Università degli Studi di Padova
Facoltà di Ingegneria

Finito di scrivere il giorno 5 dicembre 2013 utilizzando LaTeX 2ε.

UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA

—

Dipartimento di Ingegneria dell'Informazione

—

LAUREA MAGISTRALE IN INGEGNERIA
DELL'AUTOMAZIONE

Analisi ed Elaborazione di Immagini Tridimensionali Acquisite da Microsoft Kinect

Relatore: Prof. Enrico Pagello
Co-Relatore: Ing. Walter Zanette

Laureando: MICHELE SAVIETTO

ANNO ACCADEMICO 2013-2014

A Roberta...

Sommario

Questa tesi tratta un aspetto particolare dell'elaborazione delle immagini ovvero la ricostruzione tridimensionale. Questa tematica non ha diretta applicazione nell'industria ma, se combinata con altri elementi, può rivelarsi uno strumento valido ed innovativo. L'intento è infatti quello di sviluppare questo progetto per poterlo applicare a celle di lavoro automatiche. I dati ricavati dalla ricostruzione tridimensionale, adeguatamente elaborati, vengono utilizzati per controllare i movimenti di uno o più robot. Le attività che si potrebbero svolgere sono molteplici: assemblaggio, bin-picking, pallettizzazione, packaging, solo per citarne alcune.

L'elemento innovativo sta nell'usare fotocamere di bassa qualità (relativamente alle applicazioni di elaborazione di immagini) per creare un modello della scena il più preciso possibile. Ciò è ottenuto combinando i dati acquisiti dai singoli dispositivi compensando le loro scarse caratteristiche. In questo modo è possibile ridurre i costi di realizzazione dell'impianto.

Questo progetto è l'evoluzione di un simulatore robot 3D fornito dall'azienda Euclid Labs. Al termine del lavoro sarà possibile eseguire le scansioni e visualizzare la scena ricostruita direttamente nel simulatore.

I contributi innovativi di questa tesi stanno nel processo di elaborazione dei dati acquisiti e nella modalità di calibrazione delle fotocamere. Nel primo caso, per quanto è stato testato, si sono ottenuti risultati migliori rispetto ai metodi usati nei precedenti progetti da Euclid Labs; nel secondo è stata proposta una procedura nuova e differente dai metodi standard presenti in letteratura.

Ringraziamenti

Desidero innanzitutto ringraziare Matteo Peluso e Roberto Polesel per avermi dato la possibilità di svolgere questa attività e di avermi fatto avvicinare al mondo del lavoro mettendomi di fronte a problematiche che non si possono comprendere all'interno del solo ambiente universitario. Ringrazio tutti i componenti dell'azienda Euclid Labs per la disponibilità e per l'aiuto che mi hanno dato, in particolare Walter Zanette che mi ha seguito passo passo nello sviluppo del progetto.

Ringrazio poi il Professor Enrico Pagello per aver accettato l'incarico di relatore per la mia tesi.

Infine, ci tengo a ringraziare sentitamente i miei genitori, mia sorella e Roberta per aver creduto in me e per avermi sempre sostenuto ed incoraggiato durante tutto il percorso di studi intrapreso.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Questa tesi, intitolata "*Analisi ed Elaborazione di Immagini Tridimensionali Acquisite da Microsoft Kinect*", nasce dall'idea di portare un'innovazione nel mondo della programmazione della robotica industriale. Il campo in cui si inserisce il lavoro svolto è quello della computer vision e riguarda in particolare la ricostruzione tridimensionale. Tale tecnica consiste nel combinare le informazioni provenienti da diversi dispositivi per produrre un'interpretazione in tre dimensioni della scena inquadrata. É possibile visualizzare il risultato ottenuto in un ambiente grafico 3D dove è possibile manipolarlo al fine di adattarlo ad altre operazioni.

I passaggi fondamentali di questo processo sono due: il trattamento dei dati acquisiti per adattarli al meglio alle fasi successive e la procedura di calibrazione delle fotocamere. Tanto migliori sono queste fasi, tanto più preciso sarà il risultato finale. Questo fatto si rivela di fondamentale importanza in quanto i dispositivi di acquisizione utilizzati non sono progettati per questi scopi. Le fotocamere a disposizione per questo sviluppo sono le Microsoft Kinect, prodotte da Microsoft per la consolle XBox. Come si può intuire, la risoluzione necessaria per i due campi è decisamente diversa. Questo è il fulcro del progetto: cercare di realizzare un sistema con dispositivi a prestazioni ridotte che sia applicabile in ambito industriale. L'utilizzo di periferiche con scarse caratteristiche consente di contenere i costi di realizzazione e di conseguenza quelli di vendita.

L'intero progetto su cui si basa questa tesi è stato realizzato in collaborazione con l'azienda Euclid Labs. Questa è un'azienda leader nel settore della robotica, non solo a livello nazionale ma anche europeo. Euclid Labs opera in diversi campi della virtualizzazione e del CAD/CAM. Per quanto riguarda la sezione inerente al progetto, l'azienda ha sviluppato un proprio sistema di virtualizzazione degli impianti robot. Questo rappresenta uno dei

punti di forza di Euclid Labs in quanto permette di programmare i sistemi robot off-line e di effettuare tutti i test desiderati all'interno del simulatore.

I contributi apportati da questa tesi riguardano il filtraggio dei dati acquisiti da Microsoft Kinect e la procedura di calibrazione delle fotocamere. Per quanto riguarda il primo fatto, si sono ottenuti risultati migliori rispetto al metodo di filtraggio che Euclid Labs ha sfruttato in altri progetti. Per il secondo contributo invece è stata implementata una procedura rispetto ai metodi di calibrazione standard che porta a risultati finali decisamente più precisi.

Nel Capitolo 2 viene presentata l'azienda Euclid Labs. Il resto della tesi è strutturata in modo tale da fornire le conoscenze teoriche necessarie per comprendere le varie fasi di sviluppo del progetto, Capitolo 3. Vengono richiamati i concetti fondamentali di visione e di elaborazione delle immagini, di cui fanno parte algoritmi quali la trasformata di Hough e l'edge detector di Canny. Nel Capitolo 4 è illustrata la struttura di Microsoft Kinect, mentre nel Capitolo 5 viene esposto com'è stato sviluppato il progetto in ogni sua parte.

Capitolo 2

L'azienda Euclid Labs

Euclid Labs è l'azienda presso cui è stato sviluppato questo progetto, in Figura 2.1 è riportato il logo aziendale. Nata nel 2005, Euclid Labs sviluppa soluzioni per la robotica e l'automazione industriale, in particolare sistemi di programmazione automatica ed off-line, soluzioni CAD/CAM, simulazione tridimensionale, sistemi di visione e supervisione. Oltre a questi prodotti "standard", ricerca soluzioni personalizzate per problemi riguardanti la produzione robotizzata seguendo ogni passo dello sviluppo: dalla progettazione al collaudo finale dell'impianto.

Figura 2.1: Logo dell'azienda Euclid Labs s.r.l.

Attraverso l'uso di moderni strumenti software e l'analisi accurata del processo produttivo del cliente, l'azienda si prefigge di introdurre la robotica in nuovi mercati come la produzione di piccoli lotti o l'esecuzione di compiti complessi.
Euclid Labs è un'azienda leader nel settore della robotica non solo a livello nazionale ma anche europeo ed inoltre vanta collaborazioni con Asia e Stati Uniti. Gli anni di esperienza nello sviluppo di nuovi prototipi, di celle di lavoro innovative e di soluzioni ad hoc, unita a collaborazioni stabili con

altre aziende, hanno permesso lo studio e la possibilità di offrire ai clienti soluzioni sempre migliori. Alcuni tipici esempi di ambiti in cui Euclid Labs opera sono l'assemblaggio, le operazioni di carico/scarico, l'imballaggio, la pallettizzazione, il pick & place, ed altre ancora. Entrando un po' più nello specifico, le capacità di Euclid Labs si possono riassumere nei seguenti punti:

Dal disegno al prodotto Euclid Labs propone soluzioni in cui si ottiene il prodotto finito a partire dall'elaborazione di un modello CAD. Questo tipo di tecnologia trova applicazione in svariati campi, alcuni dei quali sono riportati di seguito:

- taglio di lamiere piane;
- sbavatura e finitura di bordi;
- incisione su superfici tridimensionali;
- incollaggio;
- cablaggio automatizzato;
- taglio di tubi, anche di grande diametro (oltre 1500mm).

Soluzioni CAD/CAM Euclid Labs fornisce alle produzioni automatizzate un'ampia esperienza nello sviluppo di sistemi CAD/CAM personalizzati. I programmi iniziano da un disegno 2D (tipicamente un DXF) o 3D (per esempio un STL) e, in maniera completamente automatica, oppure tramite una procedura interattiva, vengono generate soluzioni per i differenti tipi di controllo da attuare. Parte del software CAM sviluppato da Euclid Labs è applicato a macchine per la foratura o la molatura di lenti, a robot usati nello scarico e smistamento di pezzi tagliati da laser, a sistemi di lucidatura per marmo, al taglio di pelli con getto d'acqua, all'ispezione automatica tridimensionale.

Sistemi di carico/scarico Euclid Labs offre soluzioni avanzate per il carico/scarico in grado di soddisfare anche vincoli stringenti in termini di velocità o di potenza. Attualmente si riescono a spostare fino a 800Kg con un singolo robot, ma anche pesi maggiori usando due o più robot cooperanti. Questi sistemi sono collocabili su slitte per aumentare la flessibilità dell'impianto. I dati per lo scarico possono essere acquisiti da sensori, sistemi di visione o da altro software esterno.

Servizi per costruttori di macchine ed impianti Euclid Labs mette la propria esperienza a disposizione dei costruttori di macchine ed impianti speciali, aggiungendo al loro know-how le potenzialità della robotica e dei moderni sistemi di programmazione off-line. I servizi spaziano

dalla completa integrazione di robot con altre macchine, alla programmazione di sistemi CAD/CAM specifici per le applicazioni richieste. Le tecnologie dell'azienda trovano applicazione in svariati settori: dal metallo al legno, dal tessile all'oreficeria, dal calzaturiero alle lenti per occhiali.

2.1 Ambiente di sviluppo 3D

Euclid Labs ha sviluppato un proprio ambiente di virtualizzazione tridimensionale in cui si possono inserire modelli di robot, di macchinari quali ad esempio presse, di pezzi da lavorare, ecc. Con questo software Euclid Labs riesce a riprodurre le celle di lavoro che dovranno poi essere installate negli impianti reali.

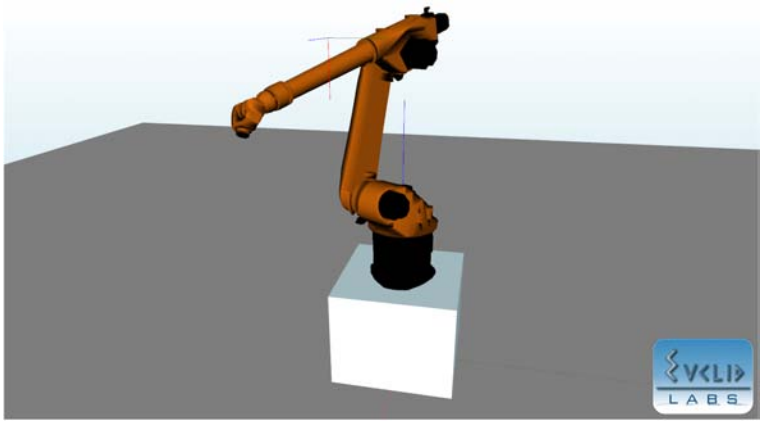


Figura 2.2: Simulatore 3D sviluppato da Euclid Labs

Di fondamentale importanza è la possibilità di eseguire simulazioni all'interno di questo ambiente. Inoltre, nel corso della sua attività, l'azienda ha creato dei moduli appositi per la risoluzione di problemi specifici. In altri termini, il cuore delle applicazioni sviluppate da Euclid Labs è il simulatore 3D in cui, a seconda degli obiettivi da raggiungere in ogni progetto, vengono inseriti i pacchetti specifici. Questa combinazione rende possibile la programmazione off-line riducendo così i tempi di fermo-macchina dovuti all'installazione del software. Inoltre, grazie all'introduzione di questo strumento, non è più necessario recarsi dal cliente per testare il software, assicurando così all'azienda un risparmio in termini di tempo e di costi. Non meno importante è la possibilità di ricostruire un'immagine tridimensionale, di identificare

la posizione di un oggetto, di generare traiettorie per il movimento del robot, di evitare collisioni fra robot ed altri macchinari oppure oggetti presenti nell'area di lavoro, ecc. La correttezza dell'intero processo è poi verificabile tramite simulazione.

Capitolo 3

Background teorico

In questa sezione vengono richiamati i concetti fondamentali di visione tridimensionale ed elaborazione delle immagini utilizzati per portare a termine il progetto. Questo al fine di rendere comprensibile la descrizione in ogni sua parte.

3.1 Visione 3D

Fra tutte le abilità sensoriali la visione è largamente riconosciuta come quella con le maggiori potenzialità. Le capacità dei sistemi biologici sono formidabili: l'occhio raccoglie una banda di radiazioni elettromagnetiche riflesse da diverse superfici e provenienti da fonti luminose diverse; il cervello elabora questa informazione formando il quadro della scena. Se si volesse dare una definizione, si potrebbe dire che la *Visione Computazionale*, o *Computer Vision*, si occupa dell'analisi di immagini con il calcolatore. L'analisi è finalizzata a scoprire cosa è presente nella scena e dove. La visione computazionale non tenta di replicare la visione umana anche perché il tentativo risulterebbe fallimentare a causa dell'intrinseca differenza tra i due hardware.

Ci sono diversi aspetti secondo cui la visione tridimensionale che considera immagini di intensità[1], è ritenuta difficile:

- il sistema di visione di una macchina e l'occhio di un essere umano compiono una proiezione prospettica della scena, ciò comporta perdita di informazione. Tutti i punti che giacciono su una linea che va dal centro ottico verso un punto sulla scena, sono proiettati in un unico punto sull'immagine;

[1]Immagine il cui contenuto informativo riguarda lo spettro luminoso della scena inquadrata dal dispositivo di ripresa.

- la relazione fra l'intensità dell'immagine e la geometria 3D del corrispondente punto della scena è molto difficile. L'intensità dei pixel dipende dal tipo di superficie e dalla sua orientazione, dal tipo e dalla posizione dell'illuminazione e dalla posizione dell'osservatore;
- la mutua occlusione di oggetti nella scena;
- la presenza di rumore nell'immagine.

3.1.1 Basi di geometria proiettiva

La visione computazionale ha avuto un rapido sviluppo e maturazione per quanto riguarda la geometria della scena da più viste. L'utilizzo di strumenti matematici ha permesso relazioni fra

- punti 3D nella scena (e più in generale linee e altri oggetti geometrici simili);
- la loro proiezione sulla fotocamera;
- le relazioni fra le proiezioni della scena 3D.

Il sensore più semplice che fornisce informazioni circa il mondo tridimensionale circostante è una fotocamera. Un'immagine bidimensionale viene creata raccogliendo la luce riflessa dagli oggetti della scena. Il processo di formazione dell'immagine a partire da una fotocamera stenopeica, o con lenti sottili, si basa sulla **proiezione prospettica**. Per maggiori dettagli su pinhole camera e lenti sottili vedere [7] e [13]. Linee parallele nel mondo non rimangono tali quando riportate in un'immagine a causa della prospettiva, come ad esempio la visuale di un binario riportata in Figura 3.1.

Si consideri uno spazio lineare di dimensione $d+1$ privato dell'origine, $\mathbb{R}^{d+1} - \{[0,\ldots,0]^T\}$, e si definisca una relazione di equivalenza

$$\begin{aligned} [x_1,\ldots,x_{d+1}]^T &\simeq [x'_1,\ldots,x'_{d+1}]^T \\ \text{se e solo se } \exists\alpha \neq 0 : [x_1,\ldots,x_{d+1}]^T &= \alpha[x'_1,\ldots,x'_{d+1}]^T \end{aligned} \tag{3.1}$$

Ciò significa che due vettori in $\mathbb{R}^{d+1}$ sono equivalenti se sono uguali a meno di un fattore di scala. Lo **spazio proiettivo** $\mathbb{P}^d$ è lo spazio quoziente di questa relazione di equivalenza. Può essere pensato come l'insieme di tutte le linee in $\mathbb{R}^{d+1}$ passanti per l'origine.

Un punto in $\mathbb{P}^d$ corrisponde ad un insieme infinito di vettori paralleli in $\mathbb{R}^{d+1}$ ed è espresso in **coordinate omogenee**. Tali vettori prendono il

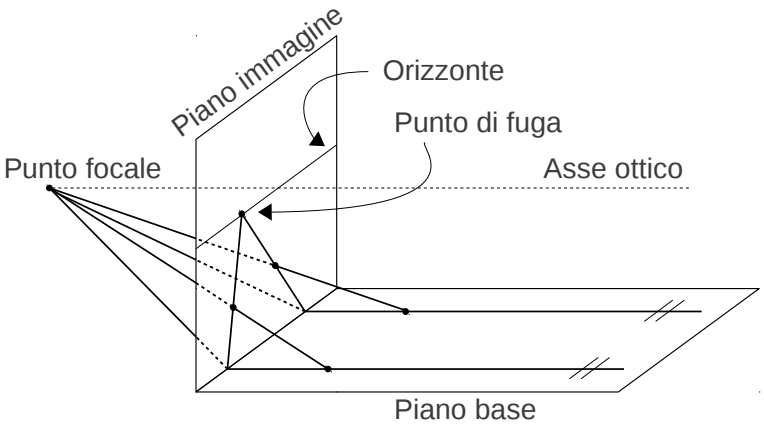


Figura 3.1: Proiezione prospettica di linee parallele

nome di rappresentazione omogenea del punto in $\mathbb{P}^d$. Un vettore omogeneo $\mathbf{v}$ rappresenta lo stesso punto di tutti i vettori che differiscono da $\mathbf{v}$ solo per un fattore di scala. Solitamente il fattore di scala è scelto in modo tale che l'ultima componente del vettore sia pari ad 1, e.g. $[x'_1, \ldots, x'_d, 1]^T$.

In genere si è abituati ad esprimere i punti in coordinate cartesiane (chiamate anche coordinate non omogenee). Queste sono coordinate di punti nello spazio euclideo n-dimensionale $\mathbb{R}^n$ che occupano il piano di equazione $x_{d+1} = 1$ in $\mathbb{R}^{d+1}$. La mappa da vettori non-omogenei in $\mathbb{R}^d$ a $\mathbb{P}^d$ è data da

$$[x_1, \ldots, x_d]^T \rightarrow [x_1, \ldots, x_d, 1]^T \tag{3.2}$$

I punti $[x_1, \ldots, x_d, 0]^T$ non hanno una corrispondente rappresentazione in coordinate cartesiane ed esprimono i punti all'infinito in una specifica direzione. Si consideri $[x_1, \ldots, x_d, 0]^T$ come un caso limite di $[x_1, \ldots, x_d, \alpha]^T$, che è equivalente a $[x_1/\alpha, \ldots, x_d/\alpha, 1]^T$, e si assuma che $\alpha \rightarrow 0$. Questo corrisponde ad un punto in $\mathbb{R}^d$ che va all'infinito nella direzione del raggio dato da $[x_1/\alpha, \ldots, x_d/\alpha]^T \in \mathbb{R}^d$.

Un iperpiano in $\mathbb{P}^d$ è rappresentato da un vettore $\mathbf{a} = [a_1, \ldots, a_{d+1}]^T$, di dimensioni $d+1$, tale che tutti i punti $\mathbf{x}$ che giacciono sull'iperpiano soddisfino $\mathbf{a}^T\mathbf{x} = 0$. Considerando i punti nella forma $\mathbf{x} = [x_1, \ldots, x_d, 1]^T$ si ottiene la formula $a_1x_1 + \cdots + a_dx_d + a_{d+1} = 0$. Segue dunque che l'iperpiano definito da d punti distinti, rappresentati da vettori $\mathbf{x}_1, \ldots, \mathbf{x}_d$ che giacciono su di esso, è dato da un vettore $\mathbf{a}$ ortogonale ai vettori $\mathbf{x}_1, \ldots, \mathbf{x}_d$. Simmetricamente il punto di intersezione di d iperpiani distinti $\mathbf{a}_1, \ldots, \mathbf{a}_d$ è il vettore $\mathbf{x}$ a loro ortogonale.

Nello specifico, sono due i casi di interesse per la visione computazionale:

1. il **piano proiettivo** $\mathbb{P}^2$, dove i punti saranno denotati come $\mathbf{u} = [u, v, w]^T$;

2. lo **spazio proiettivo** $\mathbb{P}^3$, dove i punti saranno denotati come $\mathbf{X} = [X, Y, Z, W]^T$;

3.1.2 Modello di una fotocamera

In questo paragrafo vengono richiamati alcuni concetti legati al modello geometrico della formazione dell'immagine, ovvero come sono collegate la posizione di un punto nella scena e la posizione del punto corrispondente nell'immagine. Il più semplice modello geometrico di fotocamera fa ricorso alla **fotocamera stenopeica** (o **pinhole camera**), che si adatta a molte applicazioni di computer vision. La geometria del modello è riportata in Figura 3.2. Il piano in basso è il **piano immagine** π dove il mondo reale, visto della telecamera, viene proiettato. La linea verticale tratteggiata è l'**asse ottico**. La lente è posizionata perpendicolarmente all'asse ottico nel **punto focale C** (chiamato anche **centro ottico** o **centro di proiezione**). Il **punto principale** $\mathbf{U}_0$, spesso chiamato centro dell'immagine nella procedura di calibrazione di una fotocamera, è l'intersezione dell'asse ottico con il piano immagine π. Il punto principale $\mathbf{U}_0$ espresso nel sistema di coordinate affine dell'immagine risulta $\mathbf{U}_{0a} = [u_0, v_0, 0]^T$. La **lunghezza focale** f rappresenta la distanza fra il punto focale e il punto principale, ed è un parametro caratteristico della lente.

Una fotocamera esegue una trasformazione lineare dallo spazio proiettivo tridimensionale $\mathbb{P}^3$ allo spazio proiettivo $\mathbb{P}^2$. La proiezione è dovuta alla riflessione di un raggio ottico da un punto $\mathbf{X}$ della scena, in alto a sinistra in Figura 3.2. Il raggio ottico, dopo aver attraversato il centro ottico $\mathbf{C}$, intercetta il piano immagine nel punto $\mathbf{U}$.

Vengono definiti ora i sistemi di riferimento adottati.

1. **Sistema di coordinate mondo euclideo** (pedice $_w$): ha l'origine nel punto $\mathbf{O}_w$. I punti $\mathbf{X}$ e $\mathbf{U}$ sono espressi in coordinate mondo.

2. **Sistema di coordinate della fotocamera euclideo** (pedice $_c$): ha il punto focale $\mathbf{C} \equiv \mathbf{O}_c$ come origine. L'asse Z_c è allineato con l'asse ottico e punta nella direzione opposta al piano immagine.

   C'è un'unica relazione fra le coordinate mondo e quelle della telecamera. Si possono far coincidere i due sistemi di riferimento eseguendo una trasformazione euclidea che consiste in una traslazione $\mathbf{t}$ e in una rotazione R.

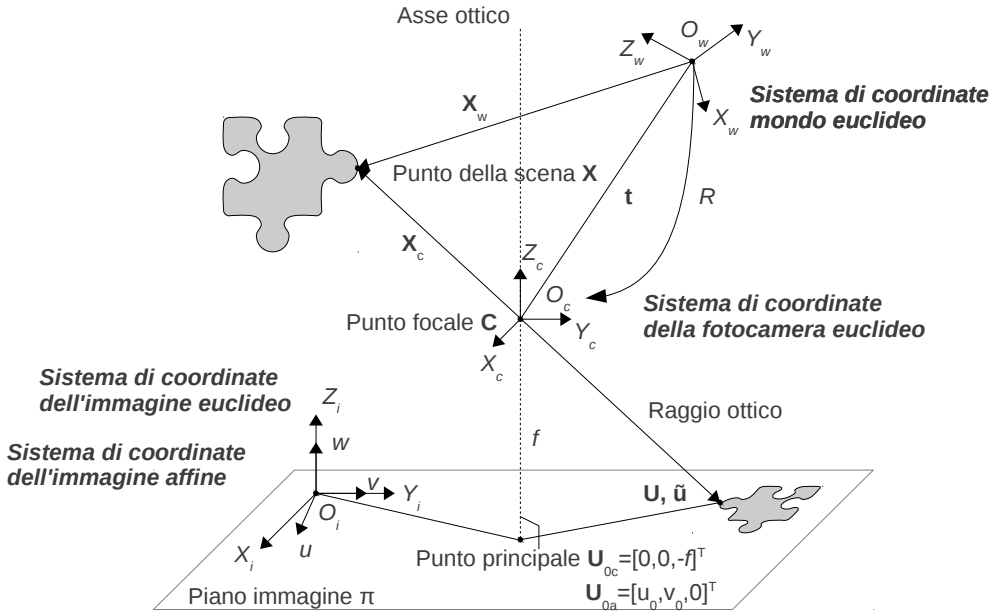


Figura 3.2: Geometria della proiezione di un oggetto nel mondo

3. **Sistema di coordinate dell'immagine euclideo** (pedice $_i$): ha gli assi allineati con quelli del sistema di riferimento della fotocamera, con X_i e Y_i che giacciono sul piano immagine.

4. **Sistema di coordinate dell'immagine affine** (pedice $_a$): ha l'origine coincidente con l'origine del sistema di coordinate dell'immagine euclideo $\mathbf{O}_i$. Gli assi v e w sono allineati con gli assi Y_i e Z_i, ma l'asse u può avere orientazione differente rispetto a X_i.

   La ragione per cui vengono introdotte queste coordinate risiede nel fatto che la matrice di pixel può non essere allineata con l'immagine, inoltre gli assi possono essere scalati in modo differente.

Nel caso generale la trasformazione proiettiva può essere suddivisa in tre trasformazioni più semplici, che corrispondono alle tre transizioni fra i quattro tipi di sistemi di coordinate appena descritti.

La *prima trasformazione* (tra 1 e 2) costituisce il passaggio dalle (arbitrarie) coordinate mondo ($\mathbf{O}_w$, X_w, Y_w, Z_w) alle coordinate della fotocamera ($\mathbf{O}_c$, X_c, Y_c, Z_c). Il sistema di coordinate mondo può essere allineato con quello della fotocamera traslando l'origine $\mathbf{O}_w$ su $\mathbf{O}_c$ con il vettore $\mathbf{t}$, e ruotando poi gli assi con la matrice di rotazione R. La trasformazione di un

punto $\mathbf{X}_w$ nel punto $\mathbf{X}_c$ espressa in coordinate euclidee risulta

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R\big(\mathbf{X}_w - \mathbf{t}\big) \tag{3.3}$$

La matrice di rotazione R esprime tre rotazioni elementari: rotazioni attorno agli assi X_w, Y_w e Z_w. Il vettore traslazione $\mathbf{t}$ dà invece tre elementi di traslazione: traslazioni lungo gli assi X_w, Y_w e Z_w dell'origine delle coordinate mondo rispetto alle coordinate della fotocamera. Dunque ci sono sei parametri: tre rotazioni e tre traslazioni, contenuti rispettivamente in R e $\mathbf{t}$ che sono denominati **parametri estrinseci** della fotocamera.

L'Equazione 3.3 espressa in coordinate omogenee diventa

$$\mathbf{X}_c = \begin{bmatrix} R & -R\mathbf{t} \\ 0^T & 1 \end{bmatrix} \mathbf{X}_w \tag{3.4}$$

La *seconda trasformazione* (fra 2 e 3) proietta il punto $\mathbf{X}_c$ della scena 3D, espresso nelle coordinate della fotocamera ($\mathbf{O}_c$, X_c, Y_c, Z_c), nel punto $\mathbf{U}$ nel piano immagine π espresso nelle coordinate dell'immagine ($\mathbf{O}_i$, X_i, Y_i, Z_i). Le coordinate del punto $\mathbf{U}$, in seguito alla proiezione $\mathbb{R}^3 \rightarrow \mathbb{R}^2$, si ricavano sfruttando le proprietà di similarità dei triangoli, Figura 3.3. Si ottengono

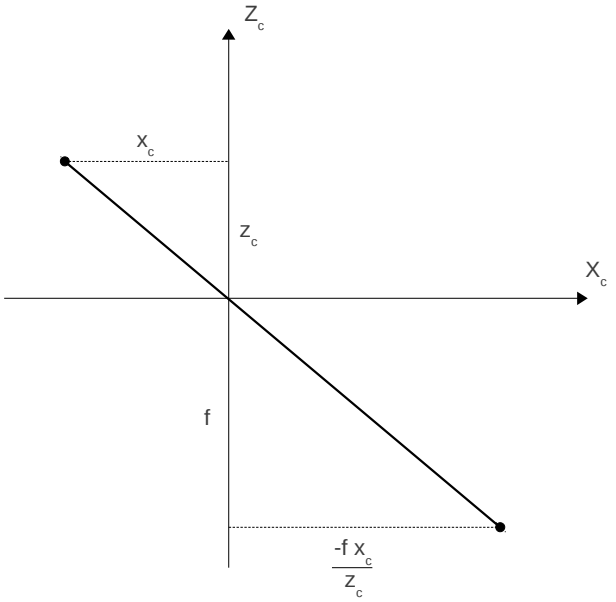


Figura 3.3: Calcolo delle coordinate del punto proiettato

così due equazioni non lineari in Z_c

$$X_i = \frac{-fX_c}{Z_c} \qquad Y_i = \frac{-fY_c}{Z_c} \tag{3.5}$$

dove f è la lunghezza focale. Ricorrendo invece alle coordinate omogenee la trasformazione si intende fra spazi proiettivi, $\mathbb{P}^3 \to \mathbb{P}^2$. In questo modo la relazione diventa lineare. Siano dunque

$$\mathbf{U} = \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \qquad \text{e} \qquad \mathbf{X}_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{3.6}$$

le coordinate omogenee di $\mathbf{U}$ ed $\mathbf{X}_c$ rispettivamente. Si noti che, ponendo l'ultima componente ad 1, si escludono i punti all'infinito (per includerli si dovrebbe usare una componente generica). Dunque l'equazione di proiezione prospettica, in questo caso semplificato, si riscrive

$$Z_c \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} = \begin{bmatrix} -fX_c \\ -fY_c \\ Z_c \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{3.7}$$

Passando alla notazione matriciale

$$Z_c \mathbf{U} = P \mathbf{X}_c \tag{3.8}$$

oppure ancora

$$\mathbf{U} \simeq P \mathbf{X}_c \tag{3.9}$$

dove il simbolo $\simeq$ sta ad indicare che l'uguaglianza è valida a meno di un fattore di scala e P vale

$$P = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.10}$$

Quando si verifica il caso speciale (ed ideale) in cui il piano immagine si trova davanti al centro di proiezione e la focale è unitaria $f = -1$ (chiamata fotocamera con piano immagine normalizzato, Figura 3.4), si ha

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} I | \mathbf{0} \end{bmatrix} \tag{3.11}$$

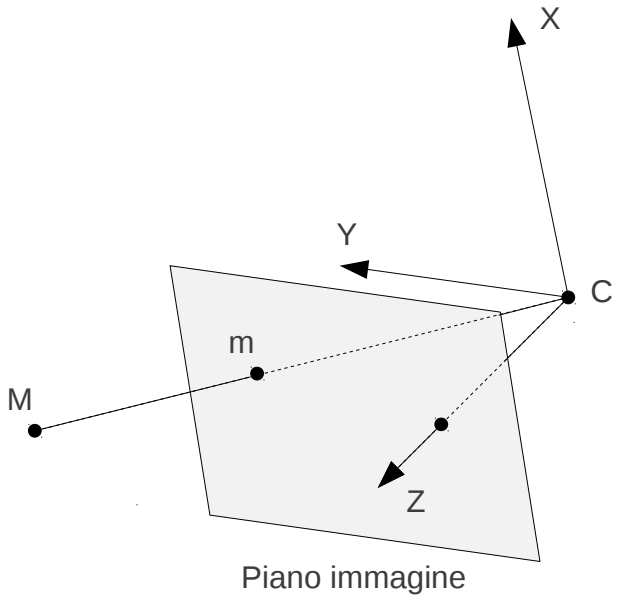


Figura 3.4: Proiezione prospettica con piano immagine davanti al centro di proiezione

La *terza trasformazione* (fra 3 e 4) mappa il sistema di coordinate dell'immagine euclideo in quello affine. In questa trasformazione si presentano dei coefficienti, detti **parametri intrinseci**, che descrivono le caratteristiche fisiche della specifica fotocamera indipendentemente dalla sua posizione ed orientazione nello spazio. É conveniente raggruppare i parametri intrinseci, la lunghezza focale f è uno di questi, in una matrice K di dimensioni 3×3 detta **matrice di calibrazione dei parametri intrinseci**.

$$\tilde{\mathbf{u}} \simeq K\mathbf{U} = \begin{bmatrix} f & s & -u_0 \\ 0 & g & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{U} \tag{3.12}$$

I parametri intrinseci sono i seguenti: f dà la riscalatura lungo l'asse u mentre g dà la riscalatura lungo l'asse v. Spesso entrambi i valori sono uguali alla lunghezza focale, $f = g$. Il parametro s invece fornisce il grado di distorsione degli assi nel piano immagine. Si è assunto che l'asse v del sistema di coordinate immagine affine coincida con l'asse Y_i del sistema di coordinate immagine euclideo. Il valore di s rappresenta l'entità dell'inclinazione dell'asse u rispetto alla direzione dell'asse v. Il parametro s viene introdotto, ad esempio, per compensare la distorsione dovuta alla non perpendicolarità dell'asse ottico con il chip fotosensibile in fase di assemblaggio del dispositivo.

Le grandezze u_0 e v_0 rappresentano l'offset delle coordinate dell'immagine euclidee rispetto a quelle affini.

Ora è possibile definire il modello generale di una fotocamera pinhole. La proiezione effettuata, come già detto, è lineare e va dallo spazio proiettivo tridimensionale $\mathbb{P}^3$ allo spazio proiettivo bidimensionale $\mathbb{P}^2$. É il prodotto di tre fattori dati dall'Equazione 3.4, dall'Equazione 3.9 con P come in 3.11 e dall'Equazione 3.12

$$\tilde{\mathbf{u}} \simeq K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & -R\mathbf{t} \\ 0^T & 1 \end{bmatrix} \mathbf{X}_w \tag{3.13}$$

Il prodotto del secondo e del terzo fattore mettono in evidenza una caratteristica interna di rilievo che si nota riscrivendo l'Equazione 3.13

$$\tilde{\mathbf{u}} \simeq K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & -R\mathbf{t} \\ 0^T & 1 \end{bmatrix} \mathbf{X}_w = K \begin{bmatrix} R | -R\mathbf{t} \end{bmatrix} \mathbf{X}_w = M\mathbf{X}_w \tag{3.14}$$

Esprimendo quindi il punto della scena in coordinate omogenee è possibile scrivere l'operazione di proiezione prospettica in forma lineare usando un'unica matrice 3×4, chiamata **matrice di proiezione prospettica** (o **matrice della fotocamera**). La sottomatrice sinistra di dimensioni 3×3 di M rappresenta una rotazione, mentre la colonna a destra una traslazione.

3.1.3 Calibrazione

La calibrazione è una procedura che consiste nel misurare con accuratezza i parametri intrinseci ed estrinseci del modello della fotocamera. Poiché questi parametri governano il modo in cui punti dello spazio si proiettano sul piano dell'immagine l'idea è che, conoscendo le proiezioni di punti 3D di coordinate note (punti di calibrazione), sia possibile ottenere i parametri incogniti risolvendo le equazioni della proiezione prospettica. Alcuni metodi classici, come l'algoritmo degli 8 punti, sono riportati in [13]. Siccome nel progetto in questione viene utilizzata una fotocamera che fornisce una mappa di profondità, verrà sfruttata questa mappa per elaborare una nuova soluzione per la procedura di calibrazione. Dato che quanto sviluppato è completamente nuovo, si rimanda al Capitolo 5 per la spiegazione passo passo.

3.1.4 Stereopsi e geometria epipolare

La differenza ovvia che ognuno può notare fra il sistema di visione umano e quello di una fotocamera, è che nel primo sono presenti due occhi mentre nel

secondo un singolo obiettivo. La caratteristica fondamentale della visione stereoscopica umana è la percezione della profondità del mondo in cui un individuo si trova. Ciò permette il movimento nell'ambiente consentendo di localizzare, e quindi evitare, eventuali ostacoli.

La visione stereoscopica ha avuto e continua ad avere una grande importanza nel campo della visione computazionale. Con l'utilizzo di due o più dispositivi si cerca di ricavare informazione circa la profondità della scena sfruttando la geometria degli stessi apparecchi.

La calibrazione di una fotocamera e la conoscenza delle coordinate di un punto dell'immagine permettono di determinare univocamente un raggio nello spazio. Se invece si hanno due fotocamere calibrate che osservano lo stesso punto $\mathbf{X}_w$ della scena, le coordinate 3D del punto possono essere calcolate come l'intersezione di due raggi. Questo è il principio base della visione stereoscopica che tipicamente si struttura in tre punti:

- calibrazione delle fotocamere;
- definizione delle corrispondenze fra coppie di punti dall'immagine destra e sinistra;
- ricostruzione tridimensionale delle coordinate dei punti nella scena.

Nel seguito di questa sezione verranno indicate le grandezze relative all'immagine sinistra senza nessun apice, mentre per quelle relative all'immagine destra verrà posto un apice primo, e.g. **u** e **u'**.

La geometria di un sistema con due fotocamere è rappresentata in Figura 3.5. La linea che collega i due centri ottici **C** e **C'** è chiamata **linea di base**, o **baseline**. La linea di base interseca i piani immagine negli **epipoli e** ed **e'**. In alternativa, un epipolo è l'immagine del centro di proiezione di una fotocamera in un'altra fotocamera, $\mathbf{e} = M\mathbf{C'}$ e $\mathbf{e'} = M'\mathbf{C}$.

Ogni punto $\mathbf{X}_w$ della scena osservato da due fotocamere, assieme ai due raggi corrispondenti che passano per i centri ottici **C** e **C'**, definisce il **piano epipolare**. Questo piano interseca i piani immagine nella **linea epipolare l** e **l'**. In alternativa, una linea epipolare è la proiezione del raggio di una fotocamera nell'altra. Tutte le linee epipolari si intersecano nell'epipolo.

Siano **u** e **u'** le proiezioni di un punto $\mathbf{X}_w$ della scena nella fotocamera sinistra e destra rispettivamente. Il raggio $\mathbf{CX}_w$ rappresenta tutte le possibili posizioni di $\mathbf{X}_w$ per l'immagine sinistra e corrisponde alla linea epipolare **l'** nell'immagine destra. Il punto **u'** nell'immagine destra, che corrisponde a **u**, deve quindi giacere sulla linea epipolare **l'** nell'immagine destra, $\mathbf{l'}^T\mathbf{u'} = 0$. La situazione è completamente simmetrica, si ha infatti anche $\mathbf{l}^T\mathbf{u} = 0$. Il fatto che la posizione della proiezione di un punto sulle due immagini non sia arbitraria, si definisce come **vincolo epipolare**.

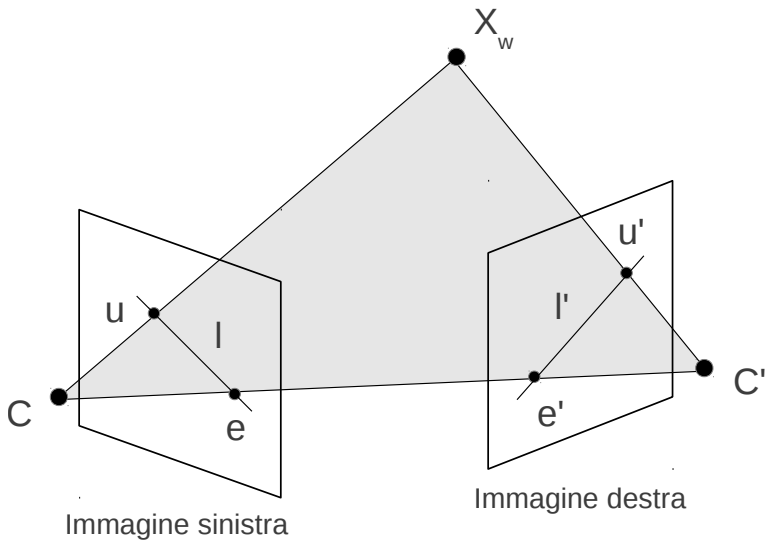


Figura 3.5: Geometria di due fotocamere

3.2 Elaborazione delle immagini

Con il termine *elaborazione delle immagini* si intende qualsiasi forma di trattamento a cui venga sottoposto un segnale di ingresso, dove il segnale d'ingresso è appunto un'immagine. Il risultato del processo può essere ancora un'immagine oppure un insieme di caratteristiche o parametri relativi all'immagine in questione. É bene sottolineare che l'elaborazione di un'immagine non aumenta il suo contenuto informativo. É però importante in quanto è possibile eliminare l'informazione che non è necessaria per quella determinata applicazione rendendo più semplici e meno onerosi, dal punto di vista del calcolo, i passaggi successivi. Quindi lo scopo dell'elaborazione delle immagini è migliorare i dati a disposizione riducendo distorsioni non desiderate e dando risalto a caratteristiche, dette anche *features*, per agevolare gli step successivi.

I sistemi di elaborazione solitamente necessitano che le immagini siano in formato digitale. Il processo di digitalizzazione consiste nella conversione da segnale analogico a digitale e si articola nei passaggi:

- campionamento dell'immagine analogica in una matrice;
- quantizzazione in cui, ad ogni elemento della matrice viene assegnato un valore intero compreso in un dato intervallo.

Per i dettagli si rimanda a [5] e [12]. In base alle caratteristiche di queste operazioni si avrà un dispositivo di acquisizione di qualità più o meno elevata che fornirà l'immagine di partenza su cui operare.

Fra i vari metodi esistenti è possibile fare una distinzione in base alla quantità di pixel che vengono utilizzati per elaborare l'immagine di partenza. Ai fini del progetto è interessante analizzare quelli che sfruttano un basso numero di pixel adiacenti rispetto ad un dato pixel di un'immagine, detti metodi di *local pre-processing*. Spesso le operazioni pre-processing prendono il nome di *filtraggio*. Per gli altri metodi, che non verranno sfruttati in questo progetto, si rimanda a [13].

3.2.1 Image smoothing

Con la locuzione *image smoothing*[2] si intende l'insieme dei metodi di local pre-processing atti alla rimozione del rumore. Lo smoothing allo stesso tempo degrada l'immagine sfocandone i bordi, elementi dal peso informativo considerevole. Sarà dunque di interesse studiare dei metodi che riescano ad attenuare il rumore e contemporaneamente preservare il più possibile la nitidezza dell'immagine. Lo smoothing locale è in grado di eliminare rumore impulsivo o che appare come linee sottili, ma non degradazioni che si presentano come grandi macchie (*blob*). Diverse sono le soluzioni applicabili. Di seguito ne vengono elencate alcune e descritte le caratteristiche.

Media temporale Si assuma che il rumore ν sia una variabile aleatoria indipendente a media nulla e deviazione standard σ. Si può ottenere un'immagine filtrata operando la media temporale sull'acquisizione di n immagini della stessa scena (che deve essere statica). Il risultato dello smoothing è una media di n punti corrispondenti nelle n immagini $d_1, \ldots, d_n$ con sovrapposto il rumore $\nu_1, \ldots, \nu_n$

$$\frac{d_1, \ldots, d_n}{n} + \frac{\nu_1, \ldots, \nu_n}{n} \tag{3.15}$$

dove ogni immagine si intende espressa nella notazione *segnale + rumore additivo*, $i_k = d_k + \nu_k$, con $k = 1, \ldots, n$. Il secondo termine descrive gli effetti del rumore, che è ancora una variabile aleatoria a media nulla ma di deviazione standard $\sigma/\sqrt{n}$. La deviazione standard è diminuita di un fattore $\sqrt{n}$ e dunque facendo tendere n all'infinito si può eliminare completamente il rumore. Ciò è vero solo in parte in quanto, all'aumentare di n, si ha una continua diminuzione del miglioramento fino

[2] In italiano: spianare, levigare. Spesso verranno lasciati termini non tradotti in quanto più comprensibili e appartenenti al linguaggio tecnico comune.

a diventare insignificante. Il valore della deviazione standard diventa pressoché costante. In altre parole, da un certo numero di immagini in poi, l'incremento di n non produce miglioramenti apprezzabili. Da tenere in considerazione il costo in termini di occupazione di memoria e il tempo di calcolo. La funzione che restituisce il pixel filtrato può essere espressa dunque come

$$f(i,j) = \frac{1}{n}\sum_{k=1}^{n} d_k(i,j) \tag{3.16}$$

dove i e j sono le coordinate del pixel nell'immagine.

Media spaziale In alcuni casi può presentarsi la possibilità di avere una singola immagine per l'operazione di filtraggio. In questo caso è possibile effettuare la media spaziale, ovvero dato un pixel si calcola la media con valori dei pixel circostanti. Il numero di pixel coinvolti nell'operazione è definito dalla maschera. Per considerare una zona di dimensioni 3×3, la maschera h è

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.17}$$

Il peso del pixel centrale è spesso aumentato nella maschera h per avere una migliore approssimazione delle proprietà del rumore con una distribuzione di probabilità Gaussiana

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.18}$$

Anche con questa modalità si ripresenta la questione della sfocatura dei bordi, problema che degrada sempre più l'immagine con l'aumento delle dimensioni della maschera.

Mediana In teoria della probabilità la mediana M di una variabile aleatoria x è definita come il valore per cui la probabilità del risultato $x < M$ è $0,5$. Nel caso in cui si consideri un insieme ordinato di valori, la mediana coincide con il valore centrale. Il filtraggio che fa uso della mediana è non lineare ma riduce la degradazione dei bordi, [16]. L'idea è di sostituire il valore di un dato pixel con la mediana dei pixel limitrofi. Questo procedimento è particolarmente efficiente per la rimozione del

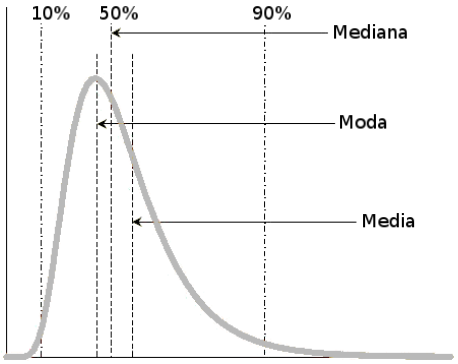


Figura 3.6: Rappresentazione grafica della mediana

rumore di tipo impulsivo. Il punto debole di questo metodo è l'onerosità dei calcoli in quanto, per ogni pixel si deve effettuare l'ordinamento dei valori su cui calcolare la mediana. In [9] è proposto un approccio differente in cui si sfrutta il movimento della finestra: nel movimento per righe e per colonne non è necessario eliminare tutti i pixel candidati al calcolo in quanto da un pixel all'altro cambia solo una riga o colonna, nella finestra. Dunque basterà sostituire la nuova riga o colonna, con la rispettiva che esce dalla finestra.
Lo svantaggio principale dell'uso della mediana con maschere rettangolari è il degrado di linee sottili e angoli ben definiti[3]. Questo effetto può essere ridotto usando forme diverse di maschera come, ad esempio, quella riportata in Figura 3.7, che preserva le linee verticali ed orizzontali.

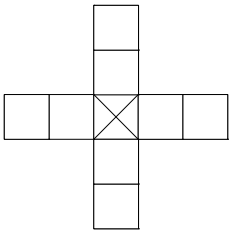


Figura 3.7: Machera alternativa per filtro a mediana

[3]Come si vedrà nel seguito, a causa della qualità del sensore, queste condizioni non sono vincolanti per l'uso di questo filtro.

3.2.2 Edge detection

L'*edge detection* è una delle operazioni più comuni ed importanti nell'analisi delle immagini tanto che, in letteratura, è presente un numero enorme di algoritmi con cui si cercano soluzioni migliori. La motivazione di questa attività di ricerca risiede nel fatto che i bordi formano i contorni degli oggetti. Un edge (o bordo) è il confine fra un oggetto e lo sfondo oppure indica la frontiera lungo la quale due oggetti si sovrappongono. Ciò significa che, se tutti i bordi degli oggetti presenti in un'immagine possono essere definiti accuratamente, tutti gli oggetti possono essere riconosciuti. In questo modo si può ricavare la collocazione nell'immagine di un dato oggetto oppure misurare proprietà quali area, perimetro, forma.

L'analisi descrive variazioni di funzioni continue ricorrendo alle derivate; la funzione dell'immagine dipende da due variabili (le coordinate nell'immagine) e dunque gli operatori che descrivono gli edge sono espressi in derivate parziali. Un cambiamento della funzione immagine può essere descritto da un gradiente che punta in direzione della crescita maggiore della funzione immagine.

Un edge è una proprietà che fa riferimento ad un singolo pixel ed è calcolato dal comportamento della funzione immagine nell'intorno di quel pixel. É una variabile vettore di due componenti, *intensità* e *direzione*. L'intensità dell'edge è l'intensità del gradiente mentre la direzione dell'edge ϕ è ruotata rispetto alla direzione del gradiente ψ di $-90°$. La direzione del gradiente dà la direzione di massima crescita della funzione, e.g. dal nero $f(i,j) = 0$ al bianco $f(i,j) = 255$. Quanto appena esposto è riportato graficamente in Figura 3.8.

L'intensità del gradiente |grad $g(x,y)$| e la sua direzione ψ sono calcolate come:

$$|\text{grad } f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{3.19}$$

$$\psi = \arg\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) \tag{3.20}$$

dove $\arg(x,y)$ è l'angolo (in radianti) dall'asse x al punto (x,y). Un'immagine digitale è per sua natura discreta, mentre l'Equazione 3.19 e l'Equazione 3.20 sono relative a funzioni continue. É quindi necessario ricorrere all'uso della derivata discreta. Le differenze nell'immagine f nella direzione verticale (per i fissato) e orizzontale (per j fissato) sono date da:

$$\Delta_i f(i,j) = f(i,j) - f(i-n,j) \tag{3.21}$$

$$\Delta_j f(i,j) = f(i,j) - f(i,j-n) \tag{3.22}$$

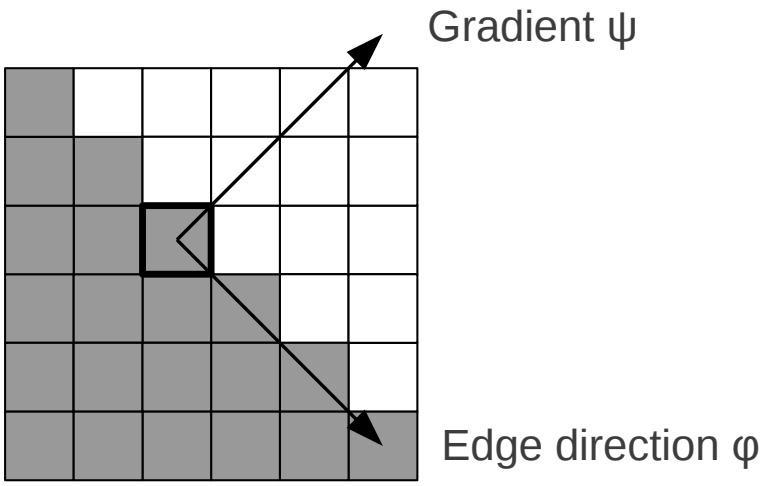


Figura 3.8: Direzione dell'edge e del gradiente

dove n è un intero, solitamente pari ad 1. Il valore di n dev'essere scelto piccolo abbastanza da fornire una buona approssimazione della derivata, ma sufficientemente grande da trascurare cambiamenti non rilevanti nella funzione immagine.

Gli operatori che sfruttano il gradiente si possono dividere in tre categorie:

1. operatori che approssimano le derivate della funzione immagine tramite differenze;

2. operatori basati sul fenomeno dello zero-crossing della derivata seconda della funzione immagine (e.g. Canny edge detector);

3. operatori che tentano di associare alla funzione immagine un modello parametrico dell'edge.

Di seguito verrà analizzata solamente la seconda categoria, in particolare ci si soffermerà sull'algoritmo di Canny. Rispetto alla prima classe si ottengono risultati migliori in quanto lo zero-crossing della derivata seconda è più preciso rispetto all'individuazione del picco della derivata prima. La terza classe invece fornisce edge ancora più precisi rispetto alla seconda ma lo sforzo computazionale è eccessivo, per cui gli operatori zero-crossing risultano essere il compromesso migliore. Si rimanda a [13] per i dettagli dei punti non trattati.

Operatori zero-crossing e Canny edge detector

Negli anni '70 le tecniche di edge detection esistenti, e.g. Kirsh, Sobel e Pratt, si basavano sulla convoluzione in intorni molto piccoli e davano buoni risultati solo per specifiche immagini. Il principale svantaggio di questi edge detector è la dipendenza dalle dimensioni degli oggetti e la sensibilità al rumore. L'accorgimento che ha segnato la svolta è stato lo zero-crossing della derivata seconda della funzione immagine.

La derivata prima della funzione immagine deve avere un estremo in corrispondenza dell'edge nell'immagine e la derivata seconda, di conseguenza, deve essere nulla nella stessa posizione. Il fatto notevole è che è più semplice e più preciso trovare una posizione di zero-crossing rispetto ad un estremo. In Figura 3.9 sono riportati, per semplicità in una dimensione, due esempi di edge con le relative derivate prima e seconda. Proprio su questo principio si basa l'edge detector di Canny.

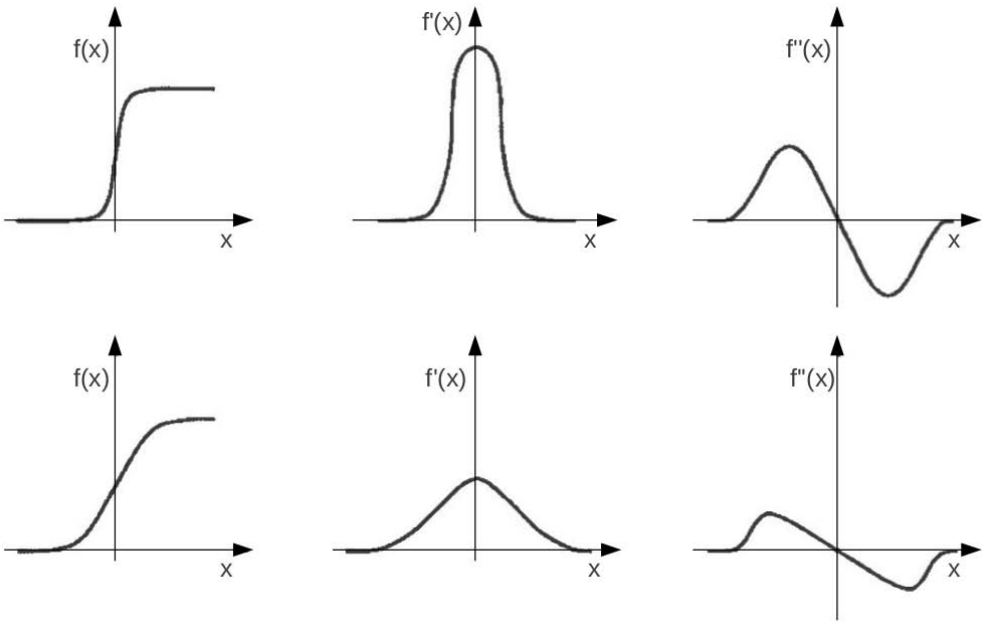


Figura 3.9: Zero-crossing di un edge in 1D

L'edge detector di Canny è considerato come l'algoritmo standard nell'industria. Fu creato nel 1983 da John Canny, ma tutt'ora si rivela un'ottima procedura tanto da avere prestazioni migliori anche rispetto ai nuovi metodi sviluppati. Il problema dell'edge detection fu approcciato da Canny come un problema di signal processing optimization. La soluzione a questo problema

si presentava come una funzione esponenziale piuttosto complessa, ma Canny riuscì a trovare diverse soluzioni per approssimare e ottimizzare la ricerca degli edge. I passi fondamentali dell'edge detector di Canny sono:

1. filtrare l'immagine con una Gaussiana bidimensionale. Solitamente questa operazione è computazionalmente onerosa, dunque viene approssimata con due Gaussiane unidimensionali: una lungo la direzione delle x e l'altra nella direzione delle y;

2. calcolare il gradiente dell'immagine. Questo per vedere i cambiamenti di intensità che indicano la presenza di edge. Questa operazione dà due risultati: uno nella direzione delle x e l'altro nella direzione delle y;

3. non-maximal suppression.

Un edge è caratterizzato da posizione, orientazione e modulo. Operando una convoluzione tra un'immagine e una Gaussiana bidimensionale e poi derivando nella direzione del gradiente (perpendicolare alla direzione dell'edge) si ottiene un semplice ed efficace operatore.
Si supponga che G sia una Gaussiana bidimensionale

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.23}$$

dove x e y sono le coordinate dell'immagine e σ è la deviazione standard. A volte l'Equazione 3.23 si può trovare espressa con un fattore di normalizzazione

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \text{oppure} \qquad G(x, y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Si assuma di voler convolvere l'immagine con un operatore G_n, che è una derivata prima di G nella direzione $\mathbf{n}$

$$G_n = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G \tag{3.24}$$

La direzione di $\mathbf{n}$ deve essere orientata perpendicolarmente all'edge. Anche se questa direzione non è nota a priori, è possibile ottenere una stima robusta dalla direzione del gradiente. Se f è l'immagine, la normale all'edge $\mathbf{n}$ è stimata come

$$\mathbf{n} = \frac{\nabla(G * f)}{|\nabla(G * f)|} \tag{3.25}$$

La posizione dell'edge è quindi in corrispondenza del massimo locale dell'immagine f convoluta con l'operatore G_n nella direzione di $\mathbf{n}$

$$\frac{\partial}{\partial \mathbf{n}} G_n * f = 0 \tag{3.26}$$

Sostituendo nell'Equazione 3.26 l'espressione per G_n data dall'Equazione 3.24, si ottiene

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * f = 0 \tag{3.27}$$

L'Equazione 3.27 mostra come trovare massimi locali nella direzione perpendicolare all'edge. Questa operazione è spesso riferita al **non-maximal suppression**[4].
Per l'associatività delle operazioni di convoluzione e derivazione nell'Equazione 3.27, è possibile prima convolvere l'immagine f con una Gaussiana G e poi eseguire la derivata seconda direzionale usando una stima della direzione $\mathbf{n}$ che si ricava dall'Equazione 3.25. Il modulo del gradiente della funzione immagine f è misurato come

$$|G_n * f| = |\nabla (G * f)| \tag{3.28}$$

Risultati spuri di un singolo edge causati dal rumore creano delle striature. Il fenomeno è molto diffuso nell'operazione di edge detection, meglio conosciuto come *streaking problem*. Per ovviare a questo problema solitamente si pone un'operazione di decisione a due soglie che sfrutta il principio di isteresi all'uscita dell'edge detector. Questo per decidere quali sono gli edge significativi e quali invece degradano il risultato. Se il valore di un pixel di un presunto edge risulta maggiore della soglia superiore, il pixel in questione viene considerato. Se un pixel di un presunto edge non supera la soglia inferiore, viene scartato. Quando invece si trova fra i valori delle due soglie, il pixel viene considerato solo se adiacente a pixel appartenenti ad un edge o altrimenti scartato. Per particolari applicazioni, un possibile procedimento per settare le soglie si basa sulla stima del rapporto segnale-rumore. A tal proposito si veda [2].

Algorithm 1 Canny edge detector

1: Convoluzione di un'immagine f con una Gaussiana.
2: Stima locale della direzione ortogonale all'edge $\mathbf{n}$ usando l'Equazione 3.25 per ogni pixel dell'immagine.
3: Trovare la posizione degli edge usando l'Equazione 3.27 (non-maximal suppression).
4: Calcolare il modulo del gradiente degli edge usando l'Equazione 3.28.
5: Eliminare edge spuri usando una funzione di discriminazione ad isteresi.

[4]Gli edge si verificano nei punti in cui il gradiente è massimo. Pertanto tutti i punti che non sono al massimo devono essere eliminati. Per decidere, devono essere considerati modulo e direzione del gradiente per ogni pixel.

Lo scopo dell'edge detection è creare un'immagine composta da linee a partire da un'immagine di una data scena. Da questa immagine si possono estrarre facilmente features come angoli, linee, curve, che sono usate poi per un'analisi a più alto livello. Da non sottovalutare che un'immagine così strutturata è molto meno complessa rispetto all'originale e permette di occupare meno memoria.

3.2.3 Trasformata di Hough

La trasformata di Hough [8] è una tecnica che permette il riconoscimento di configurazioni di punti presenti in un'immagine (segmenti, curve oppure forme prestabilite). Questa operazione si basa sulla trasformazione di tutti i punti costituenti un'immagine in punti di un nuovo spazio, detto spazio dei parametri. Nella sua versione tradizionale, la trasformata di Hough si applica ad immagini binarie[5] in cui l'informazione associata ad un punto è rappresentata unicamente dalla sua posizione. Per descrivere questa tecnica, ideata da Hough nel 1962, viene preso in esame il caso specifico di identificazione di una retta.

Una linea retta è definita da due punti $A = (x_1, y_1)$ e $B = (x_2, y_2)$, rappresentati in Figura 3.10(a). Tutte le rette passanti per A sono date dall'espressione $y_1 = kx_1 + q$ per qualche valore di k e q. Questo significa che la stessa equazione può essere interpretata nello spazio dei parametri k, q. Tutte le rette passanti per A sono dunque rappresentate dall'equazione $q = -x_1k + y_1$, Figura 3.10(b). Allo stesso modo, le rette passanti per B possono essere rappresentate come $q = -x_2k + y_2$. Nello spazio dei parametri l'unico punto in comune ad entrambe le linee è quello in cui k e q assumono i valori della retta $y = kx + q$ nello spazio dell'immagine originale. Questo significa che *qualsiasi* retta nell'immagine è rappresentata da un unico punto nello spazio dei parametri.

L'idea su cui si basa il processo di identificazione di una retta è trovare tutti i pixel dell'immagine che contengono informazione. Si scelgono due pixel di questo insieme e si valutano i parametri della retta che passa per questi dati pixel. Si ripete quest'operazione per ogni coppia di pixel e si rilevano i parametri (a, b) nello spazio dei parametri che sono risultati più frequentemente dalla trasformata di Hough della retta $y = ax + b$ nell'immagine.

Lo spazio dei parametri non è continuo e spesso viene rappresentato con una struttura rettangolare a celle. Questa matrice di celle prende il nome di **accumulatore** $\mathcal{A}$ e i suoi elementi $\mathcal{A}(k, q)$ sono accumulatori. Per ogni linea i valori dei parametri k e q sono usati per incrementare il relativo accu-

[5]Immagini in cui sono presenti due soli livelli, e.g. bianco e nero.

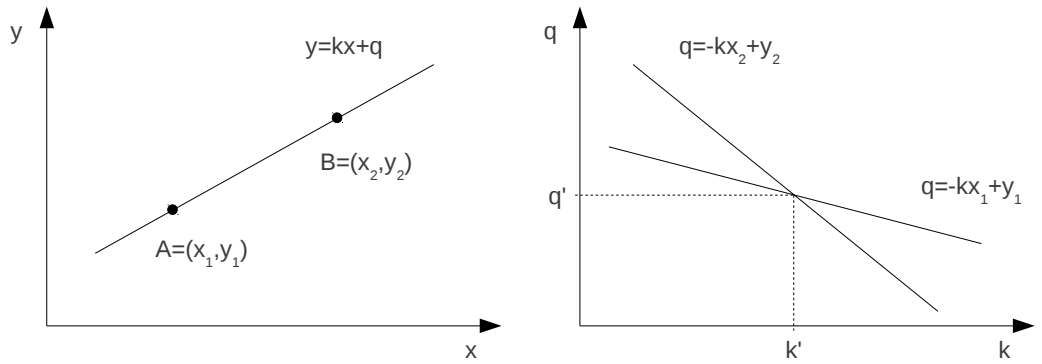


Figura 3.10: Principio della trasformata di Hough: (a) spazio dell'immagine; (b) spazio dei parametri

mulatore $\mathcal{A}(k, q)$. Chiaramente se una linea rappresentata da un'equazione $y = ax + b$ viene rilevata n volte, l'accumulatore $A(a, b)$ verrà incrementato n volte. In questo modo linee che esistono effettivamente nell'immagine contribuiscono in modo notevole all'incremento del relativo accumulatore. Al contrario linee che fanno riferimento a pixel spuri avranno differenti parametri e quindi i relativi accumulatori saranno incrementati sporadicamente. Così facendo il problema di rilevamento di una retta nell'immagine si riduce ad una ricerca di massimo nella matrice $\mathcal{A}$.

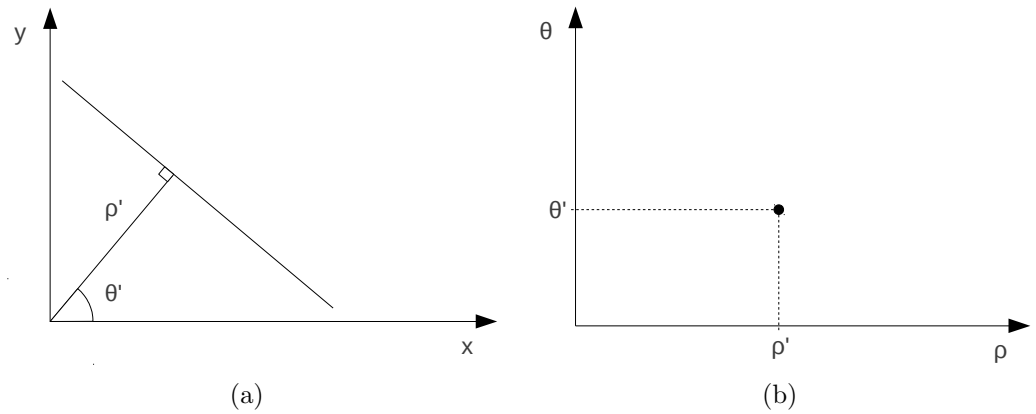
Una proprietà di importanza notevole della trasformata di Hough è l'insensibilità a dati imprecisi[6], alla mancanza di parti di linee ed al rumore nell'immagine. Ciò è dato dalla robustezza della trasformazione dallo spazio dell'immagine allo spazio dei parametri.

Si noti che l'equazione parametrica della retta $y = kx + q$ è adatta solo per la spiegazione del principio su cui si basa la trasformata di Hough. Infatti questa forma è causa di difficoltà nel rilevamento di linee verticali ($k \to \infty$). Se però una linea viene espressa nella forma

$$\rho = xcos\theta + ysin\theta, \tag{3.29}$$

la trasformata di Hough non soffre più di questa limitazione. Il modo di ragionare è il medesimo. Di nuovo, le rette vengono trasformate in punti, come riportato in Figura 3.11.

[6]Si potrà apprezzare questa proprietà nel seguito quando la trasformata di Hough verrà utilizzata nel progetto.



(a) (b)

Figura 3.11: Trasformata di Hough nello spazio ρ, θ: (a) retta nello spazio dell'immagine; (b) spazio dei parametri ρ, θ

3.3 Altre nozioni

Nelle sezioni seguenti vengono richiamati sommariamente dei concetti sfruttati nel progetto.

3.3.1 RANSAC

RANSAC è un acronimo che sta per "RANdom SAmple Consensus". Si tratta di un metodo iterativo per la stima dei parametri di un modello matematico. L'algoritmo è stato formulato per la prima volta da Fischler e Bolles nel 1980, [6].

Si supponga di disporre di dati che sono in relazione lineare fra loro. Sarebbe ragionevole derivare la relazione lineare usando un approccio ai minimi quadrati minimizzando la somma dei quadrati dei residui. Solitamente questo viene fatto ricavando un'espressione per questa somma calcolando la derivata rispetto ai parametri del fit, prima eguagliandola a zero ed infine risolvendola nei parametri.

Nel caso in cui i dati siano imperfetti, il modello risultante sarebbe molto impreciso. Se però il rumore nei dati ha un "buon comportamento" in qualche senso, è comunque possibile migliorare il modello utilizzando metodi statistici. Dall'altro lato invece, se è presente un numero considerevole di outliers nei dati, è possibile che il modello derivato sia fortemente distorto.

Tenendo in considerazione questo fatto, si cerca di identificare gli outliers come punti che hanno un residuo alto rispetto al modello fittato. Questi punti possono essere esclusi e il modello ricalcolato.

Nel caso della ricerca di un fit lineare, per definire una linea sono sufficienti due punti. Si selezionino due punti in modo casuale dal data set e si ipotizzi che la linea che passa per tali punti rappresenti il modello corretto. Per testare il modello appena trovato si valutano quanti dei punti rimanenti sono, in un qualche senso, vicini all'ipotesi fatta. Questi punti sono detti *consensus points*. Se c'è un numero significativo di consensus points allora il modello viene ricalcolato basandosi solamente sull'insieme di questi. Questo avrà l'effetto di migliorare il modello dato che non vengono considerati gli outliers.

Algorithm 2 RANSAC

1: Si supponga di avere un data set di n punti $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ con cui fittare un modello, determinato da almeno m punti ($m \leq n$, quindi per una linea $m = 2$).
2: Settare il contatore di iterazioni $k = 1$.
3: Scegliere in modo casuale m punti da X per calcolare il modello.
4: Per un certo valore di tolleranza ε, si determinano quanti elementi di X sono compresi nella tolleranza del modello ricavato. Se questo numero supera una soglia t il modello viene ricalcolato limitatamente a questo insieme di consenso.
5: Set $k = k+1$. Se $k \leq K$, per un prestabilito valore di K, andare al punto 3. Altrimenti si accetta il modello con il più grande insieme di consenso, oppure fallisce.

Ci sono molti possibili miglioramenti da apportare a questa procedura di base. Ad esempio la selezione random dei punti al passo 3 può essere migliorata cercando di prevedere, attraverso le proprietà del modello che si vuole ricavare, quali potrebbero essere più adatti.

L'algoritmo inoltre dipende dalla scelta di tre parametri, che influiscono notevolmente sul risultato finale:

- ε, la tolleranza con cui i punti vengono scelti oppure scartati per il modello. Di rado può essere determinato analiticamente. Solitamente si fitta un modello, si misurano le deviazioni standard e sulla base di queste si setta ε.

- t, la dimensione dell'insieme dei punti di consenso. Rappresenta il numero minimo di punti per confermare un modello. Lo stesso numero di punti sarà utilizzato per rifinire il modello ottenendo la migliore stima calcolabile.

- K, il numero di iterazioni che deve essere compiuto nella ricerca di un modello soddisfacente.

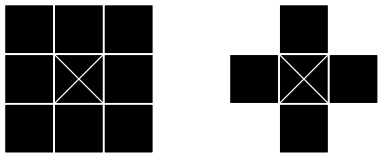
3.3.2 Basi di Morfologia Matematica

La morfologia matematica è una teoria ed una tecnica per l'analisi delle forme geometriche che trova particolare applicazione nell'elaborazione digitale delle immagini. Le operazioni morfologiche sono usate principalmente per i seguenti scopi:

- image pre-processing (filtraggio dal rumore, semplificazione delle forme);
- migliorare la struttura degli oggetti (assottigliare, ispessire, evidenziare);
- separare oggetti dallo sfondo;
- descrivere quantitativamente oggetti (area, perimetro, proiezioni).

Il campo sarà ristretto alla trattazione su immagini binarie, ovvero immagini i cui pixel possono assumere solo due valori, solitamente bianco e nero oppure equivalentemente 0 e 1. I punti che appartengono ad oggetti nell'immagine sono rappresentati dall'insieme X e raffigurati col colore nero. I punti che appartengono a X^c costituiscono invece lo sfondo e corrispondono al colore bianco. Lo spazio bi-dimensionale dell'intera immagine viene indicato con $\mathcal{E}^2$.

Una trasformazione morfologica è data dalla relazione di un insieme di punti di un'immagine X con un altro insieme (più piccolo) di punti B. B è espresso rispetto ad un'origine locale e può avere forme diverse in base al risultato che si vuole ottenere. Le più comuni sono riportate in Figura 3.12.



(a) (b)

Figura 3.12: Tipica struttura dell'insieme B

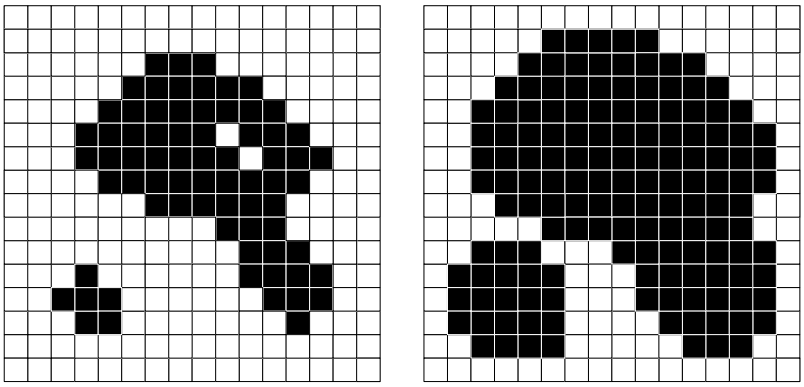
Gli operatori di base che caratterizzano questa teoria sono invarianti alla traslazione e sono: dilatazione, erosione, apertura e chiusura.

Dilatazione

La dilatazione è usata per riempire piccoli buchi o strette strisce negli oggetti. Con i simboli definiti in precedenza, l'operazione di dilatazione $\oplus$ si esprime come

$$X \oplus B = \left\{ p \in \mathcal{E}^2 : p = x + b, x \in X \text{ e } b \in B \right\} \tag{3.30}$$

Un esempio è riportato in Figura 3.13.



(a) (b)

Figura 3.13: Dilatazione

Erosione

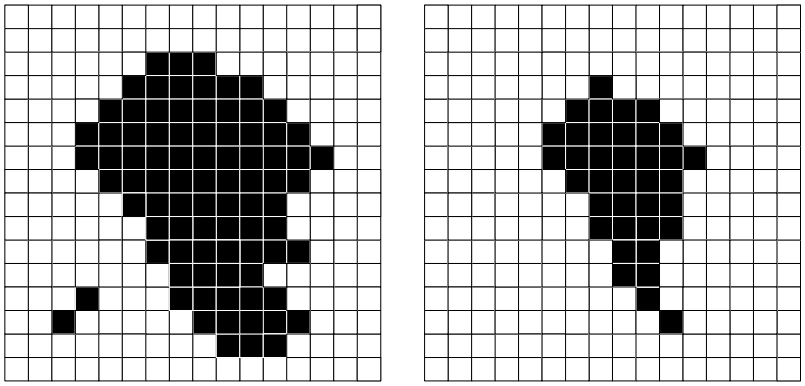
Al contrario della dilatazione, l'erosione è sfruttata per ampliare i fori presenti negli oggetti oppure per eliminare punti spuri sullo sfondo. L'operazione di erosione $\ominus$ è definita da

$$X \ominus B = \left\{ p \in \mathcal{E}^2 : p + b \in X \text{ per ogni } b \in B \right\} \tag{3.31}$$

Un esempio è riportato in Figura 3.14.

Apertura e Chiusura

Le operazioni di dilatazione ed erosione comportano rispettivamente un aumento ed una diminuzione della dimensione degli oggetti nell'immagine. Quando è invece necessario mantenere le dimensioni originali, le operazioni di dilatazione ed erosione possono essere combinate fra loro in modo che l'effetto di una venga compensato dall'effetto dell'altra. É importante notare che questi



(a) (b)

Figura 3.14: Erosione

operatori non sono uno l'inverso dell'altro, ovvero l'ordine in cui vengono eseguite le operazioni influisce in modo diverso sul risultato finale.

Quando l'erosione è seguita dalla dilatazione si attua la trasformazione morfologica chiamata **apertura**. L'apertura di un'immagine X con un insieme B si indica con $X \circ B$ ed è definita da

$$X \circ B = (X \ominus B) \oplus B \tag{3.32}$$

Viceversa, quando la dilatazione è seguita dall'erosione, si ha la **chiusura**. La chiusura di un'immagine X con un insieme B si indica con $X \bullet B$ ed è definita da

$$X \bullet B = (X \oplus B) \ominus B \tag{3.33}$$

É possibile eseguire più cicli di erosione e/o dilatazione impiegando insiemi B di forme diverse in base ai dati a disposizione ed al risultato che si vuole ottenere.

Capitolo 4

Microsoft Kinect

Microsoft Kinect, inizialmente conosciuto con il nome Project Natal, è un dispositivo originariamente pensato per la piattaforma di gioco Xbox 360. Questo apparecchio ha la particolarità di essere sensibile al movimento del corpo umano, caratteristica che consente al giocatore il controllo del sistema senza dover far uso di joystick o strumenti similari. Questa novità ha consentito all'azienda produttrice di introdurre un nuovo concetto di gioco riscuotendo grande entusiasmo fra gli appassionati e di fatto battendo sul tempo gli altri produttori concorrenti.

Figura 4.1: Microsoft Kinect

La prima versione è stata presentata a giugno 2009. La periferica era specifica per la sola Xbox 360. Nel dicembre 2010 sono stati rilasciati i driver open source che consentono di accedere alle funzioni audio, video e ai sensori di profondità. Questo ha provocato un grande movimento di ricerca e di sviluppo di software allo scopo di trovare nuove modalità di utilizzo di un

dispositivo che, nato come periferica per console, si sta affermando sempre più in applicazioni a livello industriale.

A partire dal febbraio 2012 Microsoft ha reso disponibile una versione speciale di Kinect, specifica per pc dotati di sistema operativo Windows 7 e Windows 8.

Kinect per Windows e Kinect per Xbox 360 hanno fondamentalmente la stessa struttura. La differenza sostanziale risiede nell'ottica: la focale della prima è concentrata ad appena 50cm invece che a 2 metri circa, come accade nella seconda. Questo perché Kinect per Windows è pensato per essere collocato sulla scrivania e consentire la gestione del pc attraverso movimenti delle mani sostituendo il mouse. Dunque non si tratta più di ampi spazi quali una stanza e di movimenti notevoli come lo spostamento di un braccio o di una gamba, bensì di un ambiente di lavoro ristretto quale può essere una scrivania e dei movimenti che si riducono al gesticolare delle mani, quindi più difficili da cogliere.

Siccome nel progetto in questione si è fatto uso di Kinect per Xbox 360, i valori che verranno menzionati nel seguito faranno riferimento a questa versione.

4.1 Struttura e caratteristiche di Kinect per Xbox 360

Microsoft Kinect è dotato di vari sensori per recepire i comandi dell'utente, come riportato in Figura 4.2.

Una camera RGB memorizza i colori di una scena in 3 canali (Red - Green - Blue) con risoluzione 640×480 a 30 frames al secondo. In alternativa si può avere una risoluzione di 1280×960 a scapito del frame rate, che diminuisce a $12 fps$.

Un emettitore a raggi infrarossi (IR) genera un fascio di raggi che colpisce le superfici degli oggetti e le persone. Il sensore di profondità capta i raggi riflessi convertendo l'informazione in una misura di distanza che va dal sensore all'oggetto che ha riflesso la luce. In questo modo si ottiene una mappa di profondità con risoluzione 640×480 a 30 frames al secondo. É possibile aumentare la risoluzione a 1280×960 a scapito del frame rate come nel caso precedente. Siccome in questo caso viene usata la luce nel campo infrarosso, è possibile otterene mappe di profondità in ogni condizione di illuminazione.

Un array di 4 microfoni consente di recepire i suoni in modo da poter utilizzare comandi vocali se necessario. La presenza di più microfoni consente

di individuare la direzione della sorgente sonora e quindi di isolarla dal rumore di fondo prodotto dall'ambiente.

Un accelerometro a 3 assi permette invece al sistema di capire l'orientazione della periferica.

Infine un motore posto nella base consente una rotazione verso l'alto e verso il basso di 27°per mantenere il dispositivo orizzontale.

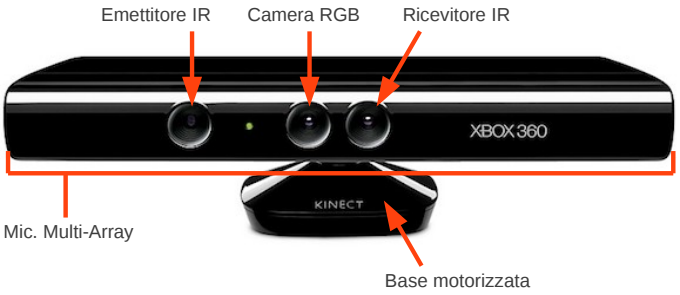


Figura 4.2: Sensori di Microsoft Kinect

Per quanto concerne il progetto realizzato, saranno d'interesse solamente la camera RGB e l'emettitore-ricevitore IR.

Un aspetto molto importante da tenere in considerazione è la distanza a cui un oggetto può essere posizionato rispetto a Kinect per essere riconosciuto. Più precisamente, la distanza minima e massima da cui i raggi che rimbalzano sulla sua superficie riescono ad essere catturati dal sensore IR. Il sensore di profondità ha due range: il *default range* e il *near range*, Figura 4.3. Il default range è disponibile sia in Kinect per Windows che in Kinect per Xbox 360, mentre il near range si può sfruttare solamente nella versione Kinect per Windows.

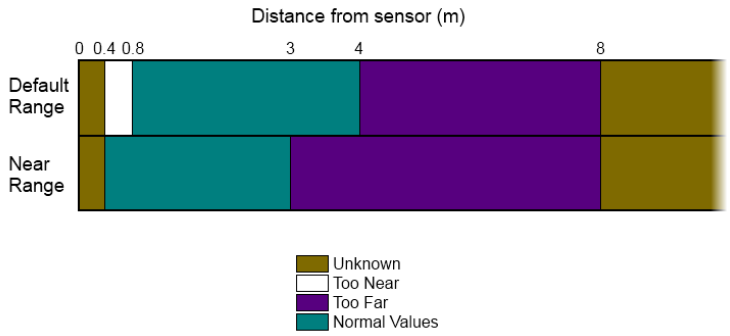


Figura 4.3: Range profondità di Microsoft Kinect espresso in metri

Attualmente è in studio un nuovo sistema di gestione della periferica che permette di ridurre il tempo di lavorazione delle immagini a 5ms.

Capitolo 5

Descrizione progetto

Come anticipato nell'introduzione, lo scopo del progetto è realizzare la ricostruzione 3D usando più Kinect. Il punto di partenza è un programma base sviluppato da Euclid Labs che gestisce la sola acquisizione dell'informazione da una singola Kinect.

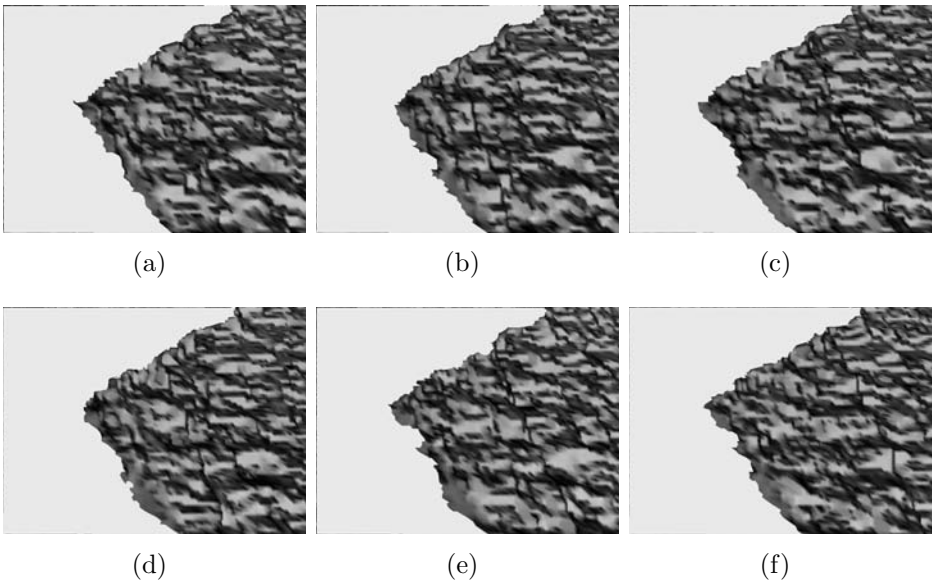
Per la sua natura, Kinect fornisce dei dati molto rumorosi (ciò giustifica in parte il basso costo dell'apparecchio). Con un'operazione di filtraggio è però possibile ridurre l'effetto del rumore consentendo maggior precisione nelle operazioni successive.

5.1 Filtraggio dei dati acquisiti

Nel progetto di partenza è già stato implementato un filtro che esegue la media temporale su 3 immagini.

Come primo step è stata cercata una soluzione alternativa per aumentare l'attenuazione del rumore nei dati. In riferimento a quanto riportato nella Sezione 3.2.1 è stato scelto di implementare il filtro basato sull'operazione di mediana. Il fatto che gli angoli vengano degradati non è un problema rilevante in quanto, l'informazione ricavabile da questi dati, è inutilizzabile per lo scopo previsto. In Figura 5.1 è riportata una serie di immagini che illustrano come lo stesso angolo venga elaborato da Kinect in acquisizioni differenti.

La mediana viene calcolata sulla singola immagine e per ogni pixel. Il numero di valori su cui viene calcolata varia a seconda della forma e della dimensione della maschera che si desidera applicare.



(a) (b) (c)

(d) (e) (f)

Figura 5.1: Serie di acquisizioni di angoli

5.1.1 Maschere per il filtro a mediana

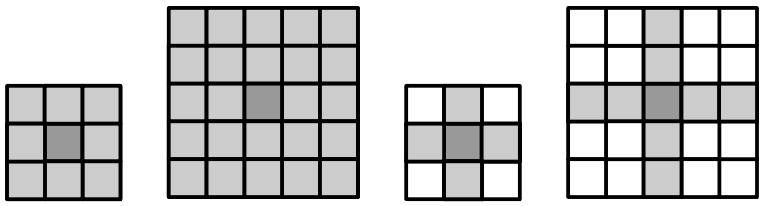
In base alla forma della maschera applicata, e quindi anche alla sua dimensione, si ottengono risultati differenti nell'immagine filtrata. Una classica forma è quella quadrata (o rettangolare). Con questa struttura vengono corrotti linee sottili ed angoli ma, vista la qualità dei dati a disposizione, ciò non rappresenta un problema. Una particolare forma a "+" consente invece di preservare linee verticali ed orizzontali. Per le operazioni successive al filtraggio non ci sono particolari necessità riguardo le direzioni dei lati indicate. Tuttavia si è voluto studiare il comportamento di questa maschera. Per entrambe queste strutture sono state considerate dimensioni differenti in modo da poter apprezzare possibili differenze. Vengono elencate in Tabella 5.1 le maschere che sono state considerate nei test, specificanone il nome e dandone una breve descrizione riguardo la struttura.

Tutte le maschere hanno come pixel di riferimento per l'applicazione il pixel centrale della maschera stessa (ovvero, applicando ad esempio la maschera 3x3 ad un dato pixel p, il pixel centrale della maschera coinciderà con p). In Figura 5.2 e Figura 5.3 è riportata una rappresentazione grafica delle maschere menzionate, dove sono evidenziati con le tonalità di grigio i pixel che partecipano all'operazione di mediana. Con la tonalità più scura è indicato il pixel di riferimento.

Per quanto riguarda la mediana denominata *hybrid* si può far riferimento

Nome	Forma
`3x3`	quadrato di 9 pixel
`5x5`	quadrato di 25 pixel
`+1`	croce composta da 5 pixel
`+2`	croce composta da 9 pixel
`hybrid`	struttura composta, vedere descrizione nel seguito

Tabella 5.1: Tipi di maschere



(a) 3x3 (b) 5x5 (c) +1 (d) +2

Figura 5.2: Maschere per l'operazione di mediana

a Figura 5.3. Il risultato finale si basa sulla composizione di più elementi: si calcola la mediana con una maschera di tipo "+1"; si calcola una seconda mediana con maschera "X1"[1]; ed infine il pixel di riferimento. Di questi 3 valori si calcola la mediana, che rappresenta il risultato finale. Rispetto alla mediana classica, questo tipo di operazione riesce a preservare maggiormente le caratteristiche degli angoli. Come detto più volte gli angoli subiscono un notevole degrado. Con la mediana hybrid si vuol capire se c'è qualche margine di miglioramento ma soprattutto come si comporta in generale.

5.1.2 Strategie di mediana

A fronte di quanto già presente nel progetto di partenza, si è deciso di testare le varie maschere in diverse modalità così definite:

single viene acquisita una singola immagine e su questa viene eseguita l'operazione di mediana;

[1]Come per i casi "+1" e "+2" si continua a usare una scrittura che richiami la forma dei pixel considerati nel calcolo: a forma di "X" e con le estremità lunghe 1 pixel, da cui "X1" appunto. É comunque possibile vedere il dettaglio in Figura 5.3.

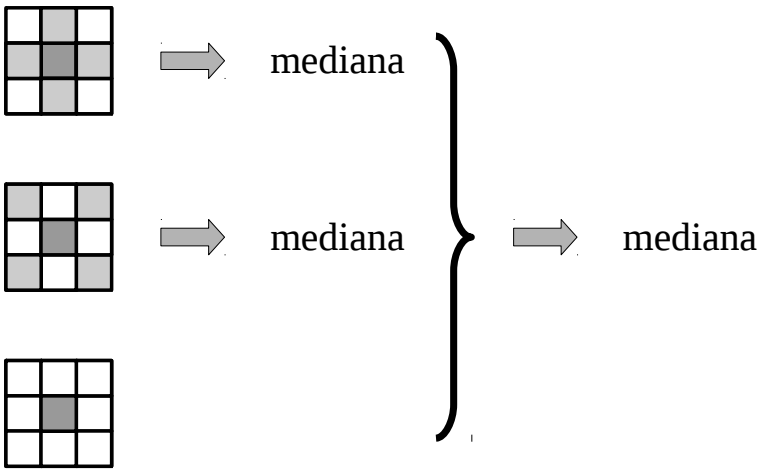


Figura 5.3: Operazione di mediana hybrid

hybrid viene acquisita una singola immagine e su questa viene eseguita la mediana hybrid;

mean vengono acquisite N immagini, su ognuna viene eseguita la mediana (mantenendo la medesima maschera per tutte le N immagini) ed infine viene fatta la media delle N immagini.

Da notare che la mediana hybrid viene utilizzata solo con la strategia hybrid appunto. I vari tipi di maschere interverranno solo negli altri 2 casi.

5.2 Valutazione della bontà del filtro

Per valutare le differenze tra i vari filtri proposti e decidere quale può essere ritenuto migliore, non è sufficiente un'analisi visiva dei dati dall'ambiente 3D. É stata dunque introdotta della statistica sugli errori per valutare la bontà delle soluzioni considerate. Il procedimento prevede di creare un riferimento rispetto al quale verranno valutate alcune grandezze caratteristiche. Si inquadra una superficie piana, ad esempio un muro[2], con i cui dati[3] viene fittato

[2]Per quanto un muro possa non essere liscio, le irregolarità sono al di sotto dell'accuratezza di Kinect, dunque non contribuiscono all'introduzione di errori.

[3]I dati si intendono non filtrati. Vengono presi così come Kinect li acquisisce.

un piano. Il piano fittato rappresenta il riferimento[4]. Si effettuano una serie di misure per ogni maschera e per ogni modalità, in modo da avere risultati più affidabili. Di volta in volta si calcolano i seguenti valori confrontandoli con quelli del riferimento:

- media dei valori acquisiti;
- scarto quadratico medio;
- varianza;
- errore massimo;
- somma (assoluta) degli errori.

Sulla base di queste grandezze si è in grado di decidere quale scelta sia la migliore, o quale sia il miglior compromesso. In Tabella 5.2 sono riportati i valori indicativi di una delle misurazioni effettuate. Per questioni di visualizzazione alcune etichette sono state troncate. L'ultimo gruppo di misure, denotato con l'etichetta *Media orig.* alla voce *Mediana*, è stato effettuato con il filtro presente nel programma di partenza. Questo per valutare se con la mediana si poteva migliorare l'operazione di filtraggio oppure se il comportamento risultava peggiore tanto da scartarlo ritornando alla vecchia soluzione.

A seguito di varie misurazioni e confronti è emerso che la maschera 5x5 è quella che dà i risultati migliori seguita dalla maschera 3x3, entrambe con media su più immagini. A causa però dell'onerosità dei calcoli per la 5x5, è stato scelto di ricorrere alla maschera 3x3 con media su 3 immagini[5] in condizioni di normale utilizzo. Tuttavia la maschera 5x5 può essere sfruttata nella fase di calibrazione dove non è fondamentale avere prestazioni di calcolo elevate, piuttosto è necessario essere il più precisi possibile per garantire una ricostruzione più accurata.

5.3 Calibrazione

Ora che è stata definita la regola per il filtraggio, si possono avere dei dati ripuliti dal rumore in una qualche misura. Il prossimo passo riguarda la

[4]Si è visto che, scartando una fascia di dati lungo i bordi dell'immagine acquisita, il fit migliora. La spiegazione sta nel fatto che la lente distorce l'immagine, distorsione che è minore al centro e che si può apprezzare anche nel visualizzatore 3D sovrapponendo piano fittato e immagine acquisita.

[5]Il numero di immagini su cui mediare è un compromesso fra velocità di calcolo e miglioramento degli indici considerati. Aumentandolo eccessivamente non si ottengono più miglioramenti significativi a scapito del tempo di elaborazione.

Media	S. Q. M.	Varianza	Er. Max	Som. Er.	Mediana	Masch.
0,167475	5,688034	32,325710	20,001228	380150	Riferimento	—
0,051697	5,699451	32,481155	18,947739	384422	single	3x3
0,021356	5,629774	31,694006	17,467146	385378	single	3x3
0,045747	5,649783	31,917881	18,061523	387329	single	3x3
0,035039	5,636100	31,764278	17,962959	386557	single	3x3
-0,166864	5,644755	31,835569	15,453640	386451	single	5x5
-0,101069	5,640883	31,809519	16,312727	386489	single	5x5
-0,060858	5,582854	31,164331	15,874800	383120	single	5x5
-0,066483	5,567991	30,998457	15,422945	382607	single	5x5
1,244938	5,946165	33,807018	20,622803	384870	single	+1
1,320024	7,422050	53,344746	22,166002	482715	single	+1
1,182202	7,403583	53,415504	22,255615	478873	single	+1
1,194625	7,347298	52,555676	22,334328	476562	single	+1
1,130949	5,884456	33,347752	19,415928	379735	single	+2
1,152370	5,805688	32,378056	19,581264	374793	single	+2
1,228477	5,911817	33,440533	19,581264	381483	single	+2
1,284351	5,993511	34,272701	19,426495	386492	single	+2
-0,019665	5,878831	34,560337	20,236677	386026	hybrid	—
0,018758	5,791627	33,542572	19,835571	383829	hybrid	—
0,021828	5,806475	33,714607	20,158194	384511	hybrid	—
0,006079	5,806671	33,717220	19,835600	383220	hybrid	—
0,070828	5,712955	32,633213	20,003906	374657	mean	3x3
0,077418	5,693120	32,405609	20,001228	375749	mean	3x3
0,114376	5,685173	32,307991	20,066952	376105	mean	3x3
0,061253	5,676791	32,222290	20,066952	374452	mean	3x3
0,066305	5,692245	32,397148	20,541035	376524	mean	3x3
0,028206	5,624692	31,636532	18,641724	369231	mean	5x5
0,099626	5,628508	31,670053	20,361958	371030	mean	5x5
0,138095	5,626137	31,634478	20,093384	371483	mean	5x5
0,158149	5,662931	32,043720	20,361958	372995	mean	5x5
0,207442	5,869632	34,409531	22,789240	379576	mean	+1
0,223309	5,794651	33,528217	22,789185	377022	mean	+1
0,261812	5,762719	33,140442	22,789240	375370	mean	+1
0,186211	5,763630	33,184673	22,789240	374648	mean	+1
0,298178	5,730242	32,746796	20,609009	374405	mean	+2
0,345001	5,747410	32,913795	18,969959	375824	mean	+2
0,337253	5,734990	32,776245	22,710449	374900	mean	+2
0,313055	5,759433	33,073185	22,710392	377053	mean	+2
0,150105	6,356176	40,378479	23,036133	398253	Media orig.	—
0,239063	5,958856	35,450863	22,857216	381995	Media orig.	—
0,263007	5,922536	35,007423	22,867968	381899	Media orig.	—
0,306332	5,917839	34,927410	22,867968	381320	Media orig.	—

Tabella 5.2: Dati misurazioni filtraggio

calibrazione di ogni Kinect. Per l'intero ragionamento si farà riferimento ad un singolo apparecchio, sia per semplicità che per evitare ripetizioni dato che la procedura è la medesima per tutti.

Con la calibrazione si vuole ottenere la posizione relativa di ogni Kinect rispetto ad un dato sistema di riferimento. In questo modo sarà poi possibile "intrecciare" i dati provenienti dalle varie fotocamere e ricostruire il mondo nel visualizzatore 3D. Per fare ciò è necessario avere un oggetto di una forma ben precisa da usare come riferimento, detto **target di calibrazione**. In Figura 5.4 è riportato il target di calibrazione adottato.

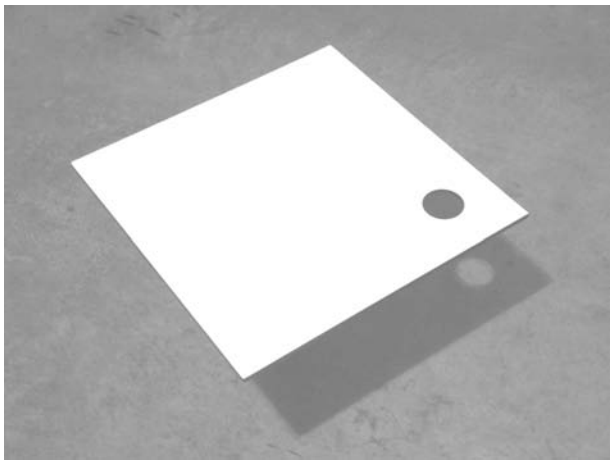
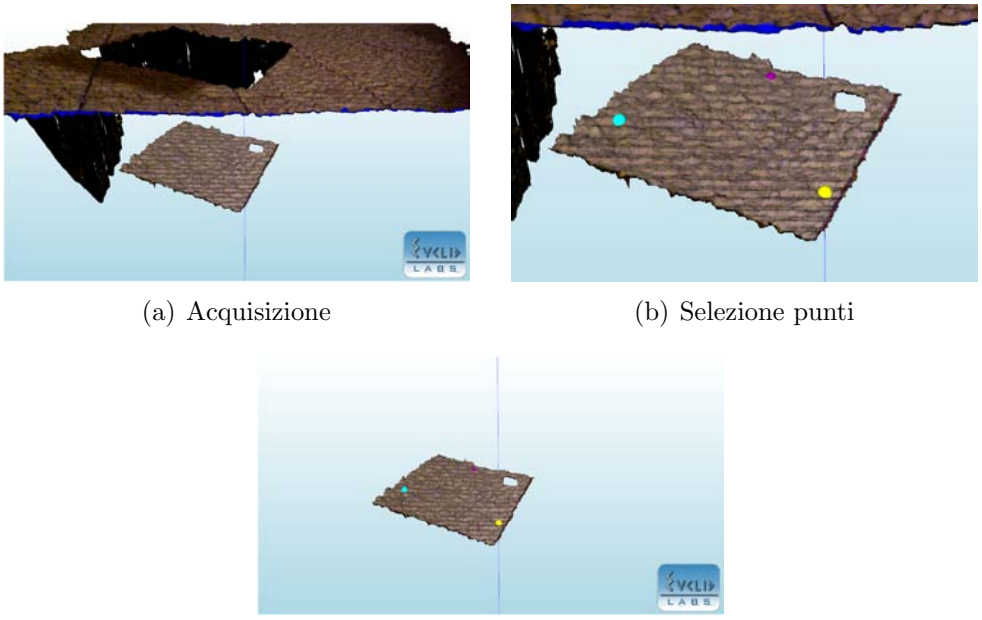


Figura 5.4: Target di calibrazione

Quello che interessa ora è riconoscere il target e la sua orientazione. Per isolare il target era stato inizialmente seguito un accorgimento già sfruttato in un progetto di prova da Euclid Labs: posizionare il target su di un piedistallo e sfruttare l'informazione sulla profondità per tagliare tutti i punti situati oltre una certa distanza.
Dopo pochi test si è scoperto che questo sistema ha due falle. La prima riguarda il vincolo di dover inquadrare il target da una direzione prossima alla perpendicolare al piano individuato dal quadrato. In caso contrario c'è la possibilità che outliers non vengano scartati e che la procedura di calibrazione venga falsata. Questo è restrittivo al fine di avere una buona ricostruzione, ovvero riprendere un oggetto da diverse angolazioni. La seconda invece riguarda il punto con distanza minima da Kinect. Se il punto a distanza minima non appartiene al target è praticamente assicurata l'errata identificazione di quest'ultimo. Dunque se si lavora in spazi ristretti non è contemplabile il fatto di non aver altri oggetti che possano interferire. Ciò è inverosimile, dato che lo scopo dei dati ricostruiti è elaborare comandi per

un robot. Robot che di solito deve aver a disposizione ceste da cui prelevare o depositare pezzi e altri macchinari di vario tipo.

Si è deciso dunque di procedere in modo differente operando come segue. Si esegue una prima acquisizione della scena e, dall'ambiente 3D, si selezionano manualmente 3 punti (non allineati) che giacciono sul quadrato di calibrazione. Con questi punti viene fittato un piano usando RANSAC. In questo modo è possibile scartare tutti i punti che stanno oltre una certa distanza dal piano e oltre un certa distanza dal baricentro dei tre punti selezionati. Si ottiene di fatto il ritaglio del target. In Figura 5.5 sono raffigurate le fasi appena descritte. Dalla selezione manuale deriva lo svantaggio di non avere una procedura completamente automatica. Dato che l'operazione di calibrazione viene eseguita una sola volta per ogni Kinect all'installazione dell'impianto, salvo guasti o necessità particolari, è stato ritenuto un compromesso accettabile. Il vantaggio sta nell'avere tutti i punti corrispondenti al target senza perdita di informazione[6], con al massimo qualche punto spurio che non rappresenta fonte di errori.



(a) Acquisizione

(b) Selezione punti

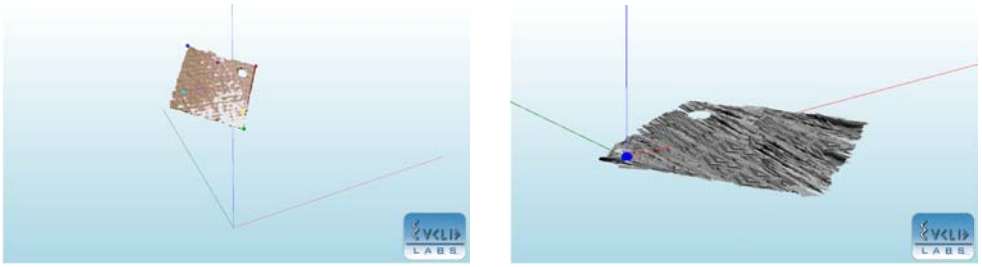
(c) Ritaglio

Figura 5.5: Ritaglio target di calibrazione

[6]La perdita di informazione è rappresentata dall'eliminazione di punti appartenenti al target, fatto che portava al fallimento la procedura precedente.

A questo punto si è pensato di spostare il target nell'ambiente 3D in modo che l'angolo più vicino al foro sia posizionato nell'origine della terna di riferimento. I lati adiacenti a quest'angolo vengono fatti coincidere con le direzioni x e y della terna di riferimento. La sequenza di rototraslazioni per arrivare alla posizione finale, composta in modo opportuno, si può inglobare in un'unica matrice. Questa matrice è a tutti gli effetti la matrice di calibrazione del Kinect considerato.

La prima operazione di questa sequenza consiste nel portare il target sul piano xy del mondo 3D, Figura 5.6. In questo modo, trascurando la componente rumorosa, i dati utili giaceranno sul tale piano.

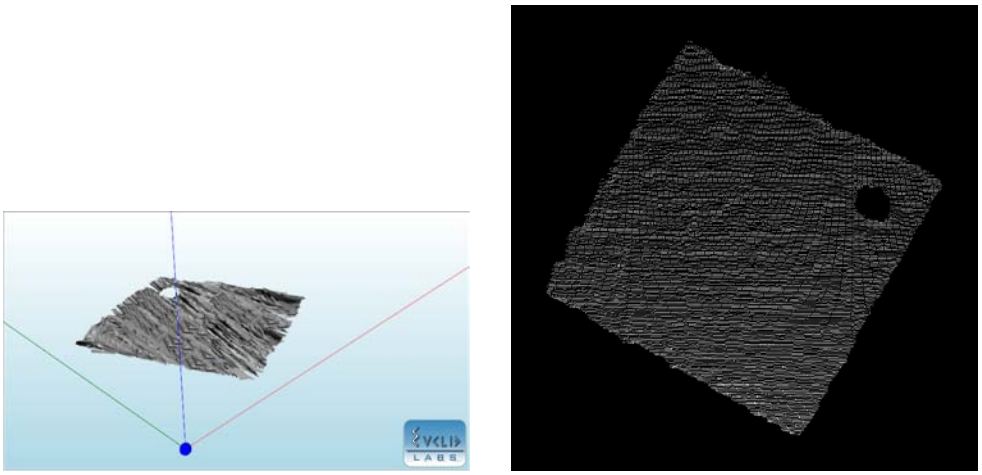


(a) Dopo il ritaglio (b) Sul piano $xy0$

Figura 5.6: Spostamento del target sul piano $xy0$ dell'ambiente 3D

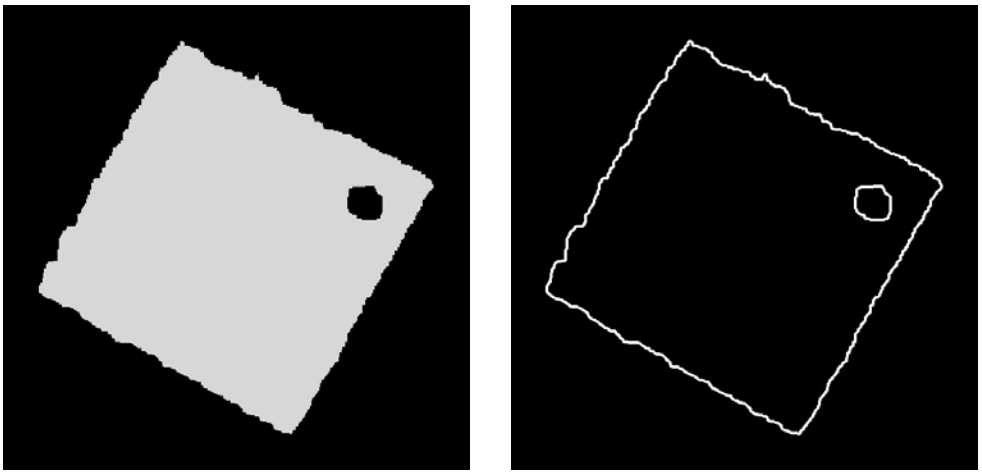
Si opera poi una semplice traslazione in modo da avere tutte le coordinate x, y dei punti positive. L'avere coordinate positive è di importanza fondamentale in quanto è possibile semplificare il problema ragionando in binario: 1 presenza di dati, 0 assenza di dati. Riportando il tutto su di una Bitmap si ottiene lo sfondo nero ed una nuvola di pixel bianchi dove sono presenti i dati. In questo modo, oltre a poter visualizzare ed interpretare più facilmente i risultati, è più semplice eseguire le successive elaborazioni, Figura 5.7.

Per cercare i bordi del quadrato, da passare poi alla trasformata di Hough, si ricorre all'edge detector formulato da Canny. Affinché questo algoritmo funzioni in modo efficace è necessario modificare la bitmap ottenuta in precedenza. Il quadrato infatti è rappresentato da una nuvola di punti, Figura 5.7(b), non da un unico blob. Si effettuano allora alcuni cicli di dilatazione seguiti da altrettanti cicli di erosione, in modo da ottenere un'unica zona bianca senza discontinuità, Figura 5.8(a). É con questa operazione che gli eventuali punti spuri vengono eliminati, come evidenziato nella Sezione 3.3.2. A questo punto si può applicare Canny, Figura 5.8. Se non si effettuasse l'operazione di chiusura, l'algoritmo di Canny restituirebbe il contorno dei singoli pixel bianchi e non il bordo del target.



(a) Passaggio a coordinate positive (b) Bitmap associata

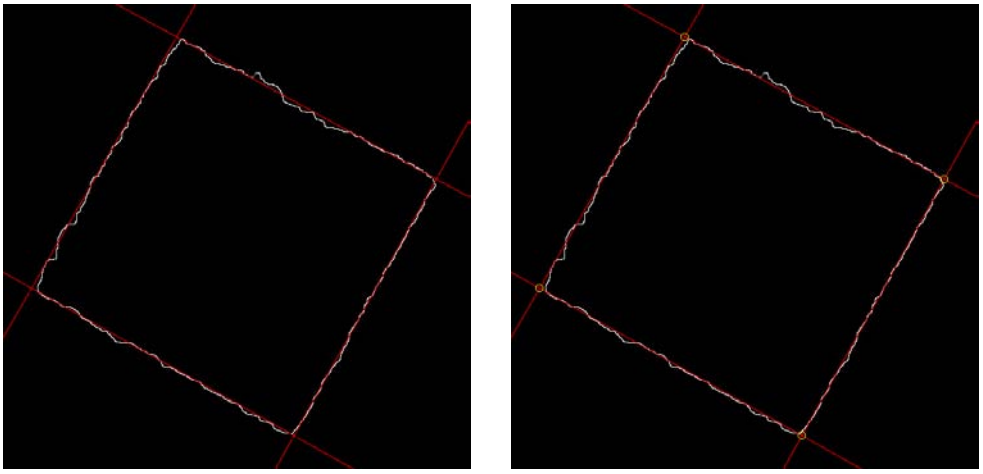
Figura 5.7: Spostamento del target in coordinate positive e passaggio in Bitmap



(a) Dilatazione ed erosione (b) Canny edge detector

Figura 5.8: Dilatazione-erosione per consentire l'edge detection

Come si può notare da Figura 5.8(b) gli angoli non sono ben definiti. Per avere un posizionamento più preciso, si vanno ad identificare i lati del quadrato mediante la trasformata di Hough per poi ricavare gli angoli dalle loro intersezioni, Figura 5.9. Per migliorare la stima della posizione degli angoli, si applicano più volte la trasformata di Hough e il calcolo delle intersezioni per poi mediare i vari risultati ottenuti.



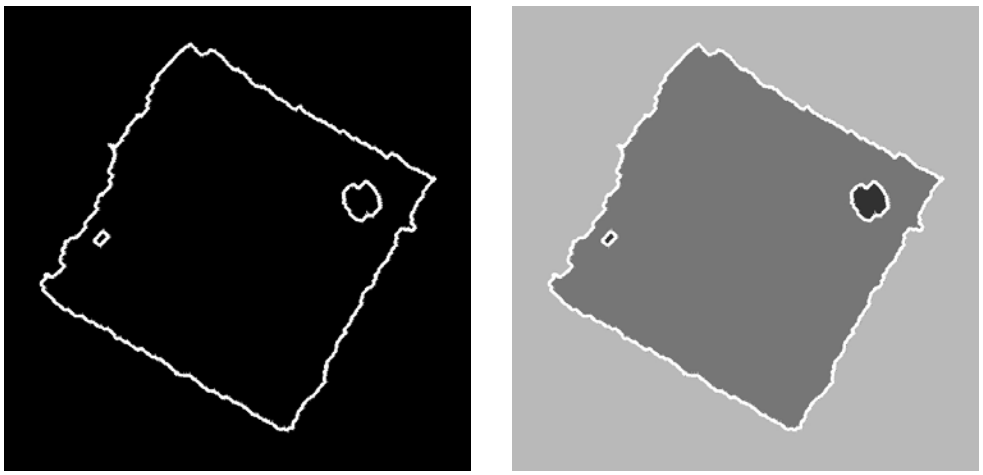
(a) Rette identificate con Hough (b) Angoli ricavati dalle intersezioni

Figura 5.9: Risultati ottenuti con la trasformata di Hough

Rimane un ultimo passo da compiere prima di effettuare lo spostamento finale. Affinché ci sia una relazione fra le visuali dei vari Kinect è necessario che lo stesso angolo sia portato nell'origine, non un angolo a caso. Come già anticipato sopra, si sceglie di portare sull'origine del sistema d'assi l'angolo più vicino al foro. Si tratta dunque di trovare una strategia per individuare la posizione del baricentro del cerchio.
Inizialmente erano state misurate fisicamente la distanza del centro dall'angolo e le dimensioni del cerchio. Una volta tradotte le misure da millimetri a pixel, si è cercato un punto nelle prossimità degli angoli a cui corrispondesse il colore nero nella bitmap[7]. In questo modo l'identificazione dell'angolo si concludeva cercando quale fra i quattro avesse distanza minore dal punto del cerchio trovato. É stato riscontrato dopo alcuni test che, cambiando la posizione di Kinect rispetto al target, il metodo appena esposto falliva aumentando la distanza. A causa dell'accuratezza del dispositivo che diminuisce con la distanza, si vengono a creare dei buchi fra i pixel acquisiti che

[7]In realtà non è necessario conoscere le coordinate del baricentro. Grazie alla posizione del foro è sufficiente trovare un punto che gli appartenga.

l'operazione di dilatazione non è in grado di chiudere perché troppo grandi, Figura 5.10(a). Utilizzando maschere di dimensione maggiore il risultato migliorava perché si riuscivano a chiudere buchi che prima restavano nell'immagine. Di pari passo però veniva diminuita l'estensione del cerchio col pericolo che si occludesse, cosa che non deve succedere. Si è scelto allora di procedere con la ricerca di tutti i possibili blob presenti nella bitmap, Figura 5.10(b). Mediante il conteggio del numero di pixel che va a formare ogni blob, viene individuato quello corretto. In generale al cerchio corrisponde il blob più esteso. Può però accadere che piccoli buchi vicini fra loro, a seguito delle operazioni di dilatazione ed erosione, si trovino a far parte ad un unico grande blob. Per evitare che questo falsi la corretta determinazione, del cerchio sono state eseguite varie misure per determinare il valore massimo di pixel che si può ottenere per il blob relativo al foro. Dunque se ci dovesse essere questo tipo di degenerazione sarebbe possibile comunque arrivare al risultato corretto.

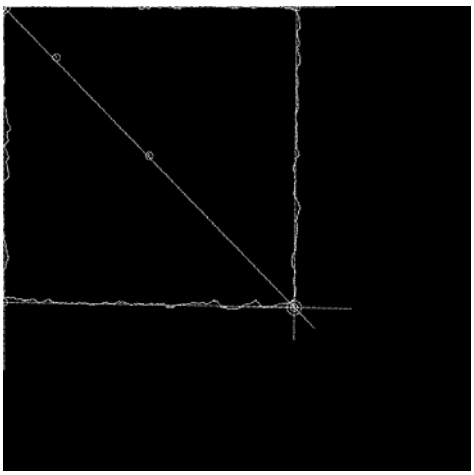


(a) Presenza di fori fittizi (b) Ricerca del cerchio con i blob

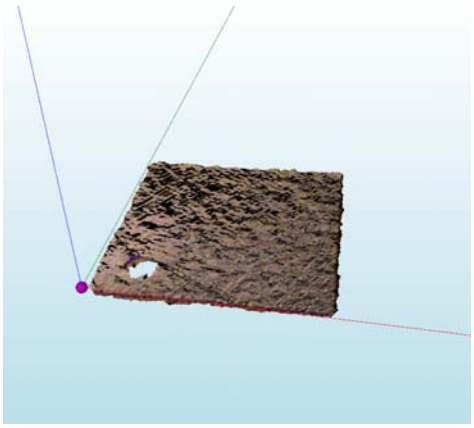
Figura 5.10: Identificazione della posizione del foro attraverso i blob

A questo punto, dai coefficienti angolari delle rette che approssimano i lati si ricava la rotazione mentre dalle coordinate dell'angolo appena individuato la traslazione. In questo modo si arriva a concludere la calibrazione, Figura 5.11.

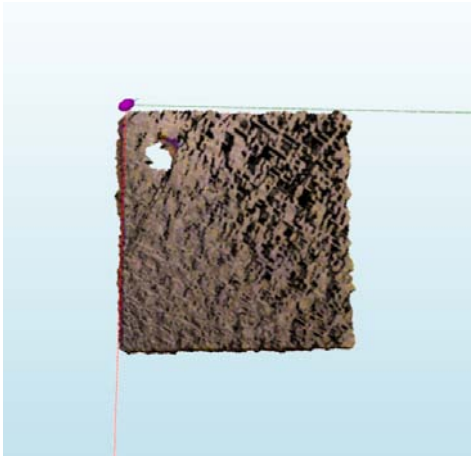
La procedura di calibrazione è strutturata in modo da poter calibrare una singola Kinect per volta. Così facendo, se uno degli n dispositivi utilizzati nell'impianto si guasta o viene spostato, è possibile ricalibrare solamente quello in questione e non l'intero sistema.



(a) Sulla bitmap



(b) Nell'ambiente 3D



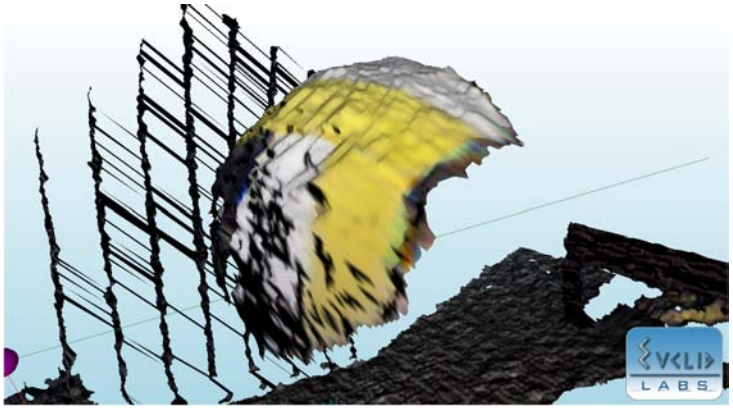
(c) Nell'ambiente 3D, vista dall'alto

Figura 5.11: Risultato finale della calibrazione

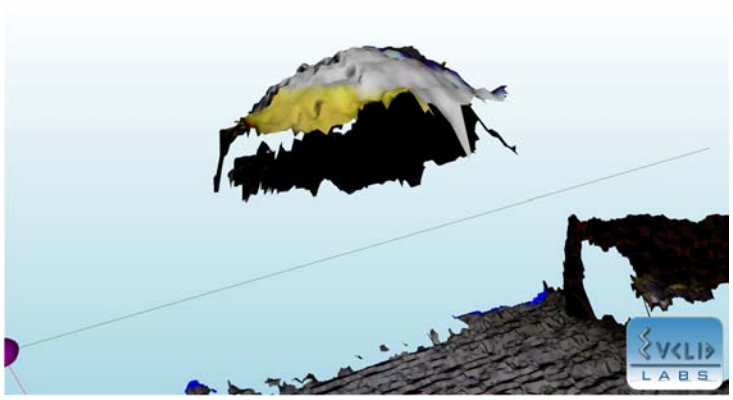
5.4 Fusione

A questo punto si deve calibrare l'intero sistema. Si posiziona il target al centro della zona di lavoro e le fotocamere tutte attorno nelle posizioni desiderate, badando di non spostare più il target fino al completamento della procedura. Ogni Kinect viene calibrata seguendo il procedimento appena illustrato. Ora che le matrici di calibrazione sono state salvate, è possibile procedere con la fusione dei dati provenienti dagli n dispositivi. Acquisendo dunque un'immagine per ogni Kinect è possibile visualizzare nell'ambiente 3D il risultato finale. In Figura 5.12 sono riportate due immagini acquisite singolarmente dove l'oggetto da ricostruire è una palla, un classico oggetto di test per questo tipo di operazioni per via della sua forma. In Figura 5.13 è stata riportata la fusione delle immagini di Figura 5.12. Come si può notare dalle varie prospettive proposte, i vari spicchi del pallone risultano coincidere fra loro.

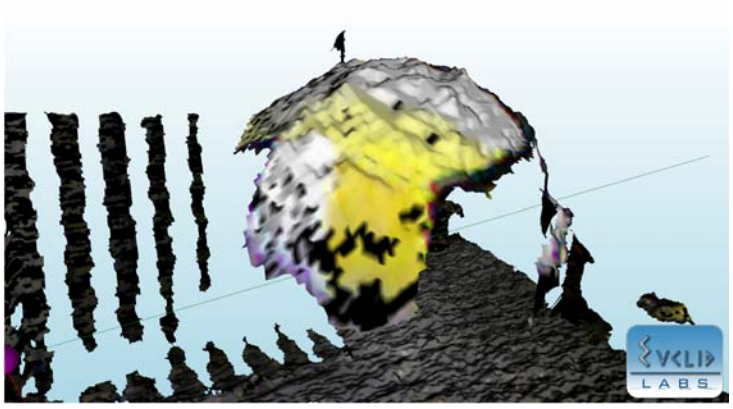
In Figura 5.14 è riportato un esempio di applicazione futura di questo progetto. Si tratta di una cassa contenente alberi motore che dovranno essere prelevati in modo automatico da un robot per alimentare le successive stazioni di lavorazione.



(a) Kinect 1



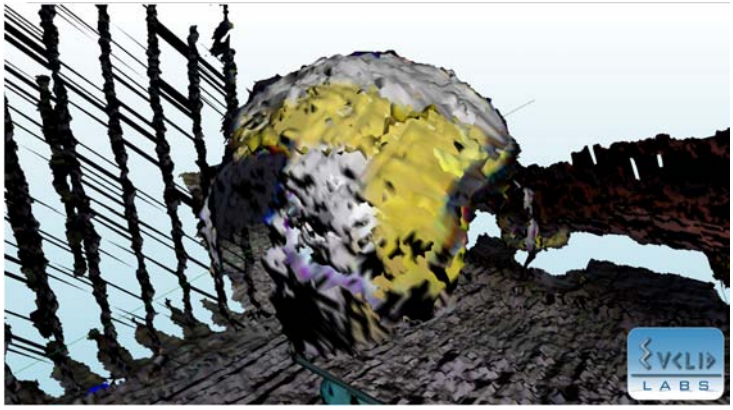
(b) Kinect 2



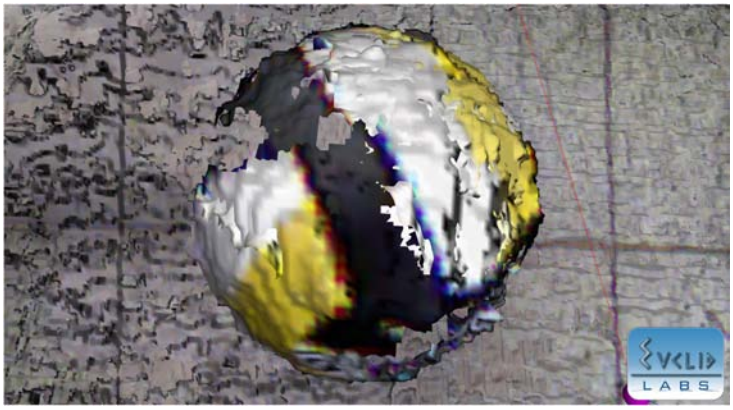
(c) Kinect 3

Figura 5.12: Acquisizioni della scena dalle varie angolazioni

(a) Immagine 2D da macchina fotografica



(b) Immagine 3D ricostruita

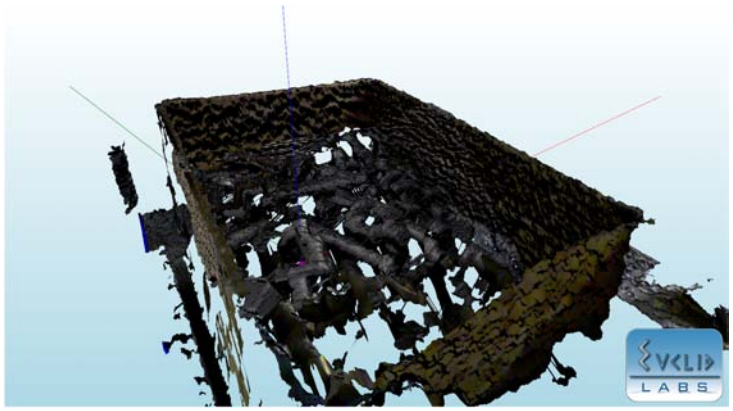


(c) Immagine 3D ricostruita

Figura 5.13: Fusione delle immagini di Figura 5.12

(a) Immagine 2D da macchina fotografica

(b) Immagine 3D ricostruita



(c) Immagine 3D ricostruita

Figura 5.14: Acquisizioni della scena dalle due angolazioni

Capitolo 6

Conclusioni

Gli obiettivi prefissati per questo lavoro di tesi sono stati raggiunti ottenendo buoni risultati. Il sistema sviluppato permette di ricostruire qualsiasi oggetto o scena. Tale ricostruzione può essere visualizzata direttamente nell'ambiente tridimensionale sviluppato da Euclid Labs.

A seguito della realizzazione è stata prevista una fase di test che ha confermato il corretto funzionamento del programma. L'unica restrizione è presentata dal range di funzionamento di Kinect. Nella fase di calibrazione bisogna dunque prestare attenzione al posizionamento dei dispositivi di acquisizione affinché il target venga acquisito completamente.

Sviluppi futuri
Avendo a disposizione altro tempo da poter dedicare a questo progetto si potrebbero migliorare i seguenti aspetti:

- In generale il filtraggio iniziale dei dati resta un punto cruciale del progetto. L'introduzione della mediana ha portato benefici ma sicuramente si può scavare ancora in questa direzione. In particolare un miglioramento applicabile sarebbe modificare il metodo per calcolare la mediana per renderlo più efficiente.

- Come brevemente accennato durante la descrizione del progetto è stato rilevato che la lente produce una distorsione dell'immagine. Questo fenomeno può essere ridotto introducendo un modello per la distorsione i cui parametri vengono individuati nella fase di calibrazione. Questo avrebbe l'effetto di migliorare ulteriormente i dati affinando sempre più la ricostruzione.

- Il progetto inerente la tesi trova compimento con la ricostruzione. Il passo successivo su cui lavorare sarebbe lo sviluppo di un programma robot per il controllo del prelievo efficace del pezzo ricostruito.
- Un aspetto fondamentale è la velocità di esecuzione. Nell'intero progetto non è mai stata analizzata con precisione in quanto lo scopo principale consisteva nel realizzare un sistema che funzionasse correttamente secondo le direttive. Nel momento in cui questo progetto dovesse entrar a far parte di un processo produttivo, i tempi di esecuzione sarebbero sicuramente un vincolo molto stringente. Già il primo punto costituisce un miglioramento da questo punto di vista e l'integrazione all'intera struttura di tecniche di multi-thread possono rappresentare una soluzione.

Appendice A

Kinect Fusion

Microsoft rilascia periodicamente toolbox con programmi di base per gli sviluppatori. A partire dalla versione 1.7 è stata introdotta l'applicazione *Kinect Fusion*. Con questo tool è possibile fare una ricostruzione del mondo spostando una singola Kinect nello spazio.

Kinect Fusion consente di scansionare degli oggetti 3D e di creare un modello usando Kinect per Windows. L'utente può disegnare la scena muovendo Kinect e simultaneamente visualizzarla e interagire con essa attraverso un dettagliato modello 3D in un apposito visualizzatore.

Questo progetto investiga tecniche per tracciare i 6 gradi di libertà relativi alla posizione di una fotocamera portatile a seguito del movimento nello spazio e produrre ricostruzioni di superfici tridimensionali ad alta qualità.

Kinect Fusion ricostruisce un modello della scena con superfici filtrate integrando i dati di profondità nel tempo a partire da diversi punti di vista. La posizione della fotocamera è tracciata in conseguenza al movimento del sensore (posizione e orientazione). Conoscendo come ogni frame è posizionato rispetto agli altri, i punti relativi a viste differenti di oggetti o di ambienti possono essere fusi in un unico volume di voxel[1] attraverso un'operazione di media. Questo volume può essere pensato come un cubo nello spazio (il volume di ricostruzione) e i dati sulla profondità (i.e. le misure della posizione delle superfici) sono integrati in questa struttura man mano che il sensore si muove.

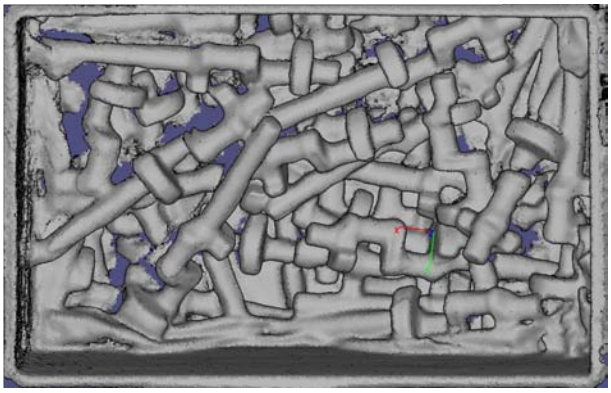
La differenza fondamentale rispetto a quanto fatto nel progetto è che la ricostruzione viene eseguita con un singolo dispositivo in movimento. Questo può essere un vantaggio in quanto si risparmia sull'hardware. É però necessa-

[1]Un voxel (volumetric pixel o più precisamente volumetric picture element) è un elemento di volume che rappresenta un valore di intensità di segnale o di colore in uno spazio tridimensionale, analogamente al pixel che rappresenta un dato di un'immagine bidimensionale.

ria la presenza di una persona o di un'apparecchiatura apposita per muovere la fotocamera. É possibile avere una ricostruzione solo se Kinect viene spostata. Per avere inoltre un riferimento nel mondo, non è necessario effettuare la calibrazione. Deve però essere previsto nella scena un particolare oggetto che svolga tale funzione. Dunque entrambe le soluzioni sono valide. In base alla specifica applicazione può essere preferibile utilizzarne una piuttosto che un'altra.

In Figura A.1 sono riportate le acquisizioni dello stesso oggetto sfruttando il procedimento della tesi nella prima e Kinect Fusion nella seconda.

(a) Metodo del progetto



(b) Kinect Fusion

Figura A.1: Fusione di immagini

Si nota che l'immagine di Figura A.1 presenta una netta riduzione del rumore rispetto a quanto fatto in precedenza, grazie alla continua integrazione dei nuovi dati con quelli già presenti nel volume di ricostruzione. Questo però a scapito dei tempi di calcolo e dell'occupazione di memoria della mesh.

Bibliografia

[1] Sito aziendale Euclid Labs. `http://www.euclidlabs.it`.

[2] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[3] Smoothing Kinect data. `http://www.codeproject.com/Articles/317974/KinectDepthSmoothing`.

[4] Elaborazione delle immagini. `http://en.wikipedia.org/wiki/Image_processing`.

[5] Image digitalizing. `http://www.engineering.uiowa.edu/~dip/lecture/ImageProperties2.html`.

[6] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *SRI International*, 1980.

[7] Andrea Fusiello. *Visione Computazionale*, 2009.

[8] P. V. C. Hough. Method and means for recognizing complex patterns. *United States Patent Office 3,069,654*, 1962.

[9] Thomas S. Huang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1979.

[10] Mediana hybrid. `http://www.librow.com/articles/article-8`.

[11] Ehsan Nadernejad, Sara Sharifzadeh, and Hamid Hassanpour. Edge detection techniques: Evaluations and comparisons. In *Applied Mathematical Sciences*, volume 2, pages 1507 – 1520, 2008.

[12] James Pawley. *Points, Pixels, and Gray Levels: Digitizing Image Data.*

[13] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision.* PWS, second edition, 1999.

[14] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision.* PWS, third edition, 2008.

[15] Informazioni su Kinect. `http://en.wikipedia.org/wiki/Kinect`.

[16] S. G. Tyan. Median filtering, deterministic properties. In *Two - Dimensional Digital Signal Prcessing II.* Verlag, 1981.