informatico. Il modello più semplice per un'applicazione stereoscopica è mostrato in Figura 1.2:

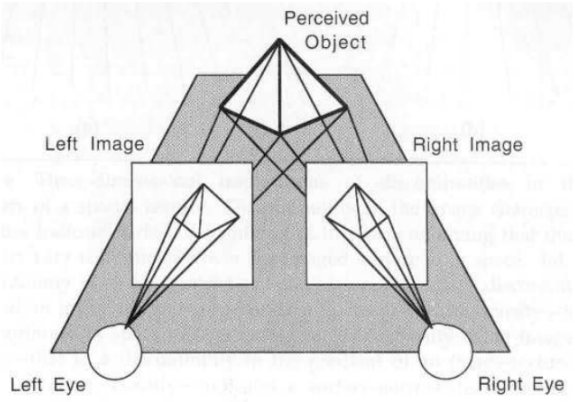

Figura 1.2: Un semplice modello di visione stereo.

Prevedendo necessariamente la presenza di almeno due immagini raffiguranti la medesima scena, catturate nello stesso momento da due punti di vista vicini tra loro e posti sullo stesso asse orizzontale (*baseline*), viene simulato un semplificato sistema visivo. Esistono una serie di relazioni geometriche tra i punti 3D nella scena reale e le loro proiezioni sui piani 2D, relazioni che è possibile definire algebricamente tramite le nozioni di *geometria epipolare*.

1.2 La geometria epipolare

Si tratta della geometria alla base di qualunque sistema di visione stereoscopica. La figura 1.3 mostra due camere che per semplicità sono perfettamente allineate e che osservano lo stesso punto P rispettivamente dai centri di proiezione O_l e O_r. Per ciascuna di esse, P viene proiettato sul rispettivo piano di proiezione nei punti p_l e p_r, aventi medesime coordinate verticali.

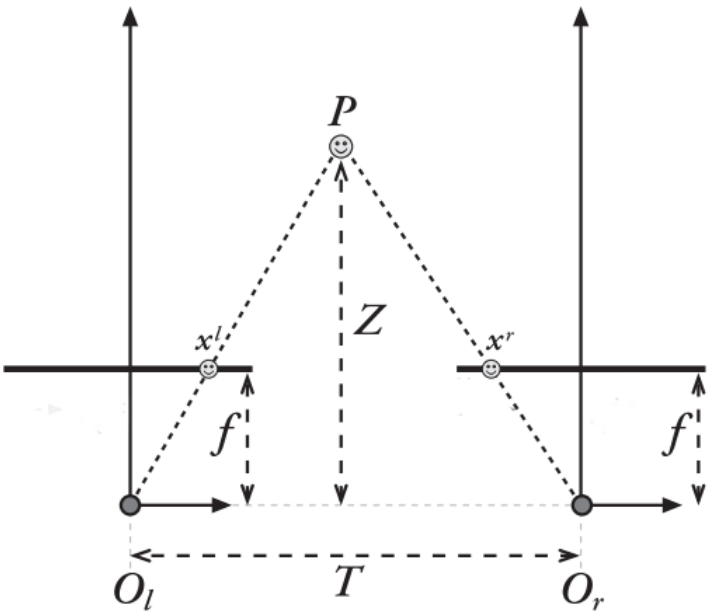

Figura 1.3: Schema della geometria epipolare.

Chiamando x^l e x^r le coordinate orizzontali di p_l e p_r sui rispettivi piani è possibile approssimare la profondità in cui si trova P rispetto ai punti d'osservazione, una misura inversamente proporzionale alla disparità tra le due viste, definita con la formula 1.1.

$$d = x^l - x^r. \tag{1.1}$$

All'aumentare della distanza tra P ed il punto di osservazione, il valore di d diminuisce. Non è difficile dimostrare questa tesi, alla base dell'effetto di parallasse, soprattutto se si utilizza un esempio della vita di tutti i giorni: a bordo di un treno in movimento, possiamo notarne gli effetti guardando un punto in lontananza fuori dal finestrino. Dopo un breve istante di tempo, la proiezione dell'oggetto osservato ha assunto una posizione molto simile a quella che ricordavano essere in precedenza, mentre gli oggetti più in primo piano sono ormai fuori dalla nostra visuale.

La stima della distanza tra il punto P e l'osservatore, partendo da d, è tuttavia un'approssimazione e un risultato più corretto si ottiene nel caso

in cui si conoscano altre due componenti: la distanza tra i due centri di proiezione T e la lunghezza focale delle due camere f, uguale per entrambe in tale scenario semplificato. L'esatta misura di profondità Z è calcolata dunque con la formula

$$\frac{T-(x^l-x^r)}{Z-f}=\frac{T}{Z} \implies Z=\frac{fT}{x^l-x^r} \tag{1.2}$$

Le nozioni matematiche fin'ora riportate vennero definite nei primi anni del 1900, ma la vera espressione di tali concetti risale a periodi storici ben più lontani.

1.3 La stereo vision nelle arti e nella cultura popolare

Già nel XV secolo, Leonardo Da Vinci comprese che gli oggetti a diverse distanze proiettano sugli occhi umani due immagini che si differenziano per una componente orizzontale (appunto, la disparità), concludendo che sarebbe stato impossibile per un pittore riuscire a rappresentare il concetto di solidità realizzando una sola tela.

Più tardi, in epoca vittoriana, furono fondamentali gli studi dell'inglese Sir Charles Wheatstone il quale, in una sua pubblicazione risalente al 1838, riporta per primo la definizione di stereopsi, integrandola negli anni successivi con la costruzione di un macchinario chiamato *stereoscopio*, un sistema di specchi per proiettare una coppia di immagini dello stesso oggetto verso l'occhio sinistro e destro, in maniera distinta, facendolo apparire solido alla vista.

Se, da un lato, gli studi di Wheatstone si affermarono in campo scientifico, dieci anni più tardi l'invenzione del *prisma stereoscopico* dello scozzese David Brewster, successivamente ottimizzata da Oliver Wendell Holmes, sancì invece la diffusione della stereo vision in ambito prettamente commerciale con la produzione di decine di migliaia di stereogrammi [7]. Si trattava di

coppie di immagini stampate su un singolo foglio di carta e visualizzabili in 3D attraverso un dispositivo binoculare portatile; la diffusione di questo formato fu tale che persino artisti come Salvador Dalí ne fecero uso [8].

Negli anni, l'attenzione verso queste tecniche crebbe nella cultura pop fino al massimo successo negli anni Novanta, grazie alla distribuzione di diverse pellicole cinematografiche 3D la cui fruizione prevedeva l'utilizzo di speciali occhiali detti *anaglifici*, dotati cioè di lenti di colori diversi (solitamente rosso e ciano) in modo da isolare le componenti "sinistra" e "destra" sovrapposte in un unico fotogramma e proiettarle rispettivamente su ciascun occhio.

Figura 1.4: Dallo stereoscopio di Wheatstone ai film in 3D anaglifico

Con l'avanzare del tempo, però, la popolarità di queste innovative tecniche di intrattenimento subì un veloce rallentamento, dovuto soprattutto alla poca efficacia delle tecnologie con le quali venivano proposte. Fino a quel momento la loro proiezione dipendeva da sistemi prettamente analogici, basati su tecniche di rifrazione della luce con uno scarso rapporto tra prezzo di costruzione e qualità offerta. L'esempio degli occhiali anaglifici nel cinema stereoscopico è evidente: dovendo dividere due informazioni provenienti dalla stessa immagine, la soluzione più diffusa fu quella di creare due fotogrammi sovrapposti ma differenti per la tonalità di colore, successivamente filtrati da ognuna delle due lenti montate sull'occhiale. Ne derivano perciò immagini dai colori troppo differenti da quelli originali, che compromettono il risultato finale senza inoltre escludere episodi di affaticamento della vista [9].

Solo negli ultimi anni si sta assistendo ad una ripresa della stereo vision grazie soprattutto all'evoluzione del cinema anaglifico, guidata da tecnologie

digitali basate sull'utilizzo di nuovi tipi di occhiali. *RealD 3D*, ad esempio, è presente in più di 1200 cinema nel mondo ed impiega lenti polarizzate [10].

Non solamente al cinema, oggi si possono osservare immagini tridimensionali anche dal divano di casa, grazie a televisori equipaggiati con appositi schermi e, nella maggior parte dei casi, occhiali dedicati.

Arrivati al XXI secolo, dunque, la ricerca di modi per dare profondità ad oggetti bidimensionali è entrata prepotentemente nella curiosità di scienziati ed inventori, anche assumendo varie forme a seconda dei mezzi utilizzati. Di seguito si riporta una lista delle tecniche più diffuse, cercando di mantenere un ordine partendo da quelle più efficaci:

- *Stampa lenticolare*: una tecnica per inserire due immagini stereo nello stesso foglio permettendo di rifletterne una sola a seconda del punto di vista dell'osservatore. E' una tecnica diffusa come strumento di marketing e alla base dei moderni televisori 3D che non richiede accessori aggiuntivi.

- *Vettogrammi*: speciali fogli costruiti sovrapponendo le due immagini tra un materiale polarizzante. La vista 3D si percepisce indossando appositi occhiati con lenti che inibiscono l'una e l'altra vista.

- *Immagini anaglifiche*, già citate in precedenza.

- *Blubblegram*: oggetti solidi, tipicamente di forma cubica e materiale trasparente, all'interno dei quali sono state prodotte tramite un laser delle piccole fratture atte a formare un'immagine interna.

- *Stereoscopia wiggle*: fornisce la sensazione di tridimensionalità attraverso la visualizzazione alternata di due o più immagini dello stesso soggetto, opportunamente ripreso da posizioni leggermente diverse.

- *Stereogrammi a punti casuali*: immagini apparentemente prive di senso ma che rivelano forme in profondità se visualizzate con appositi strumenti o fissando ad occhio nudo un punto centrale.

1.4 Nella ricerca scientifica

Parallelamente al mondo delle opere multimediali, la stereo vision crebbe con successo nel mondo della ricerca nei i più disparati settori. I primi algoritmi stereo, infatti, nacquero a supporto della *fotogrammetrica* [11, 12]: tecnica usata in cartografia, topografia ed architettura per recuperare informazioni geografiche di un oggetto (forma, posizione o altezza spaziale) partendo da una coppia di immagini della scena ripresa dall'alto, anche a notevoli distanze dal suolo terrestre.

Un settore per il quale la stereo vision assume un'importanza fondamentale è senza dubbio la modellazione del sistema visivo umano [13] e, implicitamente, la robotica [14, 15]. Gli apparati autonomi sfruttano tali nozioni durante gli spostamenti, riconoscendo ostacoli e prevedendo i percorsi migliori da intraprendere. Quello della visione binoculare si è anche rivelato essere un metodo caratterizzato da ottimo rapporto tra i costi di sviluppo e la qualità dei risultati offerti se paragonato ai più moderni sensori laser [16], spesso capaci di garantire performance superiori ma anche molto più costosi.

I requisiti hardware necessari per implementare sistemi di questo tipo variano, ovviamente, in relazione alle performance richieste ma in generale consistono in una coppia di "strumenti di cattura" (fotocamere di media o alta qualità ma anche semplici webcam [17]) ed un elaboratore per l'analisi degli input. La ricerca scientifica si dedica anche al miglioramento di tali aspetti: esistono implementazioni di visione binoculare real-time che sfruttano le GPU [18], sistemi FPGA[1] dedicati [19], oppure costruiti all'interno di una rete neurale cellulare [20]. Citando un esempio di successo, hardware di visione stereo è stato implementato a bordo dei due rover nella missione *Mars Exploration Rover* della NASA [21] (Figura 1.5/a).

Un ulteriore campo che, negli ultimi anni, ha iniziato ad impiegare la visione stereo è la medicina (1.5/b). La sensazione di profondità rappresenta un valore aggiunto durante un'operazione chirurgica, che richiede precisione

[1] *FPGA: Field Programmable Gate Array*, circuiti integrati con funzioni implementabili a livello software.

e deve focalizzarsi solo su una determinata parte del corpo o di un singolo organo.

Infine, troviamo applicazioni nei servizi di videosorveglianza [23] (movimenti sospetti, numerazione delle persone entranti ed uscenti da un edificio), nella ricostruzione di modelli 3D [22] (Figura 1.5/c) e nelle tecniche di produzione video tramite *z-keying* [24], per la sostituzione di un determinato livello di profondità in una scena reale con immagini digitali (1.5/d).

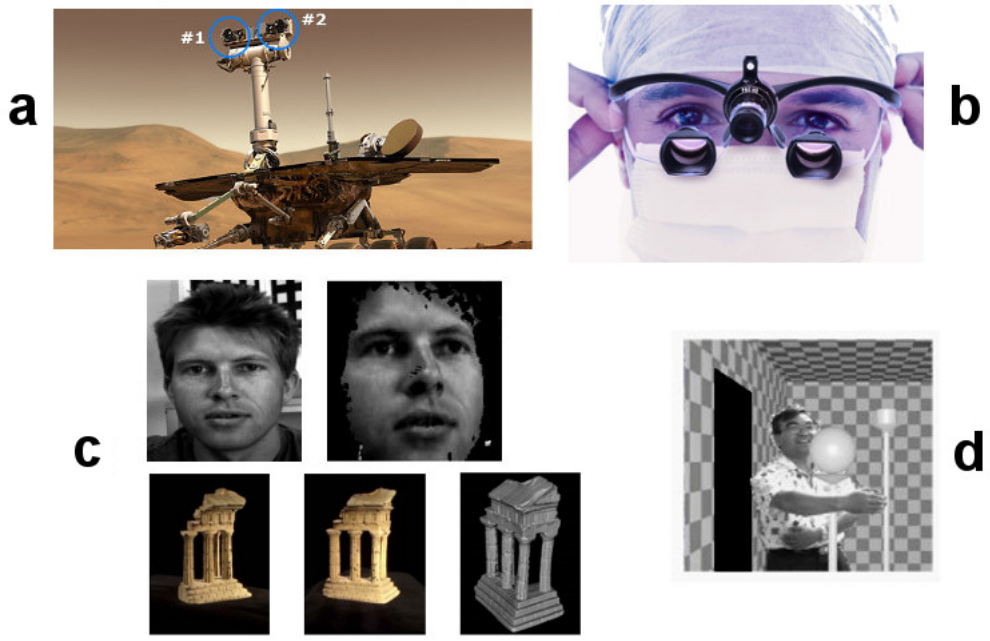

Figura 1.5: Alcune applicazioni in campo scientifico

Nella computer vision, dunque, l'analisi della profondità di una scena continua ad essere uno degli argomenti più attivi ed interessanti. Questo grazie anche alla sua adattabilità a diverse aree di studio.

Capitolo 2

La stereo vision in ambito mobile

Dopo aver analizzato ad ampio spettro lo stato dell'arte della visione stereo, il nostro gruppo di lavoro ha spostato l'attenzione verso un ambiente che, negli ultimi anni, sta ricevendo una considerevole attenzione sia da settori commerciali che da parte della ricerca scientifica. Trattasi del mercato dei dispositivi mobili *general-purpose*, nello specifico telefoni cellulari o tablet equipaggiati con hardware in grado di eseguire compiti anche relativamente complessi. Nelle sezioni successive ci si riferirà ad essi chiamandoli semplicemente "dispositivi mobili", specificando i casi in cui, invece, verranno citate architetture portatili costruite ad-hoc.

L'evoluzione delle nanotecnologie e dei sistemi per aumentarne l'efficienza energetica ci hanno permesso di poter acquistare, con un budget ristretto, un dispositivo che pone in secondo piano le ormai classiche chiamate audio ed introduce un nuovo livello di interazione sociale: trasferimento di dati in tempo reale rimanendo costantemente connessi alla rete Internet, servizi di geolocalizzazione per interagire con l'ambiente circostante, la visualizzazione e produzione di contenuti multimediali di alta qualità sono solo alcuni dei tanti possibili utilizzi. Proprio questo ultimo aspetto attrae sempre più spesso i consumatori: si sta definendo un trend per il quale, nelle decisioni di

acquisto di tali device, la possibilità di catturare e condividere fotografie qualitativamente molto simili a quelle professionali gioca un ruolo fondamentale. Si pensi solamente al "fenomeno *Instagram*", il social network di condivisione di foto e video, introdotto nel 2010 ed acquisito due anni dopo da *Facebook* per 1 miliardo di dollari, ha raggiunto i 150 milioni di utenti attivi nel mese di Dicembre 2013. Un totale di 16 miliardi di fotografie condivise fin dalla sua apertura [25].

Diventato il cellulare il primo strumento di intrattenimento oltre che di comunicazione a livello globale, non deve dunque stupire il fatto che la stereo vision abbia trovato applicazione anche in questo settore, non senza alcune importanti problematiche.

2.1 Stato dell'arte e limitazioni

Trovandoci ad analizzare lo stato delle cose per tecnologie non esclusivamente dedicate a tale tipologia di visione, è bene introdurre questo capitolo con un'osservazione. Una condizione per implementare tecniche stereoscopiche è, difatti, la presenza nel sistema di almeno due camere. Come risolvere la questione se una doppia sorgente di cattura non appare come caratteristica di default nel mercato mobile attuale? Inoltre, la suddetta condizione è veramente necessaria?

In secondo luogo, non può esistere visione stereo senza un efficace strumento di proiezione agli occhi di chi osserva. Anche in questo caso, è ancora difficile reperire smartphone o tablet equipaggiati con gli stessi schermi che troviamo sui televisori 3D.

Riprendendo, dunque, l'insieme di tecnologie attualmente in commercio e descritte in 1.3, le dividiamo in due classi principali:

- Tecnologie di acquisizione. L'insieme di strumenti necessari a catturare la scena in modalità "stereo", quindi necessarie a fornire un input al processo di conversione 2D - 3D, si dividono a loro volta in due sottocategorie:

  - Acquisizione diretta: quando nel sistema sono presenti almeno due camere che, nello stesso momento, acquisiscono la scena da due punti di vista distinti.
  - Acquisizione indiretta: situazione comune nei dispositivi mobili, la presenza di un'unica camera apre le porte a nuovi metodi di acquisizione stereo.

- Tecnologie di visualizzazione. Dedicate esclusivamente alla fruizione del risultato, anche in questo caso si dividono a loro volta in dirette ed indirette:

  - Visualizzazione diretta e basata sull'*autostereoscopia*: solitamente le più efficaci, all'immagine stessa è applicato un sistema per proiettare sull'occhio sinistro una visuale diversa rispetto a quella per l'occhio destro, apparendo quindi in rilievo senza dover usare accessori aggiuntivi.
  - Visualizzazione indiretta: richiedono di visualizzare l'immagine in 3D attraverso l'utilizzo di uno strumento intermedio, come degli occhiali specifici.

In questa sezione analizzeremo una serie di soluzioni nate dalla combinazione delle classi appena riportate ed attualmente presenti nel mercato mobile, con i relativi vantaggi e difetti.

2.1.1 Tecnologie di acquisizione

Sono ancora pochi i dispositivi mobili che, distribuiti globalmente, integrano un sistema di acquisizione diretto [26]. Il modello *HTC EVO 3D*, rilasciato nel 2011, è il primo di questi in ordine di uscita sul mercato e presenta due fotocamere installate a pochi millimetri di distanza l'una dall'altra [27].

La bassa diffusione di device di questo tipo non deve essere intesa come un fallimento della visione stereo. I motivi che spingono i produttori a non

adattare i propri prodotti sono perlopiù legati a fattori fisici (dallo spazio occupato dai vari componenti hardware al costo di produzione) ma anche commerciali. I trend attuali non considerano ancora gli smartphone o i tablet come i principali strumenti per l'acquisizione di immagini stereo, dando priorità a tecnologie di tutt'altro tipo e scopo.

A fronte di questa mancanza, la creatività degli sviluppatori indipendenti ha permesso la nascita di una serie di soluzioni sia meccaniche che software. Sono da segnalare supporti fisici che, basati su un sistema di lenti applicate alla fotocamera del device, duplicano i punti di cattura per un'acquisizione diretta: *Poppy* per iPhone [28] (Figura 2.1) è sicuramente un curioso esempio di tali meccanismi.

Figura 2.1: Poppy per iPhone, un accessorio per abilitare l'acquisizione diretta di immagini stereo

Soluzioni che, invece, non richiedono l'utilizzo di componenti aggiuntivi devono necessariamente essere implementate a livello applicativo. Riprendendo la domanda fatta introducendo questa sezione, ci si è chiesti se la presenza di due camere fisiche fosse effettivamente necessaria in questi casi. In realtà, un sistema stereo può essere approssimato utilizzando un'unica sorgente di acquisizione, catturando una prima foto e, successivamente, tra-

slando il dispositivo di una certa distanza sull'asse orizzontale per scattare la seconda.

Si introducono, ovviamente, alcuni aspetti legati al movimento imposto ed ai relativi tempi di esecuzione. Si ricorda, infatti, che la direzione di spostamento deve essere il più parallela possibile alla baseline. Inoltre, una variazione della scena tra l'una e l'altra fotografia rischia di vanificare il risultato, non essendoci relazione tra gli oggetti in ambedue le viste (pensiamo, ad esempio, ad un luogo molto affollato o ad una strada trafficata).

Tali svantaggi sono, purtroppo, quelli che affliggono la maggior parte dei software di acquisizione indiretta, presenti nei marketplace dei rispettivi sistemi mobili. Alcuni di essi impiegano step intermedi in cui è l'utente a correggere manualmente l'allineamento (Figura 2.2).

Figura 2.2: Applicazioni con fasi di acquisizione indirette

2.1.2 Tecnologie di visualizzazione

Anche per quanto riguarda la fase di visualizzazione dei risultati troviamo differenti soluzioni. Lo smartphone HTC EVO 3D, già oggetto di attenzione per il metodo di acquisizione diretto, è inoltre equipaggiato con uno schermo dedicato che sfrutta una tecnologia denominata *Parallax Barrier*. Tale schermo consiste in uno strato a cristalli liquidi al quale viene aggiunto un

materiale "forato" in punti precisi, permettendo ad ogni occhio di visualizzare solo determinati pixel (Figura 2.3).

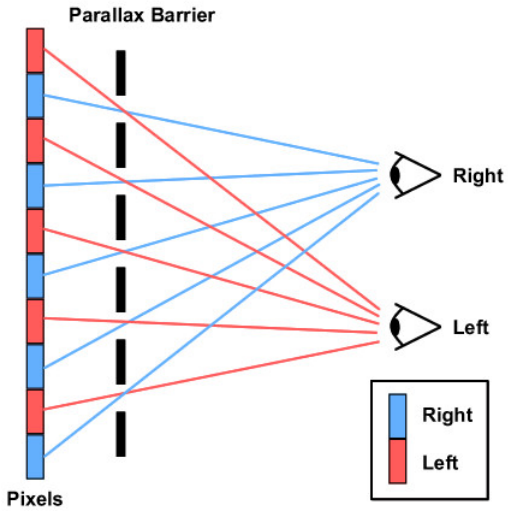

Figura 2.3: Schema di funzionamento di un display con tecnologia Parallax Barrier

Questi dispositivi, quindi, si distinguono per essere strumenti di stereo vision completamente diretti. Soddisfano efficacemente la richiesta di due punti di ripresa e forniscono un'effettiva sensazione di effetto 3D, ma non hanno ancora raggiunto una rilevante quota di mercato a causa di una serie di svantaggi. Primo fra tutti, per visualizzare correttamente un'immagine è necessario assumere una precisa posizione perpendicolare allo schermo, condizione piuttosto complicata soprattutto quando si utilizzano tali oggetti in movimento. Inoltre, non sono rari i casi in cui si avverte disagio a causa delle difficoltà dell'occhio umano nell'adattarsi a questa nuova tipologia di schermo [29], una problematica già analizzata in precedenza nel cinema anaglifico.

Contemporaneamente alla soluzione hardware proposta dalle case produttrici, diversi sviluppatori hanno rilasciato alternative "amatoriali" che tentano di emulare l'effetto tridimensionale sui normali schermi, a livello software. Nella maggior parte dei casi, ci si affida alla ormai sorpassata tecnica degli anaglifi, vincolando l'utente all'utilizzo degli specifici occhialini ed

agli svantaggi che ne derivano (Figura 2.4/a). Un'applicazione in particolare, invece, sfrutta algoritmi di *head-tracking* applicati al funzionamento della fotocamera frontale di cellulari e tablet. Si chiama *i3D* [31] e visualizza oggetti solidi variandone la prospettiva a seconda di dove l'utente stia rivolgendo lo sguardo in quell'istante (Figura 2.4/b). Il problema, oltre alla necessità di una camera che osservi il volto dell'utente, è che tale applicazione permette di visualizzare forme esclusivamente create e distribuite dagli sviluppatori, senza fornire un sistema per catturare scene personalizzate.

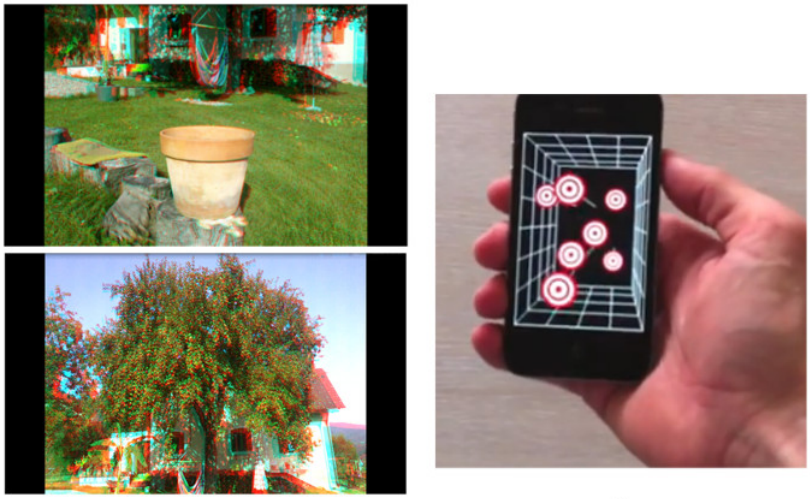

Figura 2.4: Applicazioni mobili che implementano soluzioni di visualizzazione indiretta

2.2 La stereo vision mobile in campo sperimentale

Recentemente, l'attenzione dei ricercatori si è focalizza sull'analisi della profondità di una scena utilizzando una singola fotocamera. E' infatti uno dei problemi più rappresentativi di questo campo poichè, come accennato in precedenza, un buon risultato è ottenibile acquisendo due punti di vista man-

tenendosi stabilmente sulla stessa linea orizzontale. L'utilizzo di strumenti di supporto, come un treppiede, non è sempre possibile in questi casi.

Alcune implementazioni sfruttano durante la calibrazione i dati forniti da sensori di movimento [30], componenti ormai disponibili direttamente nell'hardware degli stessi smartphone. Tali informazioni possono essere sfruttate da sistemi autonomi, per esempio a bordo di veicoli, per calcolare la distanza di un oggetto dal punto di osservazione a meno di un certo grado di errore. Il calcolo, in questo caso, è basato sulle nozioni di geometria epipolare viste in 1.2 e proprio per questo presenta un limite: tali misure non possono essere calcolate con efficacia senza conoscere a priori alcuni dati tecnici, come la lunghezza focale dell'obiettivo e la distanza tra i due punti di cattura.

2.3 La ricerca di una soluzione alternativa

Abbiamo fino a qui citato una serie di soluzioni per la visione stereoscopica (diretta ed indiretta) appositamente pensate per quella categoria di dispositivi che oggi, nel mondo, occupa il primo posto per diffusione. Soluzioni che, oltre all'aspetto puramente "ludico", introducono una serie vantaggi rispetto alle implementazioni viste nel Capitolo 1. Primi fra tutti:

- L'aspetto economico: l'utilizzo di device come smartphone e tablet per la visione stereo può tradursi in un risparmio di risorse economiche non indifferente se paragonato alle già presenti architetture dedicate.

- L'aspetto pratico: l'analisi della profondità diventa un'operazione molto più semplice se eseguita con questi strumenti, che in poco spazio includono sia una sorgente di cattura che un'unità di calcolo.

- La facilità di diffusione: una soluzione stereoscopica, quando non richiede accessori speciali o dati tecnici specifici di ogni device, ha la possibilità di essere installata ed utilizzata da un bacino d'utenza molto più ampio, soprattutto se distribuita come una comune applicazione tramite i vari negozi online.

Abbiamo notato, tuttavia, come le attuali proposte soffrano di alcuni svantaggi a seconda della loro tipologia: da un lato, le tecnologie di visualizzazione dirette non richiedono accessori aggiuntivi ma necessitano di schermi appositi, dall'altro, quelle puramente software sono molto lontane dal poter offrire risultati di qualità.

Ne deriva così l'esigenza di una soluzione alternativa, che possa unire i vantaggi di entrambe le categorie, mantenendo pressochè nulli i costi di installazione e fornendo al tempo stesso una buona rappresentazione della profondità dell'ambiente appena catturato. Organizzando un vero e proprio team di sviluppo è nata *SuperStereo*, un'originale applicazione stereoscopica studiata per il mercato mobile general-purpose.

Capitolo 3

SuperStereo: Snap, Tilt, 3D, Repeat

Presentiamo, in questo capitolo, i risultati del lavoro di progettazione ed implementazione di SuperStereo. Tale applicazione, pensata per i sistemi mobili più diffusi, si propone come soluzione ad una serie di limiti introdotti dall'utilizzo della visione binoculare in questo particolare ambito.

Ne verrà prima di tutto presentato il *concept*: i campi di utilizzo, il *target* inteso come i potenziali utenti ai quali è rivolta e le caratteristiche che la contraddistinguono dallo stato dell'arte.

In seguito, l'attenzione verterà sugli algoritmi di stereo vision sfruttati, esposti dapprima a livello teorico e poi implementativo, senza focalizzare l'attenzione verso una specifica piattaforma mobile.

Solo successivamente, verranno esposte le problematiche nate dallo sviluppo su ognuno dei diversi sistema operativi, con le relative soluzioni adottate.

3.1 Il concept dell'applicazione

Definiti, nella Sezione 2.1, i vantaggi che un'applicazione stereoscopica mobile apporta in diverse aree di utilizzo ed i difetti delle attuali proposte, il nostro gruppo di lavoro ha deciso di intraprendere un percorso di ricerca

per ridurre il gap prestazionale tra le soluzioni di stereoscopia diretta ed indiretta.

Con l'introduzione di una serie di funzionalità appositamente studiate per l'occasione, SuperStereo si inserisce nel mezzo di questi due modelli implementativi, con l'obbiettivo di acquisire alcuni punti di forza. Deve essere facile da utilizzare, deve implementare algoritmi efficienti e, soprattutto, deve fornire un efficace sistema di visualizzazione del risultato, in questo caso dell'effetto 3D.

Il raggiungimento di tali propositi renderebbe SuperStereo un'applicazione mobile in grado di affermarsi non solamente nel mondo puramente "artistico" della fotografia, dove la stereo vision occupa un posto di riguardo, ma anche in quegli ambienti dove l'analisi delle profondità è necessaria o estremamente utile e richiede strumenti compatti e possibilmente poco costosi: la robotica, l'architettura e la medicina, per citarne alcuni.

3.1.1 Il sistema di acquisizione

Con l'intento di fornire a chi utilizza l'applicazione un metodo di acquisizione stereo semplificato, è stata studiata una procedura semi-automatica, che non richiede accessori esterni e che può essere eseguita con azioni elementari.

Ispirandoci al funzionamento delle tecniche indirette, l'utente ha il compito di catturare una prima fotografia del soggetto, che verrà intesa come l'immagine *sinistra*, subito dopo esegue un movimento traslando il dispositivo verso destra, di pochi centimetri, sufficienti a scattare la foto *destra*. La distanza percorsa è infatti simile a quella che intercorre tra i due occhi del nostro sistema visivo. In questo caso, però, viene attivata un'operazione automatica per rilevare il momento giusto in cui attivare una seconda volta l'otturatore (Figura 3.1).

L'utente, durante questa fase, ha il solo compito di allineare la vista a schermo con una copia semitrasparente dell' immagine precedente, traslata di alcuni pixel per imporre il movimento del dispositivo. Il sistema verifica

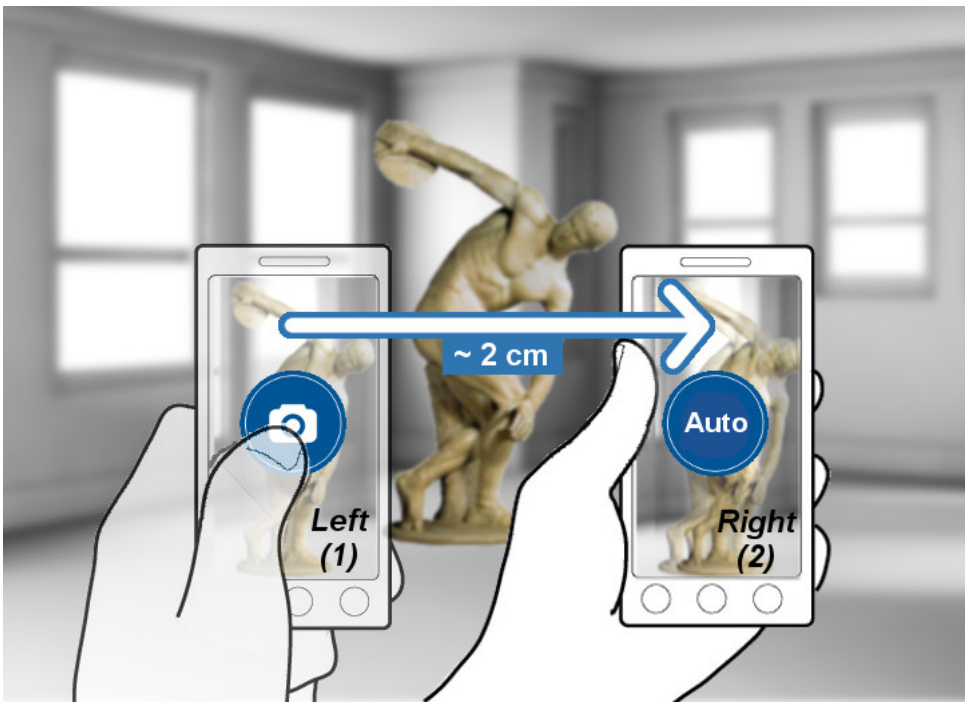

Figura 3.1: Schema del funzionamento del sistema di acquisizione semi-automatico

in tempo reale l'allineamento, comparando una piccola porzione centrale dell'immagine sinistra con la zona attualmente visibile dalla fotocamera, nella medesima posizione.

Esistono vari metodi di comparazione tra immagini (o algoritmi di *template matching*) e durante il nostro lavoro ne è stato implementato più di uno, per fasi differenti. In questo specifico caso si è scelto di utilizzare l'indice *Zero-mean Normalized Cross Correlation*: chiamate I_L e I_R le due porzioni di immagine prese in esame, tale misura è definita dalla seguente formula.

$$ZNCC(I_L, I_R) = \frac{\sum_{(i,j)\in N}(I_L(i,j) - \overline{I_L}) * (I_R(i,j) - \overline{I_R})}{\sqrt{\sum_{(i,j)\in N}(I_L(i,j) - \overline{I_L})^2 * \sum_{(i,j)\in N}(I_R(i,j) - \overline{I_R})^2}} \quad (3.1)$$

Dove N è l'insieme dei pixel che compongono ogni immagine, $\overline{I_L}$ e $\overline{I_R}$ sono rispettivamente i valori medi di grigio di I_L e I_R (Formula 3.2), mentre

al denominatore troviamo il prodotto tra le deviazioni standard dei valori in ognuno dei due input.

$$\overline{I} = \frac{\sum_{(i,j)\in N}(I(i,j))}{|N|} \tag{3.2}$$

Il risultato della funzione *ZNCC* assume un range di valori nell'intervallo $[-1; 1]$: a valori uguali a 1 corrispondono due immagini identiche, viceversa se tendenti a -1. Appena superata una soglia predefinita (fissata nei test a 0.85), viene acquisita automaticamente la seconda fotografia.

Questa misura è stata preferita alle altre della stessa famiglia poichè invariante rispetto a trasformazioni radiometriche delle immagini. Infatti, i sensori installati sulle camere degli attuali smartphone o tablet non vantano ancora una qualità eccellente, di conseguenza sono comuni i casi in cui due frame catturati a breve distanza tendano ad essere affetti da cambiamenti repentini di colore, a volte impercettibili dall'occhio umano.

Il metodo proposto è dunque un'evoluzione delle tecniche di cattura stereo viste nella precedente Sezione 2.1.1, poichè è il sistema stesso a comprendere il momento esatto in cui completare l'operazione. Si noti che tale implementazione non risolve, invece, un secondo aspetto: la presenza di eventuali rotazioni del dispositivo durante il movimento che, seppur minime, possono potenzialmente vanificare la qualità del risultato. Per questo motivo verrà analizzato un algoritmo di allineamento della coppia di immagini, il primo dei tre algoritmi di stereo vision che rappresentano il cuore del funzionamento di SuperStereo.

3.1.2 Il sistema di visualizzazione

Il vero punto di forza che contraddistingue SuperStereo dalle soluzioni attuali è il suo sistema di visualizzazione 3D, che si potrebbe inserire in un'ipotetica tipologia denominata "indiretta a livelli". Ricordiamo le due grandi classi di tecniche stereoscopiche: quelle che richiedono schermi abilitati e quelle che richiedono accessori supplementari. La tecnica qui presen-

tata non richiede l'installazione nè di schermi speciali (eliminando, tra l'altro, i problemi di affaticamento della vista già citati) nè di appositi occhiali anaglifici.

Il concetto alla base sfrutta l'effetto di parallasse: supponendo di avere a disposizione un'immagine e le relative informazioni sulla profondità, se ne può eseguire una divisione in una pila di livelli, ognuno dei quali indipendente dagli altri. Gli oggetti in primo piano, dunque, faranno parte dei livelli superiori, viceversa quelli appartenenti allo sfondo (Figura 3.2). Costruito questo stack di nuove immagini, l'effetto è garantito dal loro movimento: ad ogni modifica del punto di osservazione, ogni livello verrà traslato di un fattore inversamente proporzionale all'indice di profondità ad esso assegnato.

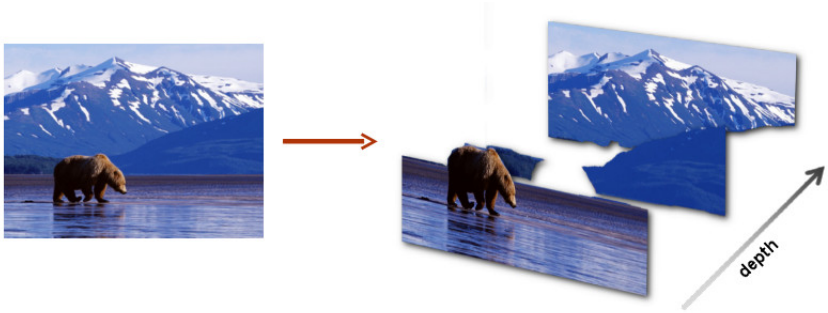

Figura 3.2: Divisione dell'immagine di input in livelli di profondità

Per rendere l'effetto più naturale possibile, a modificare l'offset di traslazione di ogni elemento è l'utente stesso, attraverso l'inclinazione del dispositivo e, quindi, in base allo stato dell'accelerometro.

Come è avvenuto per la fase di acquisizione degli input, anche in questo caso tra i tre algoritmi ne è stato sviluppato uno ad-hoc che, vedremo, introdurrà una problematica non trascurabile.

3.1.3 Il logo ed il payoff

Con questo originale modo di intendere il 3D su un dispositivo mobile sono nati il logo ed il *payoff* del progetto. Il primo (Figura 3.3/a), semplice nella sua forma, è costituito dall'unione di più simboli:

- Uno smartphone, strumento essenziale per l'utilizzo dell'applicazione.
- Una freccia sullo sfondo, a rappresentare il movimento verso destra che l'utente deve compiere in fase di acquisizione; il suo colore, dal bianco al nero, ricorda lo standard per rappresentare una mappa di profondità, nella quale oggetti in primo piano vengono marcati con un colore chiaro, più scuro per quelli in lontananza.
- Le due lettere "S", acronimo del nome dell'applicazione, il cui stile evoca l'effetto tridimensionale.

Il payoff (Figura 3.3/b) è composto da una sequenza di quattro parole inglesi: *Snap* (riferito allo scatto della fotocamera), *Tilt* (relativo all'inclinazione da applicare allo smartphone durante la fase di visualizzazione), *3D* (per l'effetto finale), *Repeat* (come auspicio per un ulteriore utilizzo).

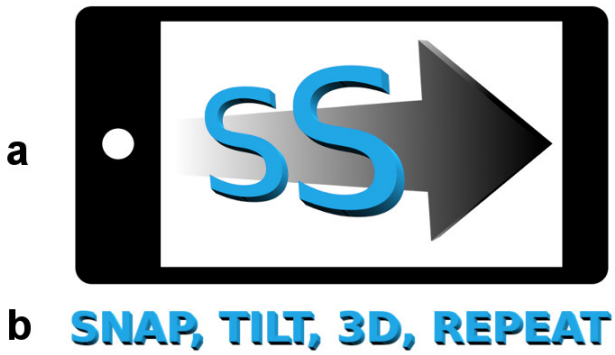

Figura 3.3: Logo e payoff di SuperStereo

3.1.4 Workflow del programma ed introduzione ai tre algoritmi principali

Definite le caratteristiche principali di SuperStereo, si introducono ora gli step intermedi che, partendo da una coppia di immagini acquisite tramite la fase di cattura, producono l'effetto finale.

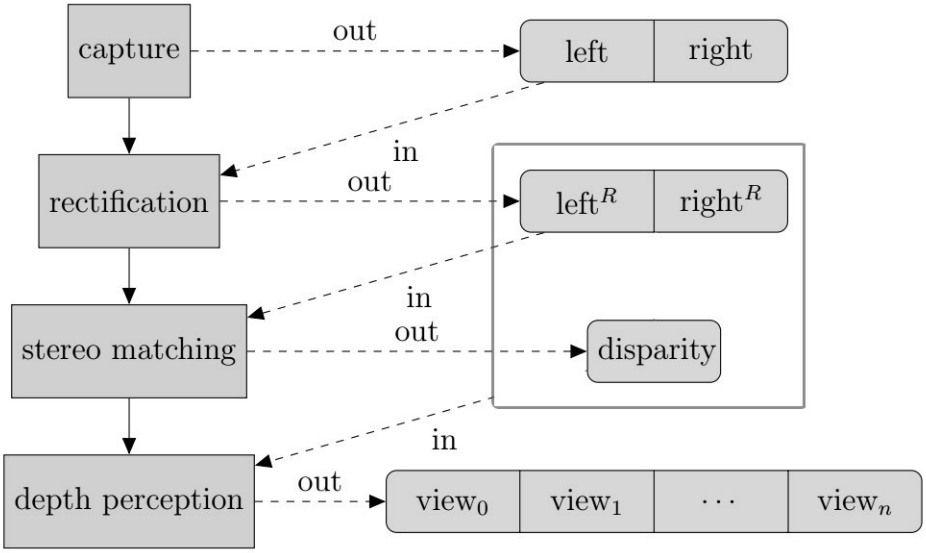

Figura 3.4: Flusso di operazioni per abilitare l'effetto tridimensionale in SuperStereo

Il Diagramma 3.4 mostra una sintesi logica di tutto il procedimento: le forme rettangolari a sinistra si riferiscono alle fasi che, in sequenza, dirigono l'utente dal primo step di cattura fino alla fase di visualizzazione del risultato. Ogni fase ha un proprio pacchetto di dati in output (forme arrotondate a destra), il quale serve da input per gli step successivi. Nelle prossime sezioni verranno descritte le operazioni interne, che non hanno bisogno di un'interazione dell'utente, caratterizzate dall'implementazione di specifici algoritmi di stereo vision:

1. *Rettificazione*: da due foto catturate con il metodo descritto in Sezione 3.1.1, ne vengono generate due versioni "allineate".

2. *Stereo matching*: il processo per comprendere le varie profondità che compongono la scena.

3. *Depth perception*: il processo che, leggendo le immagini allineate e i nuovi dati sulle profondità, concretizza il metodo in Sezione 3.1.2.

3.2 Algoritmo 1: Rettificazione delle immagini

Un algoritmo di *image rectification* [32] è una sequenza di operazioni che, partendo da due o più immagini rappresentanti una scena comune, permettono di generare una serie di altrettante immagini, come se fossero state catturate da camere virtuali, nate da una rotazione delle precedenti e poste sullo stesso sistema di coordinate (dicasi *rettificate* o *coplanari*). E' una tecnica diffusa, ad esempio, nei processi di *image stitching* [33], per combinare assieme più viste dello stesso soggetto senza che si notino discontinuità (si pensi alle immagini panoramiche).

Per i nostri scopi, tali algoritmi sono stati presi in considerazione in modo da ovviare ai problemi di disallineamento delle due viste, acquisite nella fase di cattura descritta in 3.1.1. Il movimento apportato verso una direzione per catturare la seconda immagine, infatti, sarebbe impossibile da eseguire alla perfezione da parte di una comune persona, soprattutto se a mano libera come richiesto dalla nostra applicazione.

Va altresì detto che la procedura che, in seguito, verrà utilizzata per analizzare la profondità della scena, trarrà un notevole guadagno in termini di efficienza se tale vincolo è rispettato: anticipandone i concetti, sarà necessario implementare una ricerca dei punti corrispondenti alle due viste. Tale ricerca risulterà dunque molto più rapida se eseguita esclusivamente su una singola dimensione, in questo caso quella orizzontale.

Per questi motivi, la fase che immediatamente segue il processo di acquisizione è dedicata all'allineamento delle due viste. In particolare, verrà analizzato il caso della cosiddetta *rettificazione non calibrata*, ovvero tale per cui non è necessario conoscere a priori i parametri della camera come la lunghezza focale o il formato di immagine gestito dal sensore (parametri chiamati *intrinseci*).

3.2.1 La matrice fondamentale

Per introdurre le nozioni utili ad implementare un algoritmo di rettificazione, si recuperano i concetti di geometria epipolare visti in Sezione 1.2.

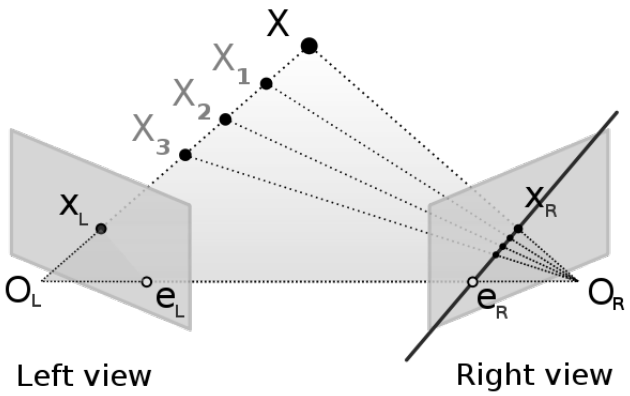

Figura 3.5: Ulteriore schema della geometria epipolare, evidenziando una linea epipolare ($e_R - x_R$).

Date due immagini raffiguranti una scena comune, sia X un punto osservato da entrambe (Figura 3.5). Nella prima foto, tale punto è mappato nello spazio a due dimensioni $\mathcal{P}^2$ in x_L e l'insieme dei punti che in $\mathcal{P}^3$ da X portano a x_L è visto come un unico punto dalla camera di sinistra. Al contrario, nella seconda immagine, questo insieme è effettivamente una linea retta, chiamata *linea epipolare*. Da qui un importante vincolo geometrico: qualsiasi punto X_i nella seconda immagine che corrisponde al punto X visto dalla prima risiede lungo tale retta.

Essendo i centri di proiezione delle camere O_L e O_R distinti, ognuno di essi è proiettato in un punto nella camera opposta detto *epipolo* (e_L e e_R).

Di conseguenza, è giusto pensare che esista un sistema di mappatura tra punti nella prima immagine e le rispettive linee epipolari nella seconda. In particolare, tale mapping è proiettivo e viene definito tramite la costruzione di una matrice 3x3 detta *matrice fondamentale* F, indicandolo con $u \to Fu$.

Se $u_i \leftrightarrow u'_i$ rappresenta un insieme di punti corrispondenti (*matching points*), allora il fatto che un punto u'_i risieda sulla linea epipolare Fu_i implica:

$$u_i'^{\mathrm{T}} F u_i = 0 \tag{3.3}$$

Dove $u_i'^{\mathrm{T}}$ è il vettore trasposto di u_i in $\mathcal{P}^3$. La matrice F è calcolabile con un minimo di 8 corrispondenze tra punti, risolvendo un sistema di equazioni lineari della forma (3.3). Nel caso di immagini perfettamente rettificate, la matrice F risultante ha la seguente forma:

$$F = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.4}$$

e qualsiasi coppia di punti corrispondenti risiede sullo stesso asse orizzontale.

3.2.2 Mappatura rigida degli epipoli e minimizzazione delle distorsioni

Definite le proprietà di F, il passo successivo consiste nel trovare una matrice di trasformazione prospettica H che, applicata ad una delle due immagini, mappi l'epipolo ad un punto in direzione "infinito". Infatti, se le linee epipolari dovranno diventare parallele all'asse x , allora l'epipolo deve essere mappato al punto $(1, 0, 0)^{\mathrm{T}}$.

Tale procedura lascia un certo grado di libertà nella scelta di H, che non è unica e permette di generare diverse trasformazioni al fine di correggere la posizione degli epipoli, rischiando comunque di produrre nuove immagini troppo deformate rispetto a quelle iniziali. Una condizione che porta a risultati visibilmente accettabili è che tale trasformazione sia il più possibilmente *rigida* nell'intorno di un dato punto trasformato u_0: ai punti vicini dovranno essere applicate, esclusivamente, traslazioni e rotazioni.

Si consideri, ad esempio, il punto all'origine dell'immagine u_0, l'epipolo $p = (f, 0, 1)$ e la seguente trasformazione:

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{pmatrix} \tag{3.5}$$

Tale trasformazione porta, come richiesto, l'epipolo $p = (f, 0, 1)^{\mathrm{T}}$ al punto all'infinito $p = (f, 0, 0)^{\mathrm{T}}$. Per un punto $(u, v, 1)^{\mathrm{T}}$, l'applicazione di G produce $(\hat{u}, \hat{v}, 1)^{\mathrm{T}} = (u, v, 1 - u/f)^{\mathrm{T}}$.

Se $|u/f| < 1$, allora possiamo scrivere:

$$\begin{aligned} (\hat{u}, \hat{v}, 1)^{\mathrm{T}} &= (u, v, 1 - u/f) \\ &= (u(1 + u/f + \dots), v(1 + u/f + \dots), 1)^{\mathrm{T}}. \end{aligned} \tag{3.6}$$

e la matrice Jacobiana:

$$\frac{\partial(\hat{u}, \hat{v})}{\partial(u, v)} = \begin{pmatrix} 1 + 2u/f & 0 \\ v/f & 1 + u/f \end{pmatrix} \tag{3.7}$$

Avendo considerato come esempio il punto d'origine, si ha che $u = v = 0$ e la matrice (3.7) altro non è che l'identità.

Per un generico punto u_i ed un epipolo p', invece, la matrice di trasformazione H è tale che $H = GRT$, dove T è una traslazione che porta u_i all'origine e R una rotazione nell'origine che porta p' ad un punto $(f, 0, 1)^{\mathrm{T}}$ sull'asse x. Si tratta, effettivamente, di una trasformazione rigida.

Come scegliere H? In particolare, come scegliere una *coppia* di trasformazioni H^L e H^R da applicare ad ognuna delle rispettive immagini I_L e I_R, in modo da renderle tra loro rettificate, minimizzando al massimo le distorsioni? Se λ e λ' sono una coppia di rette epipolari corrispondenti tra le due immagini di input, allora

$$H^{L*}\lambda = H^{R*}\lambda' \tag{3.8}$$

Dove H^{L*} e H^{R*} sono le matrici cofattori delle rispettive H^L e H^R. Ogni coppia di trasformazioni che soddisfano tale condizione è detta *match*. L'ob-

biettivo, dunque, è quello di trovare la *migliore* coppia $(H^L; H^R)$ basandosi su un certo criterio, ad esempio trovando quella per cui è minima la distanza d intesa come somma di quadrati:

$$\sum_i d(H^L u_i^L, H^R u_i^R)^2 \tag{3.9}$$

3.2.3 L'algoritmo implementato

La presenza, in letteratura, di diversi algoritmi di image rectification è dovuta a differenti metodi per approssimare la coppia $(H^L; H^R)$ migliore. Per SuperStereo, durante la scelta di quale metodo implementare si è dovuto considerare, ovviamente, il rapporto tra la qualità dei risultati e l'efficienza, a fronte dei limiti prestazionali introdotti dall'utilizzo di hardware mobile.

Inoltre, non bisogna dimenticare un altro aspetto molto importante di questa prima fase: la necessità di disporre di un insieme di punti corrispondenti tra le due immagini di partenza. Tali informazioni, infatti, sono l'input fondamentale per la fase di rettificazione non calibrata e devono essere recuperate attraverso uno step preliminare.

Ricerca dei punti corrispondenti

Anche in questo caso, esiste un'ampia famiglia di algoritmi capaci di eseguire un'operazione di *keypoint matching* tra due immagini. Il funzionamento, ad ogni modo, è comune a tutti a meno di particolari ottimizzazioni e si divide in tre sottoprocessi:

Fase 1: Keypoint detection
Letteralmente, avviene una ricerca di "punti chiave" all'interno di ognuna delle immagini. Si possono intendere, ad esempio, come i pixel che compongono gli angoli delle forme, caratterizzati da una significativa discontinuità di colore rispetto ai loro vicini. Non è un caso, difatti, che tali metodi derivino dai concetti di *edge detection* dell'analisi delle immagini (Figura 3.6/a).

Tra tutti i possibili candidati, l'algoritmo di ricerca dei keypoint scelto per SuperStereo è *FAST* [35], già sfruttato nella computer vision su dispositivi mobili per la sua facile portabilità.

Fase 2: Keypoint description

Un descrittore si può intendere come una "firma" che identifca ogni keypoint rilevato. Gli algoritmi a disposizione si differenziano a seconda di come questa informazione è codificata. Ad esempio, un descrittore può essere inteso come una stringa di bit di lunghezza prefissata, costruita combinando le diverse intensità di colore che circondano ogni angolo. Metodi più raffinati, invece, memorizzano in un istogramma i gradienti che compongono l'immagine, vale a dire vettori la cui direzione segue la variazione di intensità di grigio in specifiche zone (Figura 3.6/b).

E' il caso dei descrittori *BRISK (Binary Robust Invariant Scalable Keypoints)* [36] e *ORB (Oriented FAST and Rotated BRIEF)* [37], quest'ultimo, in particolare, implementato nel nostro lavoro per la sua efficienza in termini di memoria occupata nel salvare le varie codifiche.

Fase 3: Keypoint matching

Dati un keypoint k sulla prima immagine e k' sulla seconda, la verifica dell'effettivo match tra i due si basa sul calcolo della *distanza di Hamming* tra le stringhe di bit che li descrivono, ovvero il numero di posizioni nelle quali i valori sono diversi. Chiamate le stringhe S e S', entrambe di lunghezza n:

$$Hamming(S, S') = \sum_{i=0}^{n} S_i \oplus S'_i \tag{3.10}$$

Dove $\oplus$ è l'operatore logico XOR. Se tale distanza è minore di una certa soglia, allora la corrispondenza è considerata valida. Calcolando ogni distanza con un'iterazione "forza bruta", ovvero confrontando ogni keypoint della prima immagine con tutti quelli nella seconda, si costruisce un insieme di corrispondenze tra punti, che rappresenterà l'input dell'algoritmo di rettificazione implementato (Figura 3.6/c).

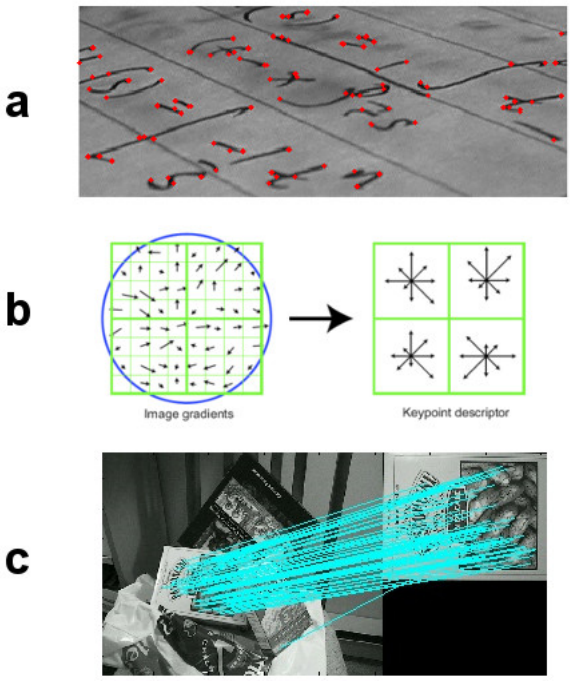

Figura 3.6: Esempi di algoritmi usati per la ricerca delle corrispondenze: keypoint detection (a), description (b) e matching (c).

Quasi-Euclidean Epipolar Rectification

Concluso lo sviluppo del keypoint matcher, la scelta del metodo di rettificazione si è rivolta verso il lavoro di Fusiello A. e Irsara L., *"Quasi-Euclidean Epipolar Rectification"* [34], che approssima una rettificazione al caso *calibrato*, stimando la lunghezza focale dell'obbiettivo, partendo da un set di punti corrispondenti.

Definita ogni omografia H, una per ogni immagine di input I, l'applicazione di (3.11) genera la nuova vista rettificata I^r.

$$I^r(x,y) = I\left(\frac{H_{11}^{-1}x + H_{12}^{-1}y + H_{13}^{-1}}{H_{31}^{-1}x + H_{32}^{-1}y + H_{33}^{-1}}, \frac{H_{21}^{-1}x + H_{22}^{-1}y + H_{23}^{-1}}{H_{31}^{-1}x + H_{32}^{-1}y + H_{33}^{-1}}\right) \tag{3.11}$$

Dove H^{-1} è la matrice inversa di H.

Figura 3.7: Le tre fasi della rettificazione: in alto, due immagini *sinistra* e *destra* non allineate, lo dimostrano i keypoints ricavati nella seconda fase, al centro. Dopo aver applicato le trasformazioni, nell'immagine in basso, i punti in comune risiedono sul medesimo asse verticale.

3.3 Algoritmo 2: Stereo matching

Terminato il processo di allineamento delle due viste, generando artificialmente due nuove immagini, il passo successivo consiste nel ricavare le informazioni circa le posizioni che gli oggetti occupano sull'asse z.

Con quale livello di precisione si possono stimare tali dati? Se ne può definire un'unità di misura? Sarà possibile concludere che un dato punto nella scena dista, ad esempio, *42 metri* dall'osservatore? E' stata trattata la geometria epipolare quale concetto algebrico per stimare la profondità di un oggetto, sotto precisi vincoli. La presenza di due camere è, ovviamente, una condizione necessaria nei sistemi stereoscopici e, qualora si volesse arrivare ad un'esatta misurazione, andrebbero considerati i loro parametri intrinseci, applicando in questo modo l'Equazione 1.2.

Dal punto di vista geometrico, tali nozioni possono effettivamente essere sfruttate da un sistema di misurazione precisa. Nel nostro caso, abbiamo invece adottato una formula semplificata, sufficiente ai nostri scopi, che richiede di calcolare esclusivamente la differenza tra le coordinate orizzontali x^l e x^r di un punto osservato (Equazione 1.1). Applicandola ad ogni singolo pixel di una delle due immagini e salvandone il risultato in una matrice di medesime dimensioni, viene a delinearsi la cosiddetta *mappa di disparità densa*. I valori in tale struttura vengono racchiusi nell'intervallo $[0; 255]$ per permetterne una rappresentazione in scala di grigi, dove pixel di intensità tendenti a 0 (colore nero) descrivono zone più lontane, viceversa per i pixel che assumono toni chiari, i quali identificano zone più vicine (Figura 3.8).

In questa fase, le difficoltà maggiori si identificano nel processo chiamato *stereo matching* o problema della corrispondenza stereo, tramite il quale, per tutti i punti della vista sinistra (o un loro sottoinsieme), si ricerca il corrispondente in quella a destra.

Figura 3.8: Mappa di disparità di una scena creata artificialmente

3.3.1 Tassonomia

La corrispondenza stereo è, da tempo, una delle tematiche più investigate nell'intero ecosistema della computer vision. La sua evoluzione è continua e, tutt'ora, vengono periodicamente rilasciati nuovi algoritmi adatti ad utilizzi generici o specifici per determinate aree di studio.

A fronte dell'elevata disponibilità di articoli e documentazioni a riguardo, nel 2002 i ricercatori Daniel Scharstein (*Dipartimento di Matematica e Informatica del Middlebury College*) e Richard Szeliski (*Microsoft Research*) ne produssero una catalogazione dal titolo *"A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms"* [38], che si affermò come la principale fonte di studio per la progettazione di tali tecniche.

Principalmente focalizzati agli algoritmi di corrispondenza *densa* (dove i valori vengono calcolati per tutti gli elementi dell'immagine, contrariamente ai metodi *sparsi*), gli autori identificano le criticità che si devono affrontare durante queste operazioni, così come una serie di punti di riferimento per la valutazione qualitativa delle soluzioni proposte[1].

Innanzitutto, le problematiche risiedono in una serie di fattori "naturali" delle immagini e di come esse vengono percepite da parte di un comune strumento di acquisizione fotografica. Date due immagini catturate a pochi

[1]Tali metriche sono disponibili al sito `http://vision.middlebury.edu/stereo/`, che include un elenco degli algoritmi in ordine di efficacia ed un benchmark per eseguire test qualitativi

centimetri l'una dall'altra, la ricerca dei pixel corrispondenti tra la prima e la seconda vista obbliga a tenere in considerazione le seguenti tematiche:

- *Distorsioni e presenza di rumore*: la ricerca può fallire se le immagini sono affette da rumore, la cui presenza è inevitabile anche nelle fotocamere di alta qualità e provoca variazioni di intensità tra due pixel effettivamente corrispondenti.

- *Occlusioni*: variando la prospettiva in una scena varia anche la visibilità degli oggetti al suo interno. Alcuni pixel comporranno parti di oggetti visibili solo in una delle due immagini, la mappa di disparià conterrà informazioni non valide in quel punto.

- *Presenza di superfici non-lambertiane*: dicasi lambertiana una superficie che riflette l'energia incidente in modo uguale in tutte le direzioni. Di conseguenza, variando il punto di vista la sua apparenza non cambia. Quando invece tale proprietà non è soddisfatta (si pensi ad una sfera di vetro) la ricerca delle corrispondenze non produce risultati corretti.

- *Texture uniformi o ripetute*: cercare l'esatto punto corrispondente all'interno di una superficie dai colori uniformi (come un muro bianco) è un'operazione complessa, dovendo scegliere tra una grande quantità di punti candidati con intensità molto simili. Lo stesso vale in presenza di pattern che si ripetono (come una scacchiera).

- *Il vincolo epipolare*, per il quale i punti corrispondenti risiedono sullo stesso asse orizzontale, se soddisfatto permette di diminuire di molto la complessità dell'algoritmo. In presenza di immagini non perfettamente allineate la ricerca deve invece avvenire in entrambe le dimensioni.

E' chiara, quindi, la varietà di elementi visivi che vanificherebbero i risultati. A fronte di questo, numerosi algoritmi sono stati sviluppati per limitare uno o più di tali problemi, a seconda degli utilizzi. In particolare, si sono definite nel tempo due grandi classi di ricerca: *locale* e *globale* all'immagine.

Metodi di ricerca locale

Nei metodi di ricerca di tipo locale, per ogni pixel analizzato se ne esegue la comparazione con i candidati non solamente attraverso il confronto dell'intensità del singolo, ma anche con quella dei pixel che ricadono nel suo intorno, quindi all'interno di una finestra, solitamente quadrata e relativamente piccola per motivi prestazionali (Figura 3.9).

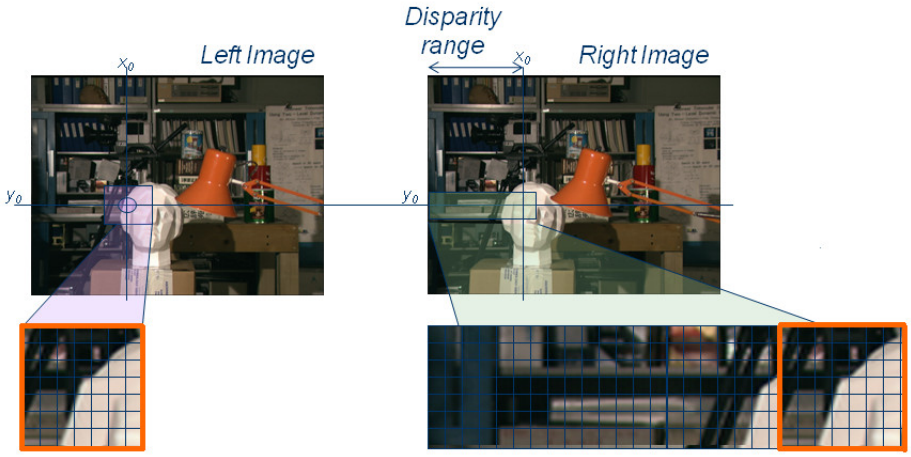

Figura 3.9: Scorrimento orizzontale di una finestra 7x7 durante una ricerca locale

Tale metodologia assume che la disparità all'interno di ogni finestra si mantenga costante e ciò, purtroppo, non è sempre vero. A fronte di questo, algoritmi detti *a supporto adattivo* modificano la forma della finestra in relazione alle proprietà della regione attualmente visitata, ottenendo risultati migliori a discapito della velocità d'esecuzione. La ricerca locale, inoltre, fallisce in presenza di superfici uniformi ed ambigue, a meno di eventuali correzioni in uno step successivo.

In generale, gli algoritmi di tipo locale spesso non producono mappe di disparità particolarmente fedeli, ma sono anche quelli che più si prestano ad implementazioni efficienti, non necessitando di strutture dati onerose in termini di memoria occupata ed essendo facilmente parallelizzabili. Per questo

motivo ne sono state sviluppate versioni su hardware appositamente studiato, che ne permette l'esecuzione in tempo reale [19].

Metodi di ricerca globale

La seconda classe di metodi non implementa la ricerca attraverso l'utilizzo diretto dei supporti locali, piuttosto considera alcune proprietà comuni ai pixel dell'intera immagine. Si introducono concetti di intelligenza artificiale [39], programmazione dinamica [40], algoritmi cooperativi [42] o l'implementazione di strutture a grafo [41] per identificare il miglior candidato tra tutti i possibili corrispondenti.

La maggiore complessità rispetto ai metodi locali rende queste operazioni piuttosto difficili da ottimizzare in ambienti hardware non particolarmente performanti, anche se la qualità del risultato finale è mediamente migliore. E' infatti semplice riconoscere la classe di algoritmi utilizzata guardando solamente la mappa di disparità risultante. La Figura 3.10 propone una comparazione unicamente visiva tra un approccio locale, basato sulla differenza di intensità tra due finestre [38], ed uno globale, basato sul partizionamento di un grafo [43].

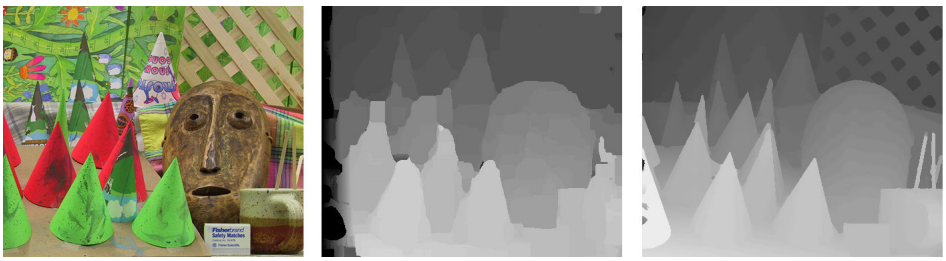

Figura 3.10: Comparazione visiva tra un metodo di ricerca delle corrispondenze locale, al centro, e globale, a destra. Si notino le differenze soprattutto nel rilevamento dei bordi (ad esempio per i coni) e dei piccoli cambi di profondità (griglia in alto a destra).

Struttura base di un algoritmo di stereo matching

Durante la progettazione di questi algoritmi, è buona norma considerare una serie di sotto-processi che non dovrebbero mancare per l'elaborazione di una buona mappa di disparità. Szeliski e Scharstein ne identificano quattro, anche se la scelta della loro implementazione è ovviamente riservata allo sviluppatore:

1. Computazione dei costi (*matching cost computation*): per ogni pixel preso in esame, se ne calcola inizialmente un "costo", ovvero un valore che misura quanto esso è diverso rispetto ai candidati nella seconda immagine. La metrica utilizzabile spazia dalla più semplice somma di differenze assolute (*SAD*), che considera le intensità di grigio, a combinazioni più complesse come *ZNCC*, già affrontata in Sezione 3.1.1. Si possono inoltre analizzare proprietà strutturali della regione, che vanno oltre il semplice colore del pixel. Vedremo, successivamente, come SuperStereo implementi una metrica di questo ultimo tipo.

2. Aggregazione dei costi (*cost aggregation*): definiti i costi per ogni punto, lo step successivo è quello di raggrupparli in modo da avere una visione più ampia della zona attualmente in analisi. Solitamente ad ogni costo si aggiungono quelli nell'intorno del relativo pixel (è il caso del metodi locali basati su piccole finestre bidimensionali), semplicemente sommandoli oppure ricorrendo a proprietà statistiche, ad esempio assegnando pesi diversi ai costi centrali.
   Volendo migliorare la resa, il supporto può anche essere tridimensionale [44], con le dovute complicazioni.

3. Computazione delle disparità (*disparity computation*): costruito un set di possibili candidati per ogni pixel, con i relativi punteggi, i metodi locali implementano questa fase in maniera piuttosto rapida tramite una selezione *"winner takes all"*. La corrispondenza che detiene il risultato migliore (quindi il costo minimo) viene scelta e ne viene assegnata

la disparità con la formula 1.1, a meno di eventuali raffinamenti come l'utilizzo di tecniche *sub-pixel* [45].

4. Ottimizzazione finale (*disparity refinement*): l'intero processo può dirsi concluso con la fase precedente, anche se, nella maggior parte dei casi, la mappa presenta artefatti più o meno evidenti. Tali imperfezioni si possono eliminare con l'applicazione di vincoli "logici" sulla disparità (come un doppio check tra la mappa prodotta partendo dall'immagine sinistra e, poi, da quella destra) o di filtri *passa-basso* (blurring gaussiano, mediani, eccetera).
   Alcuni algoritmi, tra i quali quello implementato in SuperStereo, combinano ai filtri anche un processo di segmentazione per migliorare la qualità dei bordi degli oggetti.

3.3.2 L'algoritmo sviluppato

La fase di analisi delle profondità in SuperStereo si affida al modello, appena visto, di ricerca locale. Durante la sua progettazione, infatti, è stato fondamentale mantenerne una struttura relativamente semplice che potesse comunque risultare in mappe fedeli alla realtà. Dovendo considerarne l'implementazione su dispositivi non dedicati, con poca memoria RAM e capacità di calcolo diverse dagli standard desktop, è stato sviluppato un algoritmo che ben si presta a tali ambienti hardware e che è riuscito ad inserirsi nella classifica Middlebury con un interessante risultato. Questo algoritmo, difatti, riesce ad ottenere risultati molto vicini a quelli offerti da metodi che richiedono più risorse.

Un ulteriore punto di forza è rappresentato dalla tipologia dei test effettuati. Durante l'analisi dello stato dell'arte, abbiamo notato come molti algoritmi definiscano le loro potenzialità esclusivamente sulla base del dataset ufficiale, contenente immagini qualitativamente diverse da quelle che, in media, vengono acquisite con uno smartphone. Nel secondo caso, infatti, le foto possono imprevedibilmente presentare rumore, rifrazioni della luce sola-

re e variazioni di colore tra l'una e l'altra vista. Nel nostro algoritmo, invece, tali fattori sono stati tenuti in considerazione.

Un ultimo vantaggio: il metodo proposto non richiede nessuna fase iniziale di *tuning*, per la quale i vari parametri vengono adattati al tipo di scena. Tale procedimento, se richiesto, renderebbe l'utilizzo dell'applicazione troppo complicato, vanificandone un potenziale successo.

3.3.3 Computazione dei costi ed aggregazione

Nella prima parte del processo, vengono calcolati i differenti valori di similarità, prendendo come riferimento l'immagine sinistra. Per ogni pixel $p(x_L; y_L)$ visitato, i possibili candidati $p'(x_R; y_R)$ sull'immagine di destra si trovano sullo stesso asse orizzontale, dunque $y_L = y_R$, mentre x_R varia in un range di disparità fissato arbitrariamente:

$$D = \{d \quad | \quad -d^L \leq d \leq +d^R\} \tag{3.12}$$

Da qui una prima particolarità dell'algoritmo: i metodi classici controllano i candidati variando un offset sempre positivo $[0; d^R]$, assumendo che le due viste siano traslate sempre perpendicolarmente alla baseline (è il caso dei sistemi a doppia camera, non convergenti, che non necessitano di rettificazione). Nel caso di SuperStereo, un utilizzatore potrebbe inconsapevolmente applicare una rotazione della camera attorno l'asse y, rendendo le due viste convergenti in un punto verso l'infinito. Oltre questo punto, la ricerca dei pixel più lontani deve necessariamente avvenire anche per $x_R < x_L$.

Riguardo alla metrica di confronto utilizzata, una prima scelta ha portato all'utilizzo della *somma di differenze quadrate (SSD)* sul livello di intensità di grigio, più discriminante della SAD in presenza di aree molto simili tra loro. Sia $I_l(x, y)$ l'intensità del pixel dell'immagine sinistra in posizione $(x; y)$ e d lo spostamento attuale sull'asse orizzontale, il costo di ogni candidato è calcolato nella maniera seguente:

$$\forall d \in D, \quad C_S(x, y, d) = (I_l(x, y) - I_r(x + d, y))^2 \tag{3.13}$$

La tecnica di aggregazione è anch'essa molto semplice, ogni costo è sommato a quelli nel suo intorno di dimensioni 7x7. La scelta finale del punto corrispondente, invece, è fatta attraverso la logica "winner takes all" che, tra tutti i candidati lungo l'offset D, sceglie quello con il costo minore.

Di seguito lo pseudocodice di questa procedura iterativa (Algoritmo 1), di complessità $O(D(N+NM))$, con D l'insieme degli offset di disparità (definito precedentemente), N l'area di ogni immagine, M l'area della finestra. Le strutture di supporto sono $leftCosts$ e $rightCosts$ per salvare i costi assegnati ad ogni pixel (spostando l'offset rispettivamente a sinistra e a destra della coordinata x), $minCosts$ per salvare i costi migliori e $disparity$, che rappresentarà la mappa. In essa, valori tendenti a $-d^L$ segneranno pixel in lontananza, viceversa per quelli vicini a $+d^R$. Punti uguali a 0, invece, segneranno zone dove la disparità tra le due viste è nulla ed, implicitamente, è avvenuta la convergenza tra i punti d'osservazione.

La Figura 3.11 mostra un primo output [2], evidenziando alcune criticità:

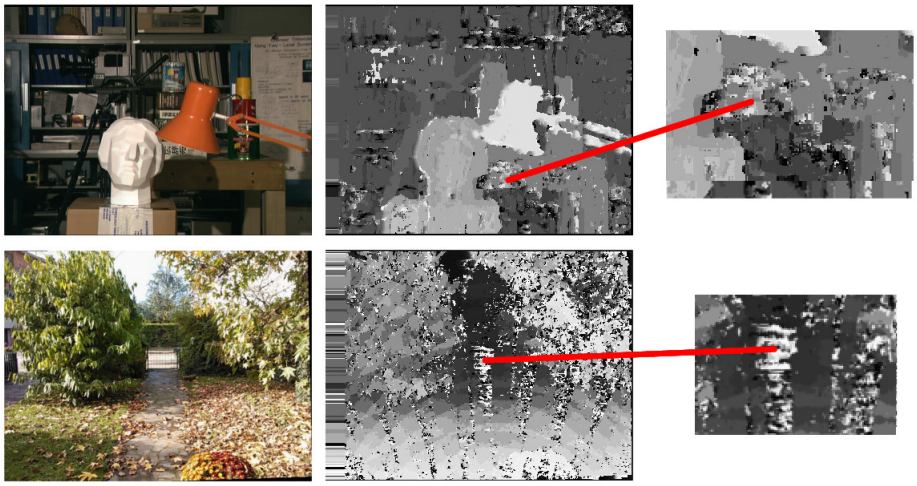

Figura 3.11: Mappe di disparità generate con metrica SSD ed aggregazione a finestra 7x7

[2]Le immagini che seguono mostrano comparazioni provenienti dal dataset ufficiale Middlebury e da fotografie catturate con un comune smartphone di fascia media.

Algorithm 1 Algoritmo di ricerca locale delle corrispondenze

for all $d \in D$ **do**
  for all $x, y \in N$ **do**
    if $x - d > 0 \wedge d \leq d^L$ **then**
      $leftCost \leftarrow C_S(x, y, d)$
    else
      $leftCost \leftarrow maxval$
    end if
    if $x + d \leq W \wedge d \leq d^R$ **then**
      $rightCost \leftarrow C_S(x, y, d)$
    else
      $rightCost \leftarrow maxval$
    end if
    $leftCosts(x, y) \leftarrow leftCost$
    $rightCosts(x, y) \leftarrow rightCost$
  end for
  for all $x, y \in N$ **do**
    $accL \leftarrow 0$
    $accR \leftarrow 0$
    for $x_m, y_m \in M$ **do**
      $accL \leftarrow accL + leftCosts(x_m, y_m)$
      $accR \leftarrow accR + rightCosts(x_m, y_m)$
    end for
    $minError \leftarrow min(accL, accR)$
    if $minError < minCosts(x, y)$ **then**
      $minCosts(x, y) \leftarrow minError$
      if $minCosts(x, y) = accL$ **then**
        $disparity(x, y) \leftarrow -d$
      else
        $disparity(x, y) \leftarrow +d$
      end if
    end if
  end for
end for

Si può notare come questa metrica fallisca in presenza di texture uniformi (dettaglio della foto *statua*) e ripetute (dettaglio nella foto *giardino*), assegnando un punteggio errato in quelle zone. E' inoltre evidente la presenza di picchi di rumore "sale-pepe" sparsi lungo tutta l'area. Il motivo di tali alterazioni è dovuto al tipo di comparazione tra i pixel che, se esclusivamente basata sulla loro intensità, non è sufficiente ad ottenere buoni risultati.

Per questo, si può considerare una seconda misura, basata su una trasformazione locale e non-parametrica introdotta da Zabih e Woodfill, chiamata *Census* [46]. Tale filtro non solo dipende dall'intensità di un pixel, ma anche dalla sua posizione rispetto ai suoi vicini. La definizione è piuttosto semplice: dato un pixel $p(x; y)$ di intensità $I(p)$ ed un suo intorno W, la trasformata si calcola costruendo una stringa di bit S tale per cui:

$$\forall p' \in W,\ 0 \leq i < |W| \quad S_i(x,y) = \begin{cases} 1 & \text{if } I(p) \geq I(p') \\ 0 & \text{otherwhise} \end{cases} \tag{3.14}$$

Successivamente, il calcolo del costo per due dati pixel avviene confrontando le relative stringhe, misurandone la distanza di Hamming:

$$\forall d \in D, \quad C_C(x,y,d) = \sum_{x,y \in W} Hamming(S^l(x,y), S^r(x+d,y)) \tag{3.15}$$

La trasformata Census risulta robusta in presenza di rumore e variazioni di luminosità tra una e l'altra immagine. Al tempo stesso, riduce sensibilmente le informazioni circa il valore di ogni pixel. E' comunque possibile combinare la precedente metrica SSD con questo nuovo operatore, in particolare è stata implementata la variante proposta da Fröba e Ernst [47], che mette in comparazione il valore medio della finestra W anzichè l'intensità del punto centrale.

La combinazione tra SSD e Census è pesata dai fattori λ_S e λ_C (nei test fissati rispettivamente a 0.7 e 0.3) e non introduce cali di performance significativi:

$$\forall d \in D, \quad C(x,y,d) = \lambda_S * C_S(x,y,d) + \lambda_C * C_C(x,y,d) \tag{3.16}$$

Il risulato è una mappa più precisa, come mostra la Figura 3.12.

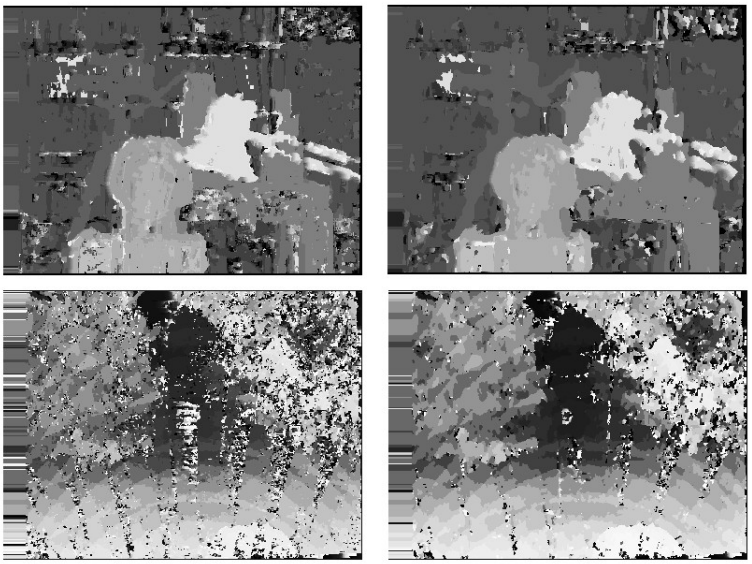

Figura 3.12: Confronto tra mappa di disparità generata con metrica SSD (a sinistra) e SSD + Census (a destra)

3.3.4 Ottimizzazione della mappa

Con la fase appena conclusa, l'algoritmo mantiene una struttura piuttosto semplice che, vedremo, non richiede un elevato quantitativo di risorse. Al tempo stesso, l'utilizzo della trasformata Census (modificata) garantisce un certo grado di robustezza quando le immagini in input presentano i classici difetti delle fotocamere amatoriali.

Nella fase successiva si ha la possibilità di raffinare le informazioni circa le profondità, ad esempio rimuovendo i rimanenti picchi di rumore e ridefinendone meglio i bordi. La mappa presenta infatti i noti svantaggi degli algoritmi stereo locali: una maschera troppo piccola (ad esempio 3x3) permette performance migliori a discapito della precisione, mentre una troppo

grande (nell'ordine dei 10x10 pixel) rende la procedura più lenta. La dimensione scelta per la nostra finestra, 7x7, è un buon compromesso ma fa in modo che i bordi degli oggetti vengano letteralmente "gonfiati", come si nota, ad esempio, nel volto dell'immagine *statua*.

Prima di introdurre una soluzione a questo problema, si è ragionato su come si potessero diminuire efficacemente le regioni affette da rumore, le quali presentano, in pochi pixel, alte variazioni di intensità.

Applicazione di filtri di smoothing

Tra i molteplici filtri di *image processing*, a loro volta derivati dalla teoria dei segnali, quelli appartenenti alla categoria di *smoothing* eseguono operazioni di sfuocatura sulle immagini, attenuando il rumore di conseguenza.

Tali operatori, detti passa-basso proprio per il fatto che eliminano dall'output le alte frequenze, hanno uno svantaggio: rendono meno definiti i bordi delle zone su cui operano, effetto che, nelle nostre mappe, si vuole invece correggere. Per questo, un secondo filtro che prende il nome di *mediano* [48] è stato preso in considerazione: dato un pixel, il nuovo valore applicatogli è, appunto, il mediano tra tutti quelli del suo intorno, solitamente definito da una finestra quadrata di ridotte dimensioni.

Nell'esempio in Figura 3.13, il pixel centrale ha intensità 0, è quindi totalmente nero e, visto localmente assieme ai suoi vicini, rappresenta un punto di rumore. Applicando il filtro mediano il suo valore sale a 60, migliorando visibilmente.

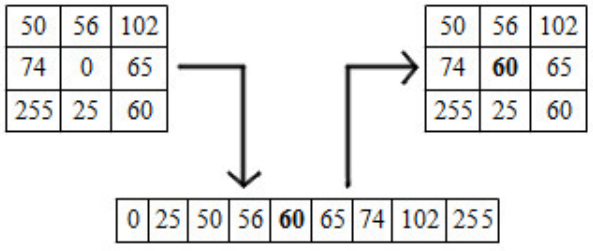

Figura 3.13: Esempio di filtro mediano con una maschera 3x3

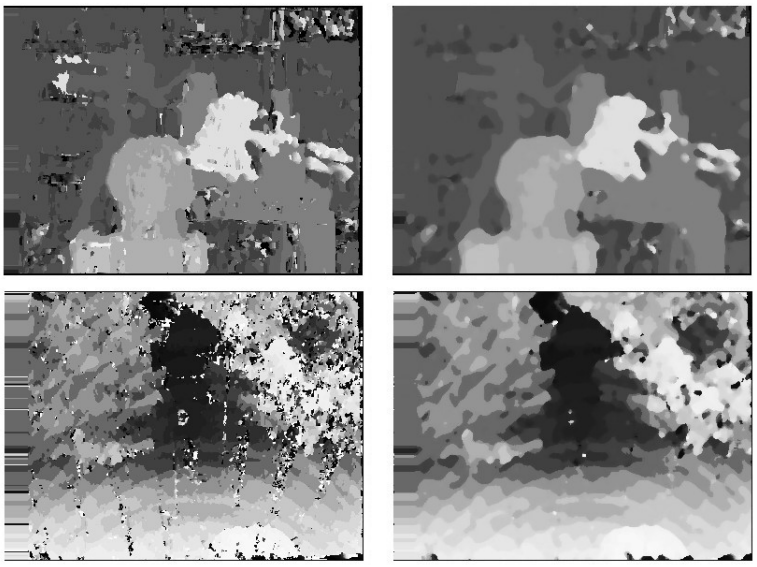

Figura 3.14: Confronto tra mappe di disparità prima e dopo l'applicazione di un filtro mediano con maschera 7x7

Recupero dei contorni tramite segmentazione

Eliminata un'evidente quantità di rumore dalla mappa, l'ultima fase è dedicata al recupero dei dettagli relativi ai bordi degli oggetti. Un'operazione di questo tipo può essere intesa in vari modi, ad esempio, se il problema maggiore risiede nei bordi delle zone più chiare, si potrebbe ricorrere ad un ulteriore filtro di imaging della classe degli operatori morfologici, detto di *erosione* [48]. Il suddetto, infatti, con un'operazione anch'essa basata su maschere, aiuta a ridurre l'area di tutte le zone esclusivamente chiare.

E' evidente che tale operazione risulta troppo generica e non si focalizzata su determinate aree, pur essendo per i suoi scopi efficace e veloce. Si è deciso, invece, di ricorrere alle più complesse tecniche di *segmentazione*, consapevoli del fatto che buona parte di tali algoritmi non brillano per efficienza computazionale.

Un'operazione di segmentazione è un processo che partiziona un'immagine digitale in *superpixel*, ovvero sezioni di pixel che si accomunano per una certa proprietà, solitamente per colori simili. Da un punto di vista più

generico, si può intendere come un'operazione che assegna ad ogni punto un'etichetta, in modo tale da identificare insiemi di elementi che condividono caratteristiche comuni.

Altamente sfruttata nella fotografia medica [49], la segmentazione permette di evidenziare in una immagine le differenze non solo di colore, ma anche logiche, delle entità in essa contenute. Tecniche di questo tipo, ad esempio, vengono utilizzate nelle operazioni di "taglio" tra oggetti di diversa natura nella stessa scena [50], o per dividere i primi piani dalle entità sullo sfondo.

Uno degli algoritmi che rappresentano il "punto di partenza" per la realizzazione di metodi più evoluti è senza dubbio il *K-means*, che iterativamente divide un input in un numero prefissato di segmenti (o cluster), appunto K:

1. A random, o secondo una data euristica, si scelgono K punti interni allo spazio.

2. Per ogni punto nello spazio, vi si assegna il cluster tale per cui è minimizzata la distanza (intesa come metrica di similarità, come lo sono SAD e SSD per le intensità di grigio) tra il punto e il centro del cluster.

3. Si aggiorna il valore del punto centrale di ogni cluster con la media dei valori racchiusi in esso.

4. Si ripete dal punto 2 fino alla convergenza (ad esempio quanto non esistono più pixel senza etichetta).

La complessità computazionale di questa procedura è $O(NKI)$, dove N è il numero totale di pixel, K il numero di cluster e I il numero di iterazioni prima di arrivare alla convergenza. E' evidente che, per valori alti di I, l'algoritmo risulta piuttosto inefficiente. Da questo sono nate tecniche ottimizzate, ad esempio basate su grafi [51], che garantiscono una complessità che dipende esclusivamente da N. Lo svantaggio di tali soluzioni è il poco controllo sul numero di segmenti risultanti e sulla loro dimensione.

Un algoritmo, invece, di recente ideazione, derivato da K-Means ma con complessità lineare $O(N)$ è *SLIC - Simple linear iterative clustering* [52]. Viene sfruttata una meno diffusa metrica di confronto tra un pixel e il centro di un cluster, basata sullo spazio-colore CIELAB, formato di scambio e conversione usato nei moderni programmi di imaging professionale[3].

Il set-up della procedura divide l'immagine in una griglia di celle quadrate di area S^2 (questa sarà intesa come la grandezza massima che ogni cluster potrà assumere). Il punto centrale di ogni cella è fissato come l'origine di un cluster. L'area di ricerca è dunque limitata a questo parametro.

Conoscendo le componenti (L, a, b, x, y) di colore e spazio di ogni pixel p, la distanza D da un centro k viene calcolata con la seguente formula:

$$\begin{aligned} d_{lab} &= \sqrt{(l_k - l_p)^2 + (a_k - a_p)^2 + (b_k - b_p)^2} \\ d_{xy} &= \sqrt{(x_k - x_p)^2 + (y_k - y_p)^2} \\ D &= d_{lab} + \frac{m}{S} d_{xy} \end{aligned} \tag{3.17}$$

Dove d_{lab} è la differenza sul colore, d_{xy} sullo spazio (entrambe intese come distanze euclidee) e m è un parametro che regola la "compattezza" di ogni cluster nel range $[1; 20]$: più è alto e più i segmenti assumeranno la forma quadrata della cella d'origine.

Implementato in SuperStereo, con alcune modifiche alle strutture dati per ottimizzare l'utilizzo di RAM e calcoli in virgola mobile, SLIC ha permesso di ricostruire i bordi originali delle forme nella mappa generata segmentando la sola immagine di sinistra. In particolare, definita una sezione, il colore al suo interno è la *moda* dei valori di profondità di tutti i pixel che la compongono. Tale passaggio ha anche permesso di eliminare ogni lieve rumore residuo della fase precedente di smoothing (Figura 3.16).

[3]Si può trovare un'analisi dello spazio-colore CIELAB a questo indirizzo: `http://www.colourphil.co.uk/lab_lch_colour_space.shtml`, mentre un algoritmo di conversione dallo spazio RGB a CIELAB esiste ed è disponibile al sito `https://github.com/THEjoezack/ColorMine/tree/master/ColorMine/ColorSpaces/Conversions`

Figura 3.15: Esempi di applicazione di SLIC variando il numero di cluster

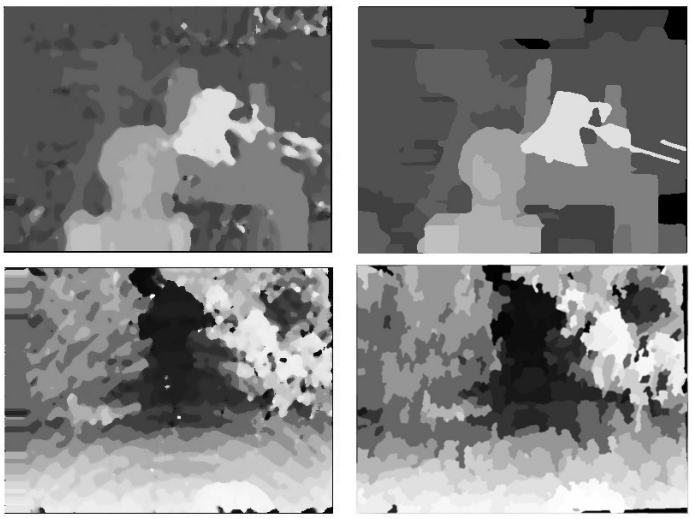

Figura 3.16: Confronto tra mappe di disparità prima e dopo l'operazione di segmentazione

3.3.5 Valutazione delle prestazioni

La possibilità di accedere liberamente al dataset Middlebury ha permesso di quantificare, con uno strumento divenuto standard, la qualità dei risultati offerti dal modulo di stereo matching fino a qui implementato.

Il dataset, in particolare, mette a disposizione una serie di coppie stereo, già allineate e con caratteristiche visive diverse tra loro, ognuna affiancata dalla propria mappa di disparità reale. Per questa analisi, sono state prese in considerazione quattro immagini.

La valutazione avviene confrontando l'output dell'algoritmo con la disparità originale di ogni immagine presa in considerazione, verificando tre metriche:

- Percentuale di pixel errati nell'intero spazio (*all*), tali per cui la differenza di valore con la disparità originale è maggiore di 1.0
- Percentuale di pixel errati nelle sole texture uniformi (*untex*)
- Percentuale di pixel errati nelle sole zone dove avvengono cambi di profondità (*disc*)

La Tabella 3.1 mette in relazione i risultati con le metriche proposte, assieme ad alcuni algoritmi di diverse tipologie. In Figura 3.17, invece, sono mostrate le mappe prodotte, oltre alle immagini di riferimento.

	Tsukuba			Venus			Teddy			Cones		
	all	untex	disc	all	untex	disc	all	untex	disc	all	untex	disc
AdaptingBP	1.11	1.37	5.79	0.10	0.21	1.44	4.22	7.06	11.8	2.48	7.92	7.32
SemiGlob	3.26	3.96	12.8	1.00	1.57	11.3	6.02	12.2	16.3	3.06	9.75	8.90
SuperStereo	2.35	3.75	9.0	1.22	1.81	9.38	11.8	18.4	25.5	6.83	13.8	11.6
GC	1.94	4.12	9.39	1.79	3.44	8.75	16.5	25.0	24.9	7.70	18.2	15.3
DP	4.12	5.04	12.0	10.1	11.0	21.0	14.0	21.6	20.6	10.5	19.1	21.1

Tabella 3.1: Punteggi ottenuti nel dataset Middlebury

Struct	Space (in Bytes)	for a 640x480 image pair K = 200 (in KBytes)
Left + Right Census data	N*2	614
LeftCosts + RightCosts	(N*4)*2	2.458
MinCosts	N*4	1.229
Segmentation	*(N*4)*3 + (K*5)*4*	*4.086*
Total[4]	**N*14**	**4.301**

Tabella 3.2: Spazio in RAM occupato dalle strutture utilizzate (i nomi si riferiscono a quelli usati nell'algoritmo 1). N è il numero di pixel che compongono ogni immagine di input, K il numero di segmenti

Gli algoritmi inseriti in questa ristretta classifica sono stati scelti, rispetto ai molti altri, per una serie di motivi: *AdaptingBP* [39], già citato in precedenza, è attualmente uno dei tre metodi considerati al "top" per qualità delle mappe generate. Come SuperStereo, anch'esso nella fase iniziale implementa una ricerca locale delle corrispondenze, con la differenza che essa avviene internamente a determinate regioni, definite in precedenza con un'operazione di segmentazione. A seguire, altri due step di miglioramento, questa volta basati su grafi, aumentano il numero di corrispondenze esatte.

Tali eccellenti risultati mostrano, tuttavia, quanto sia difficile uniformare la percentuale di pixel corretti nelle zone con variazioni di profondità (*disc*) rispetto agli altri due casi. L'immagine *Teddy*, in particolare, contiene molte di queste variazioni e tecniche veramente efficaci in questo caso non esistono ancora.

DP [56] e *GC* [57] sono, rispettivamente, tecniche di stereo matching basate su programmazione dinamica e graph-cut, conosciute per essere efficienti in termini di tempo ma non nello spazio, a causa delle strutture dati necessarie alla loro esecuzione, la cui grandezza cresce velocemente all'aumen-

[4]Si precisa che nel totale non sono inclusi i dati necessari alla segmentazione (in corsivo). Ciò è legittimo dato che essa avviene successivamente all'algoritmo di matching, in un istante dove le strutture sopra citate sono già state liberate, non essendo più necessarie.

Step	Time (in seconds)
Structures initialization	2,172
Disparity computation	0,750
Median filter	0,235
Segmentation	4,750
Total	**7,907**

Tabella 3.3: Tempi di esecuzione della fase di stereo matching per una coppia di 640x480 pixel e disparità massima 40 pixel, su architettura *Intel Core2Duo E8400* @ 3,00 GHz. Codice scritto in C++, non parallelizzato e senza l'utilizzo di istruzioni ottimizzate quali *SSE*.

tare delle dimensioni dell'input [55]. SuperStereo, invece, riesce a garantire un punteggio migliore con una complessità lineare nel numero di pixel che compongono l'input (Tabella 3.2).

Un ultima considerazione va fatta riguardo all'algoritmo *Semi-Global Matching* [54], lievemente migliore di SuperStereo nei risultati qualitativi ma decisamente peggiore, anche in questa occasione, nell'impiego di memoria RAM. A fronte di un input composto da due immagini di $N = 640 * 480$ pixel, che differiscono per una disparità massima di $D = 64$ punti, lo spazio necessario è infatti di $N * D * 3 = 59$ MByte, contrariamente ai 4 MByte nel metodo qui proposto, il quale è indipendente dal valore di disparità.

Per quanto riguarda le prestazioni in merito alla velocità di esecuzione, la progettazione ha considerato fin dall'inizio le ridotte capacità di calcolo dei dispositivi cellulari. Purtroppo, i criteri di valutazione dell'istituto di Middlebury non considerano tale aspetto ed un confronto diretto con lo stato dell'arte si rivela piuttosto complicato. Nel capitolo successivo, l'attenzione verterà sui tempi di esecuzione di SuperStereo sulle piattaforme finali.

I test, invece, in ambiente desktop non parallelizzato e privo di particolari ottimizzazioni a livello di *instruction set*, hanno fin da subito registrato risultati molto interessanti, riportati in tabella 3.3, soprattutto per quanto

riguarda la fase principale di computazione ed aggregazione delle corrispondenze, che raggiunge tempistiche vicine al real-time. E' invece un processo più lungo quello necessario a segmentare l'immagine sinistra, ma ciò non esclude possibili miglioramenti futuri. L'algoritmo, infatti, è modulare ed il metodo in questione può essere sostituito facilmente con uno più performante.

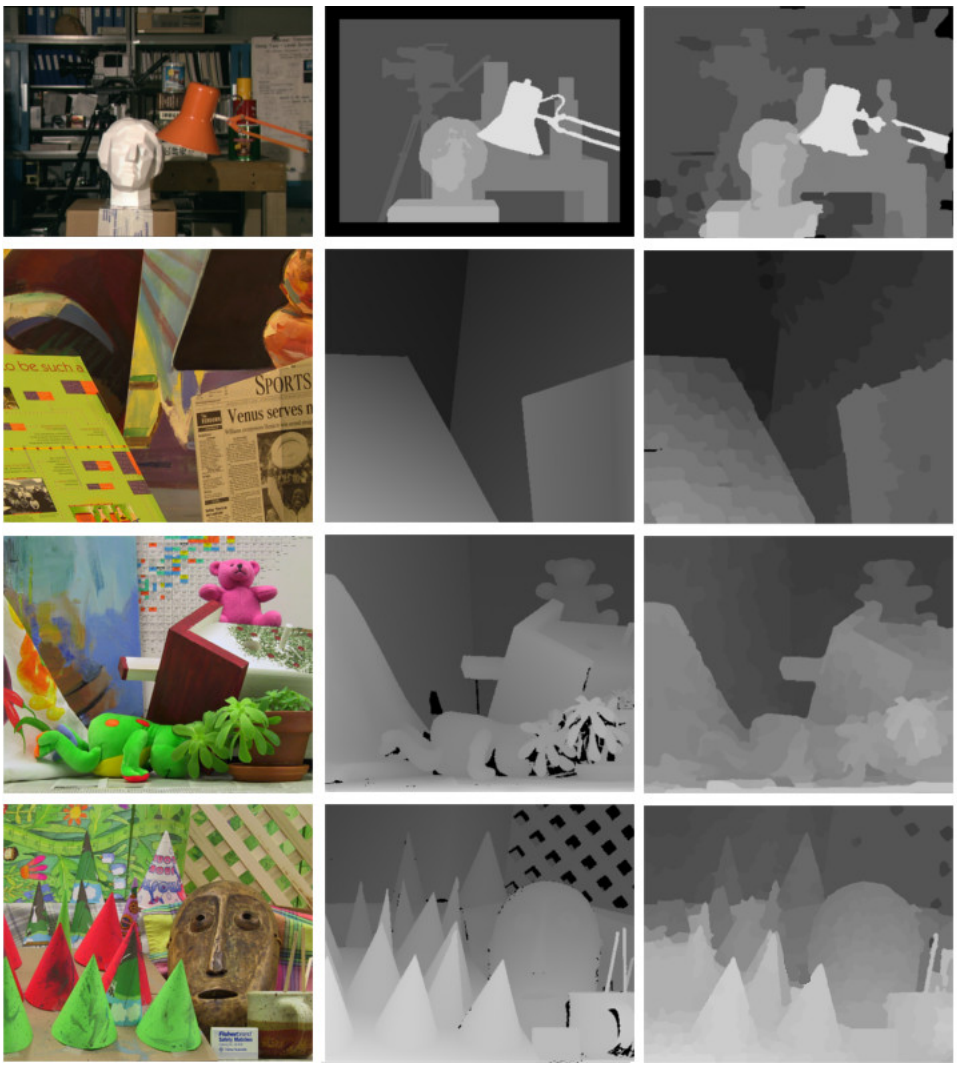

Figura 3.17: Risultati dell'algoritmo implementato in SuperStereo, confrontati con il dataset ufficiale. Da sinistra verso destra: immagine originale (solo la prima della coppia), la mappa di riferimento e quella prodotta in fase di test. I nomi utilizzati per ogni scena sono, dall'alto verso il basso: *Tsukuba*, *Venus*, *Teddy* e *Cones*.

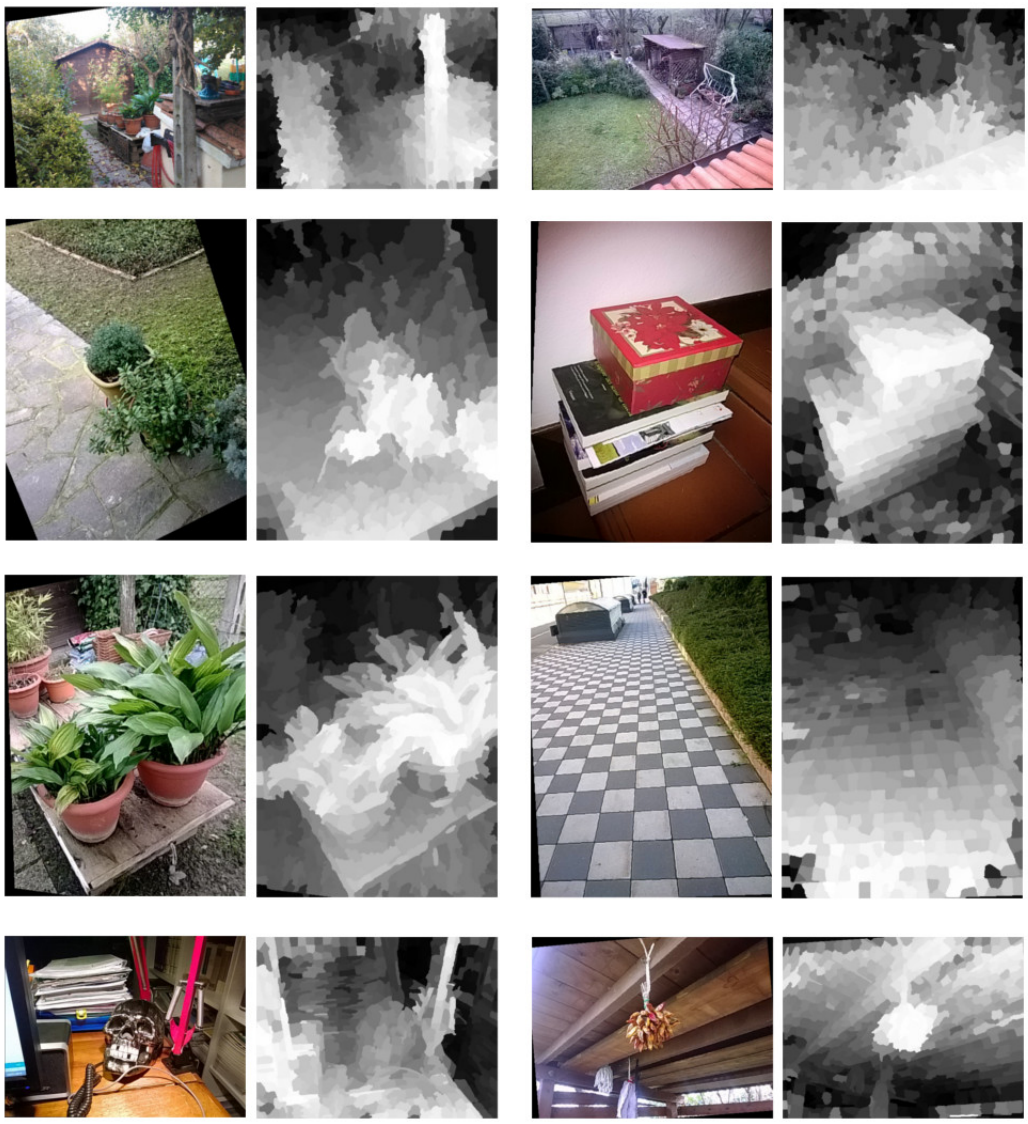

Figura 3.18: Mappe di disparità di immagini reali, catturate utilizzando uno smartphone di fascia media (*Nokia Lumia 620*) in diverse condizioni ambientali.

3.3.6 Differenza con le tecnologie laser

Per completezza, non essendo lo stereo matching l'unica tecnica disponibile a questi scopi, in Figura 3.19 viene proposto un confronto visivo tra la depth map generata da un sensore basato su tecnica laser e reperibile in ambiente consumer, *Kinect* di *Microsoft*[58], e lo stesso soggetto elaborato da SuperStereo.

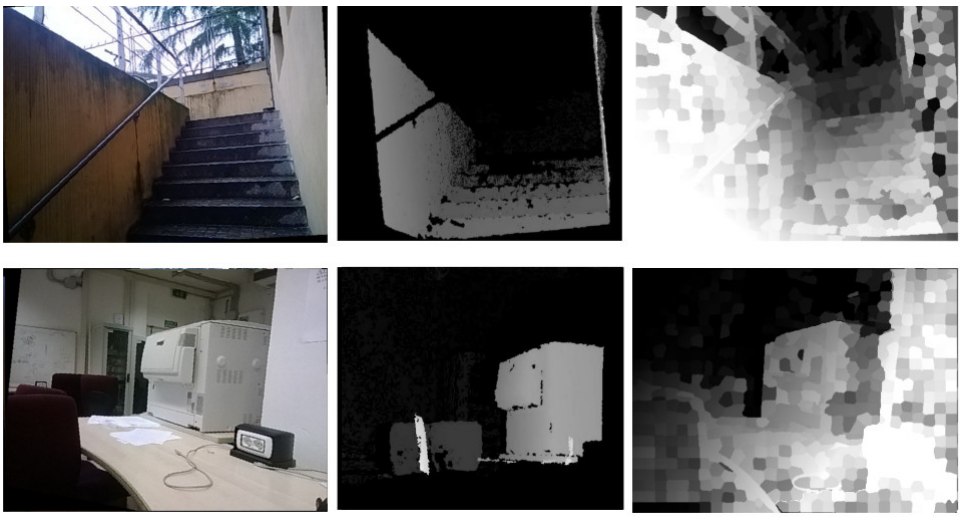

Figura 3.19: Comparazione visiva dei risultati in due differenti ambienti, catturati con Microsoft Kinect (al centro) e SuperStereo (a destra).

Osservando i risultati di Kinect, sono evidenti i limiti imposti dall'hardware basato su laser, primo fra tutti la "profondità di campo", intesa come il limite spaziale oltre il quale la sensibilità del sensore diventa nulla. Tramite un metodo puramente software come quello proposto, che esegue l'analisi su pixel, è possibile rilevare la disparità anche per soggetti in lontananza, purchè visibili nelle due bitmap in input.

Se il vantaggio principale del metodo proposto è la totale indipendenza da accessori fisici (che ne garantisce un'elevata mobilità), gli svantaggi che si incontrano sono quelli propri dell'analisi su intensità di colore, già descritti in Sezione 3.3.1.

3.4 Algoritmo 3: Divisione in livelli e traslazione

Grazie agli algoritmi fino a qui presentati, l'applicazione SuperStereo è in grado di rappresentare la profondità della scena catturata dall'utente tramite la generazione di una mappa di disparià densa. Il prossimo ed ultimo passo per completare l'intero processo ed arrivare all'effetto finale desiderato consiste nella divisione in livelli della prima immagine acquisita, traslandoli di conseguenza.

Definiti L livelli sotto forma di nuove bitmap, inizialmente vuote, si considera la sola immagine sinistra e per ogni pixel $I(x; y)$ al suo interno si determina il livello l dentro al quale verrà copiato. L'operazione che restituisce tale indice mappa la disparità assegnata in (x, y) nell'intervallo [0; L-1] tramite l'Equazione (3.18).

$$l = \frac{L}{256} * \hat{M}(x, y) \tag{3.18}$$

dove $\hat{M}$ è la mappa di disparità, con valori normalizzati nel range [0; 255].

L'idea è effettivamente molto semplice ma soffre di un evidente problema: è stato detto che due livelli appartenenti a profondità diverse devono essere traslati, in tempo reale e a seconda dell'inclinazione del dispositivo, di due misure differenti.

Consideriamo, ad esempio, un'immagine composta da L segmenti ed una ristretta area nella quale due livelli di indice l e $l + 1$ (quest'ultimo, dunque, più vicino all'osservatore), sono tra loro adiacenti. Inclinando il device di un grado $\Delta\alpha$ sull'asse x, rispetto alla posizione iniziale, ciascuno di essi verrà traslato orizzontalmente di un offset σ, simulando un effetto di parallasse:

$$\sigma_l = \frac{(O_L/A) * \Delta\alpha}{L} * l \tag{3.19}$$

Dove O_L e A sono, rispettivamente, la massima traslazione del layer in primissimo piano (con indice $L - 1$) e il massimo grado di inclinazione del dispositivo. La formula, applicabile alla stessa maniera per una traslazione

verticale, fa quindi in modo che il livello $l+1$ si possa spostare di un fattore maggiore di quello sottostante.

A seguito di questo, un certo numero di occlusioni lungo il bordo delle due zone vengono inevitabilmente mostrate. Si tratta di quei punti che nella loro rappresentazione 2D non possono esistere, poichè coperti da oggetti in primo piano nella realtà. Si veda un esempio nella Figura 3.20.

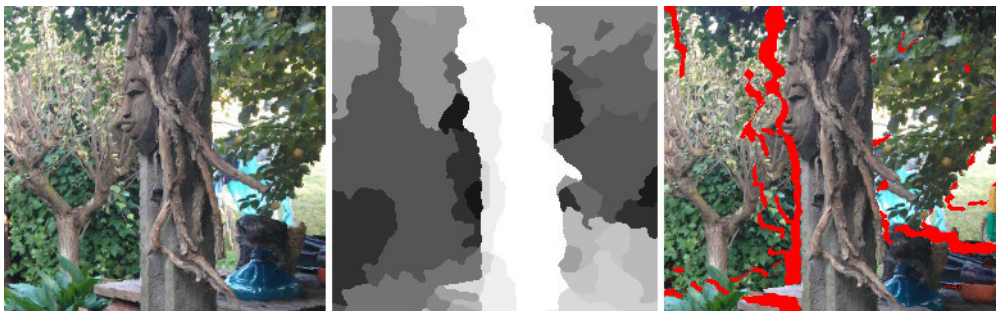

Figura 3.20: Pixel mancanti a seguito di una traslazione dei livelli (in rosso). Si noti come tali aree siano più ampie in corrispondenza di livelli in primo piano

Risulta dunque necessario fornire a ciascuno di tali pixel, inizialmente non visibili, un opportuno colore per il rendering finale. Possiamo interpretare l'intero processo come un'approssimazione del concetto di *view-synthesis* [59], materia della computer vision studiata per generare nuove viste partendo da più immagini dello stesso soggetto.

3.4.1 L'algoritmo sviluppato

La procedura, come d'altronde quelle viste fin'ora, è stata pensata per essere efficiente in spazio e tempo computazionale, in particolare perchè necessita di essere chiamata ogni qualvolta ci sia bisogno di visualizzare una foto catturata con SuperStereo. Questo vincolo è stato imposto per questioni legate al formato con il quale ogni "super-immagine" verrà salvata in memoria. Anzichè scrivere, in unico file, tutti gli L livelli generati, impiegando un quantitativo pari a $(N*4)*L$ Byte (con N dimensione dell'immagine

originale, ipotizzando un peso di 4 Byte per pixel), si è scelto di salvare esclusivamente l'immagine di sinistra e la sua profondità (la quale necessita di un solo Byte per punto).

Aperto questo pacchetto composto da soli $(N * 4) + N$ Byte, la procedura genera i livelli a run-time, copiando ogni pixel nel rispettivo layer e riempiendo le occlusioni con l'Algoritmo 2.

Algorithm 2 Algoritmo di riempimento delle occlusioni

for all $x, y \in N$ **do**
  if $p(x, y)$ is on the horizontal edge of layer l **then**
    if previous layer $l' > l$ **then**
      $R(p) \leftarrow \{\ p(x + r; y)\ \mid\ r \leq R\ \}$
      stretch $R(p)$ from x to $x - (O_{l'} - O_l)$ in layer l
    else if previous layer $l' < l$ **then**
      $R(p) \leftarrow \{\ p(x - 1 - r; y)\ \mid\ r \leq R\ \}$
      stretch $R(p)$ from $x - 1$ to $x + (O_l - O'_l)$ in layer l'
    end if
  end if
  if $p(x, y)$ is on the vertical edge of layer l **then**
    if upper layer $l' > l$ **then**
      $R(p) \leftarrow \{\ p(x; y - r)\ \mid\ r \leq R\ \}$
      stretch $R(p)$ from y to $y + (O_{l'} - O_l)$ in layer l
    else if upper layer $l' < l$ **then**
      $R(p) \leftarrow \{\ p(x; y + r + 1)\ \mid\ r \leq R\ \}$
      stretch $R(p)$ from y to $y - (O_l - O_{l'})$ in layer l'
    end if
  end if
end for

Per ogni punto $p(x; y)$, si controlla se giace sul bordo del livello assegnatogli. In caso positivo, viene considerato un segmento lineare di R pixel adiacente a p (compreso p stesso), "allungandolo" in direzione dello spazio vuoto immediatamente vicino al bordo. La nuova lunghezza di tale sezione

dipende dal massimo spostamento che può assumere il livello più in alto, meno quello del livello inferiore (altro non è che la lunghezza massima che assumerà la zona scoperta).

L'algoritmo è dunque molto efficiente, avendo bisogno di visitare gli N pixel una sola volta, durante il quale, in casi speciali, deve copiare un massimo di O_L punti per riempire un'area occlusa.

Figura 3.21: Applicazione del metodo di riempimento delle zone occluse

Capitolo 4

Implementazione su piattaforme mobili

Dopo aver descritto le tre operazioni fondamentali per visualizzare una scena con un effetto 3D, questo capitolo tratta degli aspetti implementativi affrontati durante il porting da una prima versione di test, su architettura desktop, verso due delle principali piattaforme mobili attualmente presenti sul mercato: *Windows Phone* e *Android*.

E' in questa occasione che l'applicazione SuperStereo ha veramente preso forma, consentendo di eseguire prove qualitative dell'effettiva tridimensionaltà del risultato, anche per mano di utenti esterni al progetto.

4.1 Windows Phone

Windows Phone è un sistema operativo, di proprietà di *Microsoft*, dedicato a dispositivi mobili quali smartphone e tablet. Contrariamente al suo predecessore *Windows Mobile*, dedicato ad ambienti business, questa piattaforma è stata concepita per affermarsi in un mercato decisamente più consumer, grazie ad un'interfaccia totalmente rinnovata ed un catalogo di device supportati a prezzi variabili. L'azienda *Nokia*, per prima, produce la mag-

gior parte dei modelli sui quali Windows Phone è installato, caratterizzati dal nome *Lumia*.

Al momento dello sviluppo di SuperStereo, tale sistema operativo era presente sul mercato in due versioni, 7.8 e 8.0. Pressochè identiche nella maniera con cui l'utente vi si interfaccia, si differenziano sensibilmente a livello architetturale. Il kernel di Windows Phone 8.0, in particolare, condivide molti aspetti di *Windows 8*, per piattaforme desktop. Ne deriva la possibilità di scrivere applicazioni mobili sfruttando il framework *WinRT*, il quale supporta il linguaggio C# e, in special modo, C++ per eseguire più facilmente il porting di codice già testato per altre architetture, con un occhio di riguardo alla velocità d'esecuzione.

Windows Phone 7.8, invece, permette lo sviluppo utilizzando esclusivamente il primo dei due linguaggi, unito al diffuso framework *Silverlight* [60].

Trovandosi, dunque, in un momento nel quale due versioni della medesima piattaforma occupavano entrambe il mercato, il gruppo di lavoro ha dovuto affrontare una scelta circa la compatibilità minima garantita. Si è scelto di procedere allo sviluppo mantenendo il target di compilazione alla versione 7.8, poichè, al tempo, i device che supportavano tale release coprivano una percentuale di distribuzione maggiore rispetto a quelli abilitati a Windows Phone 8, comunque retrocompatibili.

4.1.1 Il codice gestito C#

La scelta ha di certo aumentato le possibilità di distribuzione del prodotto, anche se, per quanto ne concerne lo sviluppo, il limite di utilizzo del solo linguaggio C# ha reso necessari una serie di accorgimenti sul codice. C# è infatti un linguaggio ad oggetti *gestito*, termine coniato da Microsoft stessa per definire una famiglia di linguaggi di programmazione ad alto livello che, al momento della compilazione, vengono tradotti in un comune codice intermedio (*Common Intermediate Language*). Per mezzo di una macchi-

na virtuale apposita (*Common Language Runtime*), tale codice viene poi tradotto a run-time dal cosiddetto *JIT - Just In Time compiler*.

Ne derivano una serie di vantaggi in termini di sicurezza, ottimizzazione e velocità di scrittura del codice:

- Eliminazione di "codice morto"
- Gestione automatica di allocazioni e deallocazioni (*garbage collection*)
- *Inlining* dei metodi
- Riduzione delle espressioni con valori costanti (*constant folding*)
- Ottimizzazione dei loop (*loop unrolling*)
- Controllo automatico di eventuali overflow

Se tali procedure possono apparire utili per determinate situazioni, è implicito l'inserimento di istruzioni da parte di JIT all'interno del codice compilato. Queste aggiunte rischiano di rendere gli algoritmi molto meno veloci delle controparti scritte, ad esempio, in C++. Per fare un esempio fortemente connesso con il tipo di applicazione sviluppata, si consideri l'applicazione del seguente filtro su un'immagine a colori:

Algorithm 3 Conversione dallo spazio-colore RGB a Luminanza

for all $x, y \in N$ **do**

$I'(x,y) \leftarrow I_R(x,y) * 0.2126 + I_G(x,y) * 0.7152 + I_B(x,y) * 0.0722$

end for

La versione di tale algoritmo in codice gestito C# risulta essere, in media, 26 volte più lenta della versione scritta in codice nativo C++. E' stato, dunque, di fondamentale importanza considerare alcune ottimizzazioni, come l'utilizzo della parola chiave *unchecked* [61] in corrispondenza di operazioni aritmetiche sugli interi. Tale costrutto fa in modo che JIT non controlli eventuali overflow di memoria scaturiti da questo tipo di istruzioni (la correttezza è quindi a discrezione dello sviluppatore stesso), con una sensibile riduzione dei tempi d'esecuzione.

4.2 Android

Android è un sistema operativo mobile basato sul kernel Linux e su un modello open source, con componenti software integrati proprietari di *Google*. Al momento della scrittura di questo documento (Gennaio 2014), tale sistema occupava il 68,8% del mercato mobile in Europa, contrariamente a Windows Phone, il quale deteneva "solamente" il 10,3%[1].

Android, arrivata alla versione 4.4, è stata scelta come la seconda piattaforma sulla quale implementare il software SuperStereo. I linguaggio principale per lo sviluppo di applicativi in questo caso è Java, sia per quanto riguarda il controllo dell'interfaccia grafica che per la scrittura del *back-end*. Inoltre, da tale linguaggio è possibile richiamare procedure scritte in codice nativo attraverso il *Native Development Kit*, una possibilità molto interessante che ha permesso di garantire velocità di esecuzione soddisfacenti anche sui modelli meno evoluti.

4.2.1 Native Development Kit

Per Native Development Kit, abbreviato in *NDK*, si intende una collezione di strumenti, in aggiunta a quelli forniti dal *Software Development Kit* di Android, per l'implementazione di procedure (o interi moduli nell'applicazione) in un linguaggio più a basso livello, in questo caso C++ [62]. L'utilizzo di tale ambiente non è obbligatorio ai fini di produrre una buona "app", è anzi indicato solamente in determinate circostanze: nel caso in cui, ad esempio, si intendesse importare codice progettato in precedenza, senza la necessità di convertirlo in Java. Inoltre, l'NDK dà la possibilità di agire su uno strato di progettazione più interno ad Android, bypassando la macchina virtuale di Java e interfacciandosi direttamente con l'hardware del device, per averne un controllo maggiore.

[1] Necessita di essere citato anche il sistema operativo *iOS* di *Apple* che, pur non essendo oggetto di studio in questa tesi, nel Gennaio 2014 deteneva il 18,5% del market-share europeo. Fonte: *Kantar Worldpanel ComTech*.

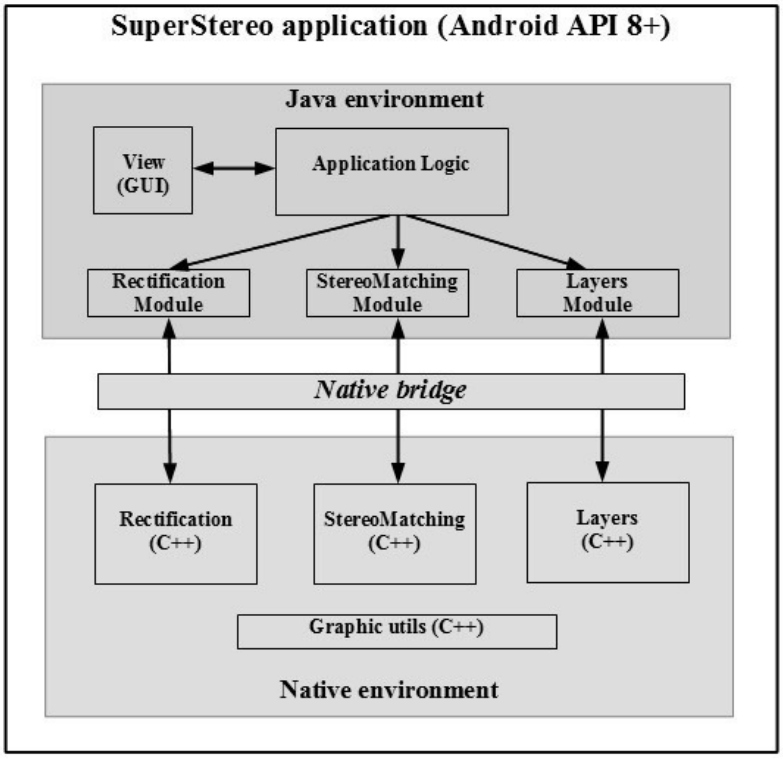

Figura 4.1: Schema dell'interoperabilità tra lo strato gestito in Java e quello nativo in SuperStereo per Android

Le nostre necessità rientrano nella prima di tali categorie, avendo precedentemente testato il progetto in ambiente desktop, e implicitamente in una terza. Già analizzata durante lo sviluppo di Windows Phone, si tratta della possibilità di beneficiare dei sensibili guadagni prestazionali offerti da un codice a basso livello.

E' stato implementato, quindi, un sistema di "comunicazione" tra i moduli scritti in Java e quelli che operato nello strato nativo (Figura 4.1). Ogni qualvolta sia necessario uno scambio di dati tra l'uno e l'altro ambiente, viene interpellato un *bridge* che trasmette la richiesta di input (ad esempio una coppia di immagini da rettificare), al componente nativo dedicato. Una volta completata l'operazione, il risultato viene allocato in una struttura dati leggibile dall'ambiente Java e trasmesso al modulo che l'aveva richiesto.

4.3 Prestazioni sui device di test

Viene ora descritta una panoramica dei risultati ottenuti nei dispositivi di test, in termini di velocità di esecuzione dei vari algoritmi. La Tabella 4.1 elenca le varie specifiche tecniche di nostro interesse.

Model	Operative system	CPU	RAM (app limit)	Image size
Nokia Lumia 800 *(NL8)*	Windows Phone 7.8	Q-MSM8255 @ 1.4 GHz *SC*	150 MB	640x480
Nokia Lumia 620 *(NL6)*	Windows Phone 8.0	Q-S4 @ 1.0 GHz *DC*	150 MB	640x480
LG-E720 *(LGE)*	Android 2.2.1	Q-MSM7227 @ 600 MHz *SC*	16 MB	320x240
Sony Xperia L *(SXL)*	Android 4.1	Q-MSM8230 @ 1 GHz *SC*	128 MB	640x480
Galaxy Nexus *(SGN)*	Android 4.3	A-Cortex-A9 @ 1.2 GHz *DC*	256 MB	640x480

Tabella 4.1: Caratteristiche tecniche dei dispositivi utilizzati nella fase di test. Le sigle *SC* e *DC* nella colonna CPU identificano processori single-core o dual-core.

Si noti l'eterogeneità dell'hardware a disposizione, sotto diversi aspetti: tipo e versione del sistema operativo, potenza di calcolo e spazio in RAM disponibile per una singola istanza dell'applicazione. L'analisi ha interessato le operazioni di stereo vision descritte in precedenza, a fronte di una coppia di immagini le cui dimensioni sono riportate nella colonna "Image size". Per questioni legate al tipo di hardware di ogni dispositivo ed alle caratteristiche della fotocamera, durante il test non si è potuta fissare una dimensione dell'immagine uguale per tutti (è il caso del device *LGE*, con un Per questioni legate al tipo di hardware di ogni dispositivo ed alle caratteristiche della fotocamera, durante il test non si è potuta fissare una dimensione dell'immagine uguale per tutti (è il caso del device *LGE*, con un quantitativo di memoria troppo basso per poter gestire il formato 640x480). Ad ogni modo, i risultati ottenuti presentano dettagli interessanti, mostrati in Figura 4.2.

Nel grafico, una media del tempo di esecuzione di ogni algoritmo, su un campione di 5 test per dispositivo, è stata messa a confronto. Nell'ordine, sono stati misurati i tempi per le fasi di rettificazione (R), computazione delle corrispondenze (SMc), segmentazione (SMs) e divisione in livelli (L).

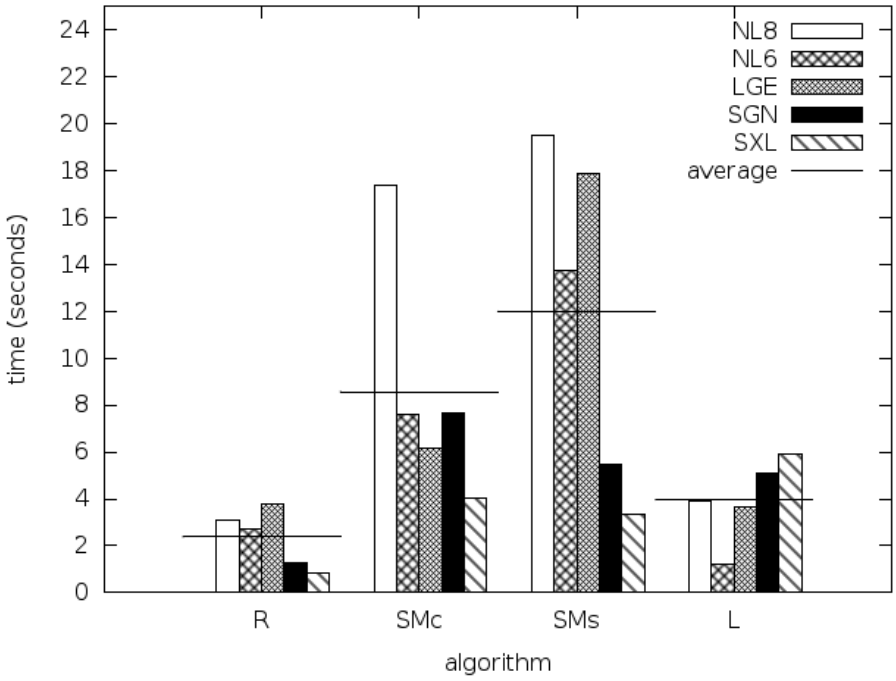

Figura 4.2: Media (in secondi) dei tempi d'esecuzione degli algoritmi su ogni dispositivo di prova. Dati generati da un campione di 5 prove per istanza di test

Si nota, innanzitutto, la netta differenza rispetto alle tempistiche ottenute precedentemente in ambiente desktop, un dettaglio tenuto in considerazione fin da subito. Piuttosto inaspettata, invece, è la disparità che intercorre tra i device *NL8* e *NL6*, che, pur avendo entrambi installata una versione di SuperStereo scritta in codice non nativo, gestiscono il lavoro con un diverso grado di ottimizzazione. Il primo device risulta il più lento di tutti quelli provati, pur montando un processore di fascia alta (ma single-core). Nel secondo, invece, l'architettura alla base di Windows Phone 8.0 riesce evidentemente a garantire performance migliori senza che lo sviluppatore debba introdurre modifiche al codice[2].

Un secondo dettaglio conferma, invece, i vantaggi introdotti dall'utiliz-

[2]Si ricorda che, in questa prima versione, nessun algoritmo è stato strutturato per sfruttare un'architettura multiprocessore

zo di codice nativo sulle piattaforme Android. Grazie all'NDK, anche il dispositivo meno evoluto, *LGE*, ha potuto vantare tempistiche che non compromettono il normale utilizzo dell'applicazione. Lo stesso, ovviamente, vale per i rimanenti device, i quali impiegano tempi nettamente migliori di quelli sulla piattaforma Windows Phone.

4.3.1 Casi di successo e limitazioni

Oltre ad un'analisi prettamente legata ai tempi di attesa che intercorrono tra la cattura di una scena stereo e la sua visualizzazione, nel caso di SuperStereo è doveroso evidenziare anche le condizioni che permettono una resa qualitativamente migliore dei risultati, assieme ai casi per i quali l'effetto può ancora essere sottoposto a miglioramenti.

Nella rettificazione

La fase di rettificazione, come spiegato in Sezione 3.2, non include esclusivamente un metodo per allineare una coppia stereo. Precedentemente all'algoritmo implementato, infatti, è stato necessario introdurre un sistema che riconoscesse punti corrispondenti alle due viste. Durante i test, non è mancato di analizzare anche la precisione di tale operazione: sono stati valutati casi in cui essa restituisce un insieme di match, alcuni dei quali errati. Trasmessi come input per la rettificazione, quest'ultima non impiega, purtroppo, un sistema abbastanza robusto da generare due trasformazioni che possono essere ritenute valide per i nostri scopi.

Il risultato rischia di essere, infatti, una trasformazione che rende le due immagini iniziali incomprensibili. E' dunque necessario prevenire il riconoscimento di corrispondenze errate fin da subito, durante la fase di keypoint-matching, attraverso una serie di vincoli introdotti per "filtrare" i match sbagliati. Dato un match M tra due punti p_L e p_R, il primo vincolo impone che la distanza di Hamming tra i relativi descrittori sia un valore inferiore ad una soglia molto bassa (nei test fissata a 15 a fronte di stringhe lunghe 256 bit).

Il secondo ed ultimo vincolo impone, invece, che le distanze tra le coordinate orizzontali e quelle verticali di p_L e p_R siano anch'esse molto limitate. Ci troviamo infatti in un caso "speciale" di keypoint-matching, dove le due viste si differenziano solo per una piccola traslazione nello spazio. Non risulta utile, dunque, ricercare una corrispondenza per distanze molto ampie.

Tali accorgimenti rendono dunque le istanze errate piuttosto rare. E' indubbio che, in futuro, un aggiornamento di tale fase possa impiegare controlli ancora più raffinati, su basi statistiche o vincoli della stessa geometria epipolare (come ORSA o RANSAC [63]).

Nello stereo matching

Le problematiche che, invece, vanno affrontate durante uno stereo matching denso, basato sul colore, sono state descritte nella Sezione 3.3.1. L'algoritmo qui progettato riesce a rappresentare con buona precisione una mappa di disparità, soprattutto a fronte del poco tempo e delle modeste risorse impiegate. La fase di segmentazione riduce efficacemente le zone nelle quali la stima della profondità fallisce, ma solo se tali aree sono ristrette a piccole *patch*.

E' un problema diffuso, infatti, quello di ricavare la disparità in una zona, ad esempio, dai colori uniformi. A fronte di grandi superfici prive di texture, SuperStereo non riesce ad assegnarvi un giusto valore, approssimando piuttosto l'area ad un colore tendente al nero (Figura 4.3/a).

Un secondo caso che rende il risultato lontano dal poter essere ritenuto valido è, anch'esso, un problema aperto nello stereo matching a camera singola: la creazione di una mappa partendo da una scena che, durante la fase di acquisizione indiretta, ha cambiato la sua struttura. E' ancora impossibile, infatti, stimare una profondità densa e precisa in queste condizioni, mantenendo la ricerca esclusivamente lungo l'asse orizzontale e senza aumentare la complessità algoritmica, a sfavore dell'usabilità su hardware mobile.

Fortunatamente, in SuperStereo viene fornito un sistema di acquisizione abbastanza rapido da evitare la maggior parte di queste situazioni. Ambienti

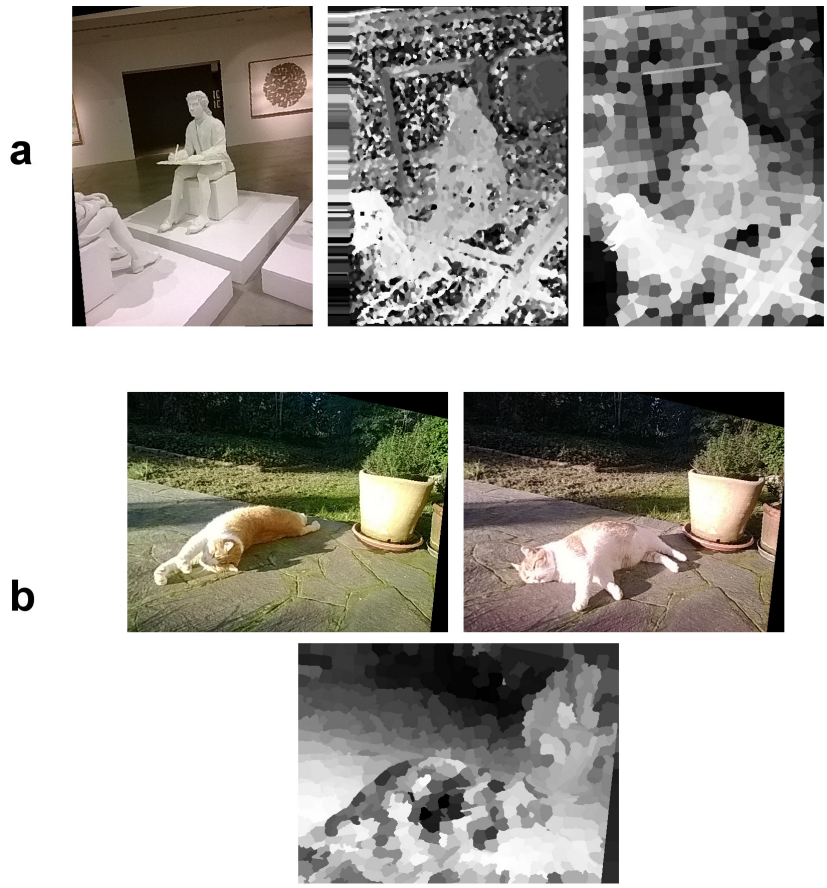

Figura 4.3: Due casi in cui lo stereo matching fallisce. In alto, in presenza di grandi superfici prive di texture (prima e dopo la segmentazione). In basso, quando tra le due viste vi è un evidente cambiamento strutturale.

che si modificano per piccoli dettagli (si pensi ad un albero i cui rami sono mossi dal vento, o automobili in movimento sullo sfondo), non creano dunque problemi di sorta. Da segnalare, inoltre, un punto di forza dell'algoritmo di stereo matching progettato, visibile attraverso la Figura 4.3/b: pur fallendo la stima nei punti senza corrispondenze, la mappa è invece corretta nelle rimanenti aree anche quando è evidente un cambiamento di tonalità tra l'una e l'altra immagine, per la qualità del sensore utilizzato. Questo grazie, soprattutto, all'impiego della trasformata Census.

Nella divisione in livelli

Non sono da segnalare note particolari riguardo all'ultima fase, che permette una variazione della prospettiva in maniera "seamless", grazie quindi a movimenti fluidi e senza che l'utente percepisca gli artifici impiegati dall'algoritmo.

Ovviamente, dovendo impiegare un brevissimo tempo tra l'apertura del file e la visualizzazione dell'effetto voluto, questo metodo è stato ideato per essere veloce ma non il migliore possibile. L'interpolazione di uno sfondo artificiale per creare gli eventuali pixel mancanti nella scena si basa sul colore delle zone limitrofe, ma potrebbe offrire maggiore realismo se venissero considerate anche proprietà strutturali, logiche, della zona coperta.

Una migliore comprensione della zona mancante eviterebbe l'effetto di "allungamento" che il nostro metodo introduce in alcune situazioni, soprattutto quando l'analisi della profondità da parte dello step precedente non risulta sufficientemente corretta.

Si potrebbe, inoltre, prevedere la creazione real-time di un effettivo modello 3D della scena (Figura 4.4). Ciò sarebbe già possibile con le informazioni disponibili, ma risulterebbe un'operazione complessa per i dispositivi mobili più semplici, che non offrono hardware particolarmente dedicato alla rappresentazione di grafica tridimensionale. Anche se tale aspetto potrebbe risultare eludibile, limitando l'utilizzo di SuperStereo solamente su determinati dispositivi, un secondo conferma invece la necessità di mantenere il metodo corrente. Dai test visivi effettuati anche per mano di *beta tester* esterni, infatti, è risultato che il movimento di parallasse qui proposto permette una percezione migliore dell'effetto, rispetto alla visualizzazione di un modello 3D.

Figura 4.4: Modellazione 3D di scene precedentemente processate con SuperStereo. A sinistra, l'immagine originale. A destra, il solido generato in ambiente desktop, utilizzando le informazioni sulla profondità e la libreria *three.js* (`http://www.threejs.org`).

4.4 Schermate dell'applicazione

Di seguito, una serie di schermate provenienti dalla versione 1.0 di Super-Stereo per Windows Phone.

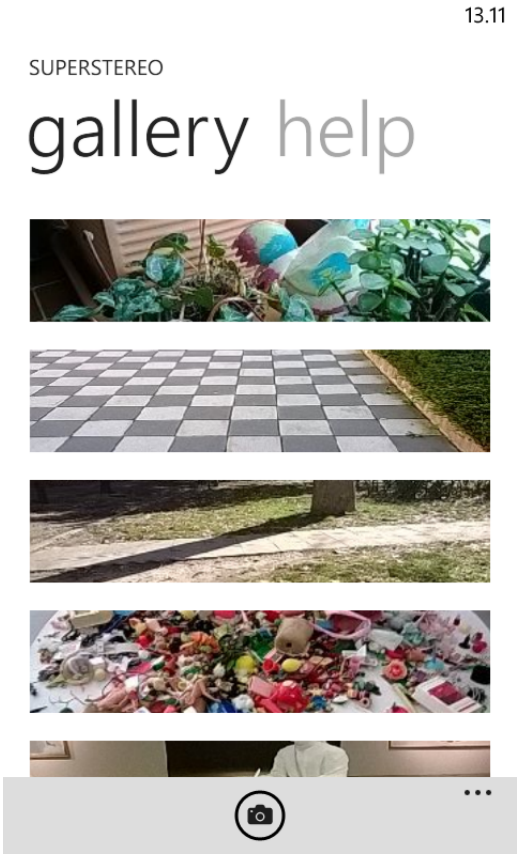

Figura 4.5: Schermata di avvio con una galleria delle foto acquisite.

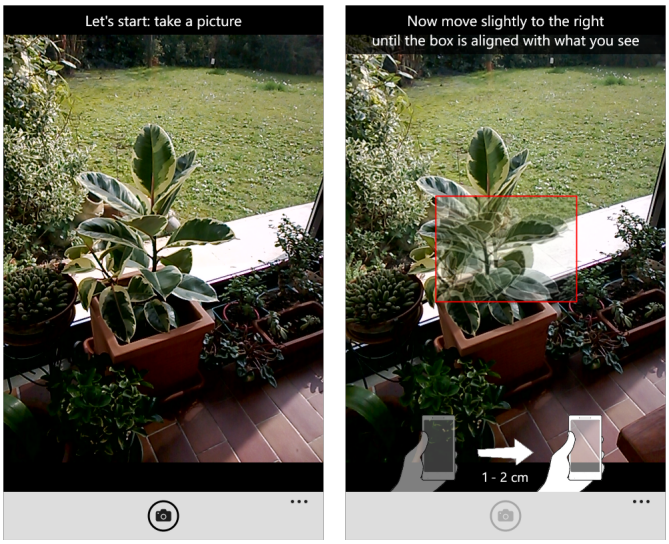

Figura 4.6: Le due fasi di cattura, con alcuni messaggi per guidare l'utente durante la procedura.

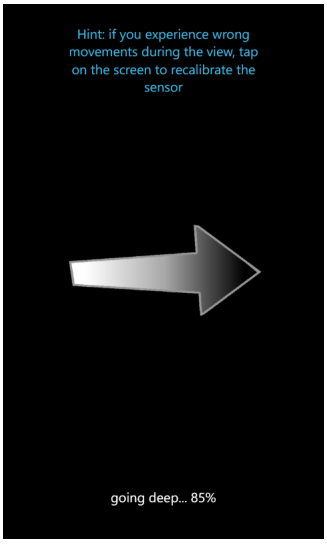

Figura 4.7: La schermata di processamento, mostra una barra di avanzamento in linea con lo stile grafico dell'app ed alcuni consigli per ottenere i risultati migliori.

Figura 4.8: Due istanti durante la visualizzazione dell'effetto. La linea verticale è stata inserita per dimostrare come livelli più vicini all'osservatore vengono traslati di un fattore maggiore rispetto a quelli sullo sfondo.

4.5 Confronto con lo stato dell'arte

La possibilità di implementare, in ambiente mobile, una tecnica di analisi della profondità e successiva fruizione è un aspetto studiato già da alcuni anni. Lo dimostra la presenza di soluzioni come quelle descritte nella Sezione introduttiva 2.1.

Si può facilmente notare, allora, l'eterogeneità di tecnologie disponibili oggi. Un'immaginaria "linea di confine" divide due grandi aree di studio: quella focalizzata all'implementazione di tecniche stereo che utilizzano supporti hardware (come dispositivi a doppia sorgente di cattura e/o muniti di schermi 3D) da quella, invece, che tenta di eliminare tali vincoli fisici. E' appurata la differenza, anche concettuale, che intercorre tra queste due categorie ed un metro di comparazione qualitativo tra di esse risulta essere, per natura, piuttosto complesso da definire.

Quanto alle caratteristiche di SuperStereo, i principali vantaggi sono quelli propri delle tecniche appartenenti alla seconda tipologia:

- Indipendenza da ogni tipo di accessorio fisico, sia durante la fase di cattura che di visualizzazione.

- Elevata portabilità e facilità di distribuzione, grazie all'implementazione su differenti sistemi operativi e, di conseguenza, alla possibilità di utilizzo su hardware eterogeneo.

Chiaramente, lo svantaggio è rappresentato dalle architetture sulle quale tali algoritmi possono funzionare: ambienti general-purpose, assolutamente non dedicati a questi scopi, introducono tematiche come l'eventuale scarsa qualità della singola fotocamera ed un hardware con il quale non è semplice raggiungere buoni risultati in tempo reale. Esistono, tuttavia, una serie di possibili miglioramenti come la parallelizzazione dell'algoritmo di stereo matching e l'utilizzo di metriche più precise per analizzare la disparità tra le due viste in input.

Contemporaneamente, lo scopo di questo elaborato è anche quello di migliorare il funzionamento delle attuali soluzioni esclusivamente software, o indirette. A questo proposito, SuperStereo porta i seguenti vantaggi:

- Possibilità di cattura di una doppia visuale in maniera semi-automatica, utilizzando una singola camera e richiedendo un contribuito minimo da parte dell'utente.
- Assenza di una fase di set-up.
- Visualizzazione dell'effetto tridimensionale basata su stereoscopia di parallasse, a livelli, anzichè su anaglifi o tecniche similari.

Svantaggi, in questo caso, possono emergere dal sistema di analisi delle immagini implementato. Una ricerca delle corrispondenze di tipo denso richiede che le immagini catturate siano comprensibili ed i punti affetti da ambiguità siano circoscritti a poche aree. Altre soluzioni, partendo da un numero maggiore di viste e con una ricerca sparsa, focalizzano l'analisi solo su punti "rilevanti" e descrivono la lontananza di questi dall'osservatore tramite triangolazioni geometriche. Con tali informazioni, approssimano dunque la profondità anche al resto della scena.

Capitolo 5

Conclusioni e sviluppi futuri

In questo elaborato è stata trattata la stereo vision quale interessante tecnica per il riconoscimento della profondità di una scena reale, da anni eseguita per mezzo di molteplici tipologie di strumenti, unendo soluzioni hardware e software sempre più precise.

Dopo averne introdotto i concetti generali, l'attenzione si è indirizzata verso un ambito che, attualmente, non riveste un ruolo particolarmente attivo in tale settore. A causa, soprattutto, dei limiti imposti dall'hardware rispetto alla versatilità delle architetture desktop, gli attuali dispositivi mobili quali smartphone e tablet non vengono ancora utilizzati in maniera diffusa per l'esecuzione di queste particolari operazioni, se non attraverso l'impiego di supporti fisici aggiuntivi. Un maggiore utilizzo di tali device in questo caso aprirebbe scenari nei quali un utente, anche non esperto, potrebbe sfruttare la visione stereo per i compiti più disparati, aiutato dalla loro facilità di utilizzo e diffusione.

Considerato questo concetto, è stato organizzato un ristretto team di sviluppo che, partendo da un attento studio dello stato dell'arte, ha rilasciato la prima versione di un software denominato SuperStereo, applicazione per dispositivi mobili general-purpose che, analizzando la profondità di una scena acquisita, ne permette la visualizzazione con un'evidente componente tridimensionale. A segnare un punto di innovazione per tale categoria di

applicativi è il fatto che l'intera procedura non necessita di strumenti ottici o altri supporti fisici per giungere al risultato desiderato. Il solo dispositivo, infatti, unito ad una semplice procedura di acquisizione e ad un metodo di visualizzazione esclusivamente software, ne è l'unico requisito.

Oltre ad aver definito, dunque, un diverso modo di visualizzare una fotografia su uno smartphone (un'operazione altrimenti classica ma che, ad oggi, ha dato vita a importanti fenomeni di *online business*), tale studio ha implicitamente portato alla realizzazione di un algoritmo di analisi di profondità densa a camera singola, che si distingue da altre soluzioni per essere facilmente portabile, leggero nelle risorse impiegate, libero da complesse fasi di set-up iniziale e, soprattutto, veloce nel suo compito.

La fase di test ha potuto contare sul supporto di beta-tester esterni al progetto, mentre lo sviluppo vero e proprio è ancora aperto ad ulteriori e progressivi miglioramenti, mirati all'applicazione stessa ma anche allo sviluppo di un futuro sistema di condivisione online delle "super-immagini" create dagli utenti, tramite il sito ufficiale (`http://www.superstereoapp.com`).

Bibliografia

[1] Roccetti M., Marfia G., Semeraro A., "Playing into the Wild: A Gesture-based Interface for Gaming in Public Spaces" (2012), Journal of Visual Communication and Image Representation, Elsevier, Vol. 23, n. 3, pp. 426-440

[2] Zhao W., Chellappa R., Rosenfeld A., Phillips P.J., "Face Recognition: A Literature Survey, ACM Computing Surveys" (2003) pp. 399-458

[3] Tian, Yingli, Takeo Kanade, and Jeffrey F. Cohn. "Facial expression recognition." (2011) Handbook of face recognition. Springer London. 487-519.

[4] Levinson, Jesse, et al. "Towards fully autonomous driving: Systems and algorithms." (2011) Intelligent Vehicles Symposium (IV), IEEE. IEEE, 2011.

[5] Van Krevelen, D. W. F., and Poelman R., "A survey of augmented reality technologies, applications and limitations." (2010) International Journal of Virtual Reality 9.2 : 1.

[6] Hubel D., "Eye, Brain, and Vision", Harvard University `http://hubel.med.harvard.edu/`

[7] CenturyTel Portal: `http://home.centurytel.net/s3dcor/Bates/Bates.htm`

[8] Horibuchi S., "Salvador Dalí: the stereo pair artist". (1994) San Francisco: Cadence Books. ISBN 0-929279-85-9

[9] Ukai K., "Human Factors for Stereoscopic Images", in IEEE ICME, pp. 1697-1700, July 2006.

[10] RealD, the new 3D `http://www.reald.com`

[11] Hannah M. J., "Computer Matching of Areas in Stereo Images." (1974), Ph.D. thesis, Stanford University.

[12] Fakhar K., "Introduction to Remote Sensing and Photogrammetry", (2013) Dep. of Pharmaceutical, Chemistry & Environmental Sciences

[13] Marr D., "Vision: A Computational Investigation into the Human Representation and Processing of Visual Information." (1982) W. H. Freeman, San Francisco.

[14] Moravec H., "The Stanford cart and the CMU rover". (1983) Proceedings of the IEEE, 71(7):872 - 884.

[15] Konolige, K., "Small vision systems: Hardware and implementation." (1997) In Eighth International Symposium on Robotics Research.

[16] Antunes M. et al., "Can stereo vision replace a Laser Rangefinder?." (2012) Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on. IEEE, 2012.

[17] Eren H. et al., "Stereo vision and statistical based behavior prediction of driver", (2007) Proc. IEEE Intell. Veh. Symp., pp. 657 - 662

[18] Yang R. et al., "Stereo vision on GPU." (2006) Workshop on Edge Computing Using New Commodity Architectures (University of North Carolina).

[19] Mattoccia S. et al., "An embedded 3D camera on FPGA", (2013) Multimedia Communications Technical Committee, IEEE Communication Society, E-Letter, Vol 8, No. 3.

[20] Zanela A. et al., "A cellular neural network stereo vision system for autonomous robot navigation." (2000) Cellular Neural Networks and Their Applications. Proceedings of the 2000 6th IEEE International Workshop on. IEEE, 2000.

[21] Goldberg S. B., et al. "Stereo vision and rover navigation software for planetary exploration." (2002) Aerospace Conference Proceedings. IEEE. Vol. 5. IEEE, 2002.

[22] Multi-View Stereo, University of Middlebury `http://vision.middlebury.edu/mview/data/`

[23] El-Alfy H. et al., "Recent Developments in Video Surveillance" (2012) Publisher: InTech, ISBN 9789535104681.

[24] Kanade, T. et al., "A stereo machine for video-rate dense depth mapping and its new applications." (1996) In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 96), pp. 196202, San Francisco.

[25] Instagram Press `http://instagram.com/press/`

[26] Wikipedia - List of 3D-enabled mobile phones `http://en.wikipedia.org/wiki/List_of_3D-enabled_mobile_phones` (Dicembre 2013)

[27] GSM Area: HTC Evo 3D `http://www.gsmarena.com/htc_evo_3d-3901.php`

[28] Poppy - Turn your iPhone into a 3D camera `http://poppy3d.com/`

[29] Takashi S., "The zone of comfort: Predicting visual discomfort with stereo displays". (2011) Journal of Vision July 21, 2011 vol. 11 no. 8 article 11

[30] Holzmann C., "Measuring Distance with Mobile Phones Using Single-Camera Stereo Vision." (2012), Distributed Computing Systems Workshops (ICDCSW), 32nd International Conference.

[31] Engineering Human-Computer Interaction Research Group - 3D displays on mobile device `http://iihm.imag.fr/en/demo/hcpmobile/`

[32] Hartley R. I., "Theory and practice of projective rectification". (1999) International Journal of Computer Vision, 35(2), 115-127.

[33] Szeliski R. "Image Alignment and Stitching". (2005)

[34] Fusiello, A., Irsara L., "Quasi-Euclidean epipolar rectification of uncalibrated images." (2011) Machine Vision and Applications, 22(4), 663-670.

[35] Rosten E., Drummond T., "Machine learning for high-speed corner detection." (2006) Computer Vision ECCV. Springer Berlin Heidelberg. 430-443.

[36] Leutenegger S. et al., "BRISK: Binary robust invariant scalable keypoints." Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[37] Rublee E. et al., "ORB: an efficient alternative to SIFT or SURF." Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

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

[39] Klaus A. et al., "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure", (2006) Int. Conf. Pattern Recognition.

[40] Bobick A. F. and Intille S. "Large occlusion stereo." (1999) International Journal of Computer Vision 33.3 : 181-200.

[41] Kolmogorov V. and Zabih R., "Computing visual correspondence with occlusions using graph cuts" (2001) ICCV.

[42] Wang Z. and Zheng Z., "A region based stereo matching algorithm using cooperative optimization". (2008)CVPR.

[43] Shi C. et al., "High-accuracy stereo matching based on adaptive ground control points". (2012) Submitted to IEEE TIP 2012

[44] Grimson W. E. L., "Computational experiments with a feature based stereo algorithm" (1985) IEEE TPAMI, 7(1) 17-34.

[45] Di Stefano L., Mattoccia S., "Real-time stereo within the VIDET project Real-Time Imaging", (2002)

[46] Zabih R. and Woodfill J., "Non-parametric local transforms for computing visual correspondence". (1994) In Proceedings of the third European conference on Computer Vision (ECCV). Secaucus, NJ, USA: Springer-Verlag New York,Inc., pp. 151-158.

[47] Froba B. and Ernst A., "Face detection with the modified census transform." (2004) Automatic Face and Gesture Recognition. Proceedings. Sixth IEEE International Conference on. IEEE.

[48] Gonzales, Rafael C., and R. E. Woods. "Digital image processing" (1993)

[49] Pham, Dzung L., Chenyang Xu, and Jerry L. Prince. "Current methods in medical image segmentation 1." (2000) Annual review of biomedical engineering 2.1 : 315-337.

[50] Sun, Deqing, Erik B. Sudderth, and Michael J. Black. "Layered segmentation and optical flow estimation over time." (2012) Computer Vision and Pattern Recognition (CVPR), IEEE Conference.

[51] Felzenszwalb, P., Huttenlocher, D.: "Effcient graph-based image segmentation."(2004) IJCV 167 181

[52] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk, "SLIC Superpixels" (2010), EPFL Technical Report 149300, June.

[53] C. Shi, G. Wang, X. Pei, H. Bei, and X. Lin. "High-accuracy stereo matching based on adaptive ground control points". (2012) Submitted to IEEE TIP

[54] H. Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutual information". (2005) CVPR, PAMI 30(2):328-341.

[55] Zureiki, Ayman, Michel Devy, and Raja Chatila, "Stereo Matching and Graph Cuts". Stereo Vision: 349-372.

[56] A.F. Bobick and S.S. Intille, "Large occlusion stereo" (1999) International Journal of Computer Vision, vol. 33, pp. 181-200.

[57] Boykov, Veksler, and Zabih, "Graph cuts using alpha-beta swaps" (2001) PAMI

[58] Microsoft - Kinect for Windows `http://www.microsoft.com/en-us/kinectforwindows/`

[59] S. M. Seitz and C. R. Dyer, "Physically-Valid View Synthesis by Image Interpolation " (1995) Proc. Workshop on Representation of Visual Scenes, 18-25.

[60] MSDN - Concepts and architecture for Windows Phone: `http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff967549.aspx`

[61] MSDN - The unchecked keyword `http://msdn.microsoft.com/it-it/library/a569z7k8.aspx`

[62] Android Developers - Android NDK `https://developer.android.com/tools/sdk/ndk/index.html`

[63] Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." (1981) Communications of the ACM 24.6: 381-395.

Ringraziamenti

La stesura di questo elaborato ha coinciso con il rilascio della prima versione del software, un passo importante che non si sarebbe concretizzato senza un gruppo di lavoro compatto e preciso nello svolgimento dei vari impegni. E' per questo che, prima di tutto, voglio personalmente ringraziare i miei amici (poi colleghi) Gian Piero Limone e Luca Milioli, i quali hanno preso parte attiva fin dalla nascita di questo progetto, discutendone le funzionalità ed introducendone aspetti chiave.

Progetto che, essendo nato dapprima come elaborato d'esame, non avrebbe avuto modo di esistere in questa forma senza l'approvazione ed il supporto del Professor Marco Roccetti, che ringrazio sentitamente. I ringraziamenti vanno anche a due docenti con i quali non ho avuto modo di avere un rapporto diretto e costante, poichè appartenenti a facoltà diverse: il Professor Andrea Fusiello dell'Università di Udine, per le delucidazioni circa il suo stesso algoritmo di rettificazione, ed il Professor Stefano Mattoccia della Facoltà di Ingegneria di Bologna, per avermi fornito materiale fondamentale per lo studio di tecniche di stereo matching. A tutti faccio i più sentiti auguri per un proseguimento di carriera pieno di successi.

Assieme al gruppo di lavoro di SuperStereo non posso, inoltre, ignorare i consigli che i beta tester (amici, colleghi e parenti) hanno dato durante le ultime fasi della scrittura dell'interfaccia, a volte anche in maniera inconsapevole. Sono stati fondamentali e sono sicuro che verranno considerati ancora per eventuali, nuove, fasi di sviluppo. Il futuro di un'idea, anche la più semplice, può difatti apparire incerto e gli impegni che gravano su ogni

componente del team ne possono ostacolare la realizzazione. In ogni caso, non occorre necessariamente seguire un percorso prefissato ma, anzi, la bellezza sta nell'imprevedibilità. Come disse un famoso scienziato (che se fosse veramente esistito avrebbe sicuramente fatto parlare di sé):

"Strade? Dove stiamo andando non c'è bisogno... di strade!"