Politecnico di Milano
V Facoltà di Ingegneria
Dipartimento di Elettronica e Informazione
Corso di Laurea Specialistica in Ingegneria Elettronica


# ACQUISIZIONE DI MODELLI 3D TRAMITE SENSORI A LUCE STRUTTURATA MULTIPLI

Relatore: Prof. Luca BASCETTA
Correlatore: Ing. Simone MILANI

Tesi di Laurea di:
Daniele Ferrario
Matr. 740017

**Anno Accademico 2011-2012**

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Elenco dei codici

# Sommario

Questo lavoro di tesi analizza le prestazioni di un sistema multi sensore (MS Kinect) per l'acquisizione di modelli 3D da utilizzare in applicazioni automotive.
Sono state analizzate in particolare le prestazioni del dispositivo in diverse condizioni di illuminazione ed è stata messa a punto una metrica no reference per la classificazione dei modelli tridimensionali acquisiti dai sensori.
Inoltre, è stata proposta una strategia di correzione del rumore cross-talk, ovvero causato dall'interferenza di più dispositivi fra di loro, e una metodologia di data fusion che permette di ottenere modelli della scena tridimensionale molto più accurati e densi.

# Abstract

This thesis analyzes the performance of a multi sensor system (MS Kinect) for the acquisition of 3D models for automotive applications.
The performance of the device was analyzed in various lighting conditions. A no reference evaluation method has been developed to characterize the 3D models acquired by the sensors.
In addition, a correction strategy is proposed for the cross-talk noise, caused by interference of multiple devices to each other, and a methodology of data fusion that permits obtaining much more accurate and dense 3D models.

# Capitolo 1

# Introduzione

La tesi presentata parte dall'idea di trovare un'alternativa ai sensori tradizionali usati normalmente sui veicoli autonomi. L'idea di partenza è quella di poter usare il dispositivo Kinect, periferica per la console Xbox di Microsoft, come sensore per il rilevamento degli ostacoli.
I motivi che hanno portato a questa scelta sono essenzialmente legati al costo del dispositivo. Infatti con all'incirca un centinaio di Euro, si hanno a disposizione una serie di dispositivi (una telecamera a colori, un sensore di profondità ad infrarossi, una coppia di microfoni) che permettono una ricostruzione tridimensionale della scena acquisita. Questo dato deve essere paragonato alle migliaia di Euro che occorre spendere per i classici sensori automotive per il rilevamento di ostacoli.

Il primo aspetto che si è voluto affrontare nello svolgimento del lavoro è stato quello di creare un sistema utilizzabile per valutare le prestazioni del dispositivo in modo da avere un'idea delle possibilità di utilizzo e contemporaneamente avere un metodo valutativo relativo alle acquisizioni effettuate. Tale approccio si basa sul fatto di trovare un parametro che possa dire in un funzionamento real-time se l'acquisizione effettuata può essere utilizzata nella creazione di un modello 3D oppure se deve essere mediata con altre acquisizioni per avere dei dati che potranno essere considerati più affidabili.
Il secondo aspetto considerato in questa tesi è relativo ad un sistema costituito da due Kinect. In questa situazione si è dovuta considerare, per poterla risolvere, l'interferenza reciproca causata dagli emettitori infrarossi dei due dispositivi contemporaneamente attivi. Una volta risolto il problema dell'interferenza questo sistema è stato usato per creare un modello 3D fondendo i dati provenienti dai due dispositivi valutando il miglioramento ottenuto rispetto al caso del modello costruito con i dati di un singolo Kinect.

Questo lavoro di tesi è stato strutturato nel seguente modo:

- **Capitolo 2:** verranno analizzati i vari metodi di misura utilizzabili per individuare eventuali ostacoli calcolandone la distanza in modo da poter costruire una mappa di profondità. Inoltre si confronteranno i diversi sensori laser, attualmente presenti sul mercato, utilizzabili come radar.

- **Capitolo 3:** in questo capitolo verrà introdotto il dispositivo usato nella tesi, la telecamera Kinect di Microsoft, spiegandone il funzionamento ed il metodo con cui è stato possibile utilizzarlo interfacciato con un computer.

- **Capitolo 4:** all'interno di questo capitolo verrà analizzato il sistema costituito dal dispositivo Kinect e da una fotocamera digitale. In questa situazione si potrà analizzare il funzionamento del dispositivo in varie condizioni di illuminazione in modo da valutarne le possibilità di utilizzo. Sempre in questa situazione si analizzerà una serie di parametri che porterà ad una metrica utilizzabile per la valutazione delle prestazioni del dispositivo.

- **Capitolo 5:** verrà analizzato il sistema costituito da due Kinect concentrandosi inizialmente sull'interferenza provocata dai due emettitori infrarossi e successivamente fondendo i dati di entrambi i dispositivi per ottenere un modello 3D

# Capitolo 2

# Gestione di un veicolo autonomo

L'architettura e il software di un veicolo autonomo possono diventare molto complessi, infatti sono necessari una grande varietà di sensori per mantenere il veicolo sul percorso ed evitare gli ostacoli. Gli errori di stima dei sensori sono frequenti e la necessità di una risposta real-time richiede metodi di elaborazione veloci e precisi [1].

## 2.1 Sensori per il rilevamento di ostacoli

Per sviluppare la funzionalità di riconoscimento ostacoli in un sistema automotive è necessario introdurre sensori di distanza e prossimità. I più utilizzati sono i dispositivi di misura basati su ultrasuoni (*Sonar*) e su luce infrarossa (*IR*).

Il sonar si basa sull'emissione di un'onda sonora emessa e sull'analisi delle componenti riflesse che ritornano verso il dispositivo; calcolando il tempo di volo è possibile stimare la distanza dall'oggetto che ha riflesso l'onda sonora. I problemi relativi a questi dispositivi si riscontrano con i materiali fonoassorbenti e con superfici che riflettono il suono in una direzione diversa da quella di provenienza, tali oggetti non permettono una corretta ricostruzione delle superfici. Solitamente viene utilizzato un numero elevato di dispositivi sonar per coprire tutta l'area di interesse. Questo può provocare false rilevazioni dovute al fenomeno del cross-talk, cioè alla ricezione da parte di un sensore di un segnale generato da un altro sensore.

I sensori di distanza a infrarossi emettono un pattern luminoso al posto di impulsi sonori e hanno quindi difficoltà nello stimare la struttura di superfici fotoassorbenti (es. oggetti di colore nero).
La categoria di sensori più evoluta e precisa è quella degli scanner laser (*Laser Range Finder*, LRF). Uno scanner laser è costituito da un sensore di misura montato su una struttura rotante. Il sensore emette un fascio di luce coerente e misura

la distanza dalla superficie che riflette il fascio emesso misurando il tempo trascorso tra emissione e ricezione. Dopo ogni misura la struttura rotante compie un movimento che permette di cambiare l'orientamento del fascio laser emesso in modo da effettuare una scansione planare e conoscere le distanze rilevate ad ogni orientamento del laser. Il costo di un sensore di questo tipo è elevato ma sono sufficienti uno o due LRF per garantire la copertura della zona interessata, a differenza di un sensore che emette il fascio laser in un'unica direzione. L'angolo di scansione di questi scanner laser è solitamente di 180°, ma varia in base ai modelli presenti sul mercato.

Un altro metodo di rilevamento ostacoli è dato dall'uso della visione artificiale che comporta l'uso di telecamere e software di analisi di immagini. I costi di questa soluzione sono contenuti rispetto a quelli dei sensori laser, ma le difficoltà maggiori sono legate alla precisione degli algoritmi per il corretto riconoscimento degli ostacoli e per il rilevamento delle loro distanze.

### 2.1.1 Misurazione della distanza

Per misurare la distanza tra un punto ed una superficie si possono usare tre approcci base:

- Misura del tempo di volo (*Time-of-flight*, TOF) tra l'istante di emissione di un impulso di energia e il suo ritorno alla sorgente dopo aver subito la riflessione da parte di una superficie.

- Misura dello sfasamento (*phase-shift measurement* o *phase-detection*) tra un'onda continua emessa e quella riflessa.

- Sensori basati su radar a modulazione di frequenza, ovvero sulla rilevazione del cambio di frequenza tra l'onda emessa e quella restituita.

### 2.1.2 Misura di distanza a tempo di volo

Il tempo che intercorre tra l'emissione del segnale e la sua ricezione è pari al doppio della distanza da misurare diviso per la velocità dell'onda (Figura 2.1), da cui si ricava facilmente la distanza dalla superficie:

$$t = \frac{2d}{v} \qquad \Longrightarrow \qquad d = \frac{tv}{2} \tag{2.1}$$

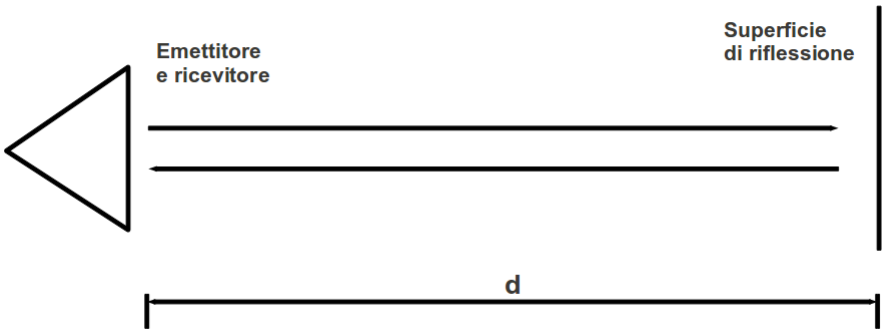


**Figura 2.1:** Principio di funzionamento di un sistema di misura della distanza a tempo di volo

dove:

- $t$: tempo misurato tra l'emissione del segnale e la sua ricezione;
- $d$: distanza da misurare;
- $v$: velocità dell'onda.

L'impulso emesso può essere un ultrasuono, un'onda radio o un impulso luminoso; a seconda del tipo di segnale emesso, la velocità di propagazione varia notevolmente: il suono in aria si propaga a circa 300 m/s, la luce a $3 \cdot 10^8$ m/s.

**Cause d'errore** Tra le cause d'errore che possono influenzare la precisione della misura di un sistema a tempo di volo bisogna considerare:

- Variazione della velocità di propagazione dell'onda.
- Errori nella rilevazione dell'istante di tempo di arrivo del segnale.
- Errori dovuti al sistema di conteggio del tempo.
- Interazione tra l'onda emessa e una superficie riflettente.

La velocità della luce si può considerare costante, mentre quella del suono è fortemente influenzata dalla temperatura dell'ambiente e dall'umidità.

Gli errori nella rilevazione dell'istante di tempo in cui il segnale di ritorno arriva al rilevatore sono causati dalla degradazione dei fronti del segnale impulsivo. Se si usa un meccanismo a soglia fissa, si identificano in maniera errata le superfici più riflettenti, in quanto il loro segnale giunge al rilevatore non degradato e viene quindi calcolata una misura inferiore a quella reale. Per risolvere questo problema si utilizzano soglie variabili in funzione del tempo trascorso dall'invio del segnale.

Il sistema di conteggio del tempo può causare errori perché occorre essere in grado di misurare intervalli inferiori ai nanosecondi, soprattutto se si desiderano misure precise su distanze brevi.

Quando un'onda incontra una superficie, parte di essa viene riflessa, parte viene assorbita e, in base al materiale che costituisce la superficie, una terza parte può attraversare la superficie stessa. In base al tipo di materiale e all'angolo di incidenza l'onda riflessa può essere degradata al punto da non essere riconosciuta dal sensore. In questo caso la distanza misurata risulta erroneamente infinita, cioè non viene evidenziata la presenza di alcun oggetto.

### 2.1.3 Misura di distanza con *phase-detection*

Un raggio laser con la potenza ottica modulata sinusoidalmente, è inviato verso l'oggetto di cui si vuole conoscere la distanza.

$$P(t) = P_0 \left[1 + m \cdot \sin\left(2\pi f_{mod} t\right)\right] \tag{2.2}$$

dove:

- $P(t)$: potenza ottica modulata sinusoidalmente;
- $P_0$: potenza ottica in uscita dal laser;
- $m$: fattore di modulazione;
- $f_{mod}$: frequenza di modulazione.

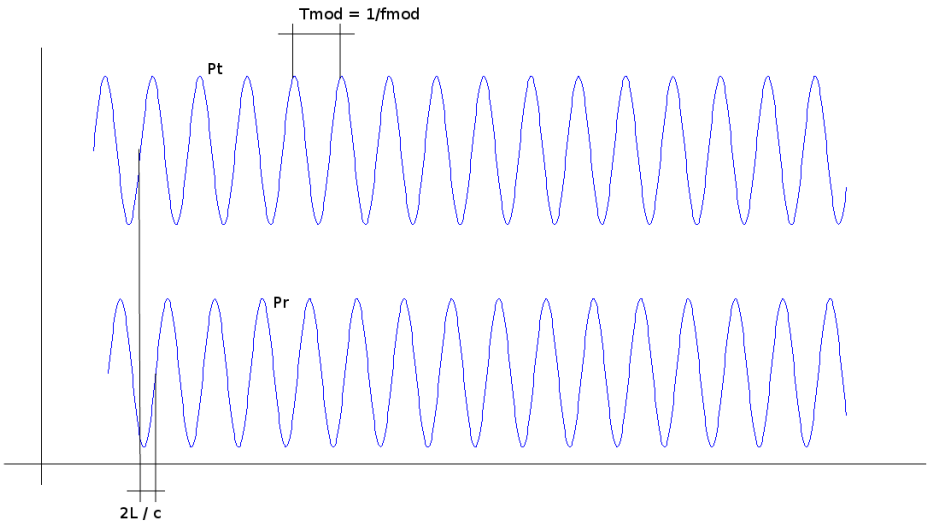


**Figura 2.2:** Sfasamento tra il segnale inviato e quello ricevuto

La luce riflessa dall'oggetto è catturata e la fase della modulazione di potenza è comparata con quella della luce inviata (Figura 2.2). Lo sfasamento ottenuto è $2\pi$ volte il tempo di volo della modulazione in frequenza.

$$\frac{\Delta\varphi}{2\pi} = \frac{\Delta T}{T_{mod}} \tag{2.3}$$

$$\Delta\varphi = 2\pi\frac{\Delta T}{T_{mod}} \tag{2.4}$$

con $\Delta T = \frac{2L}{c}$ tempo di volo.

$$L = \frac{c}{2}\frac{1}{2\pi f_{mod}}\Delta\varphi = S^{-1}\Delta\varphi \tag{2.5}$$

dove:

- $L$: distanza da misurare;
- $c$: velocità della luce;
- $f_{mod}$: frequenza di modulazione;

- $\Delta\varphi$: sfasamento misurato;

- $S$: sensibilità della misura.

La sensibilità $S$ della misura dice come varia $\Delta\varphi$ per una variazione della distanza $L$.

$$S = \frac{\delta(\text{uscita})}{\delta(\text{ingresso})} = \frac{\delta(\Delta\varphi)}{\delta(L)} = \frac{2\pi f_{mod}}{c/2} \qquad \propto \qquad f_{mod} \tag{2.6}$$

L'equazione 2.6 mostra che la sensibilità della misura cresce all'aumentare della frequenza di modulazione, ma se questa frequenza è troppo alta si avranno altri problemi relativi all'ambiguità della misura.
Infatti, essendo il segnale trasmesso, quindi anche quello ricevuto, periodico, nasce un problema di ambiguità nel riconoscere oggetti a distanza diversa che forniscono un segnale di ritorno con la stessa informazione di misura, cioè con lo stesso sfasamento accumulato.
Per non avere ambiguità deve valere:

$$\Delta\varphi_{MAX} = \varphi(L_{MAX}) = 2\pi f_{mod} T_{MAX} \leq 2\pi \qquad \Longrightarrow \qquad f_{mod} \leq \frac{1}{T_{MAX}} \tag{2.7}$$

dove $T_{MAX}$ è il massimo tempo di volo corrispondente alla massima distanza $L_{MAX}$, detta $L_{NA}$, correttamente misurabile.

$$T_{MAX} = \frac{2L_{NA}}{c} \leq \frac{1}{f_{mod}} \tag{2.8}$$

Esempio di telemetro a modulazione sinusoidale:

laser a diodo con $f_{mod} = 10\text{MHz} \div 10\text{kHz}$

da $f_{mod} = \dfrac{1}{T_{MAX}} = \dfrac{1}{2L_{NA}/c}$ si ottiene

$L_{NA} = \dfrac{c}{2} T_{MAX} = \dfrac{c}{2} \dfrac{1}{f_{mod}} = 15\text{m} \div 15\text{km}$

Si può quindi affermare che per avere una buona accuratezza sulla misura bisogna lavorare con frequenze alte ma per andare lontano si deve tenere $f_{mod}$ bassa. Per ovviare a questa situazione si può pensare di usare due distinte frequenze, $f_{mod1}$ e $f_{mod2}$.

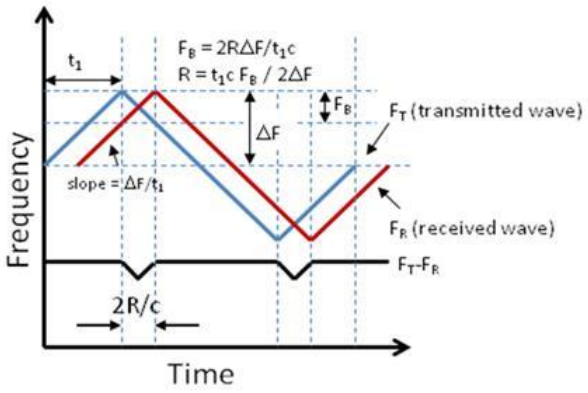


**Figura 2.3:** Forme d'onda per un radar FMCW con un trasmettitore e un obiettivo statici

Con la frequenza bassa si stabilirà in quale range di misura si trova l'oggetto, mentre con la frequenza alta si stabilirà il valore della misura con buona accuratezza senza più il problema dell'ambiguità.

### 2.1.4 Misura di distanza con radar a onda continua

Il radar a onda continua (CW) è un particolare tipo di radar che trasmette onde continue, tipicamente con andamento sinusoidale, anziché trasmettere e ricevere semplici impulsi elettromagnetici di durata limitata come nel radar classico a impulsi.
Nel caso lo spettro di trasmissione CW sia modulato in frequenza si parla di FM-CW (*Frequency-modulated continuous-wave radar*); in essi la frequenza trasmessa viene variata in funzione del tempo in modo noto. Si supponga che la frequenza aumenti linearmente col tempo; se un oggetto riflettente si trova ad una distanza $R$, il segnale d'eco tornerà dopo un tempo $T$.

$$T = \frac{2R}{c} \tag{2.9}$$

Il tempo trascorso tra la trasmissione e la ricezione assicura che il trasmettitore e il ricevitore stiano lavorando a due frequenze diverse, con il risultato che la differenza delle frequenze è direttamente proporzionale alla distanza dell'oggetto.
La figura 2.3 mostra sia le forme d'onda trasmesse (blu) che quelle ricevute (rosse) per un radar FMCW con un trasmettitore e un obiettivo statici.
$F_B$ è la differenza in frequenza tra il segnale trasmesso e quello ricevuto; dal grafico si deduce che:

$$F_B = \frac{2R\Delta F}{t_1 c} \tag{2.10}$$

dove $\frac{\Delta F}{t_1}$ è la pendenza con cui varia la frequenza del segnale trasmesso.
Ricalcolando si ottiene la distanza dal trasmettitore al target:

$$R = \frac{t_1 c F_B}{2\Delta F} \tag{2.11}$$

Le cose cambiano quando l'oggetto di cui si vuole misurare la distanza è in movimento.

**Effetto Doppler** L'effetto Doppler è un fenomeno fisico che consiste nel cambiamento apparente della frequenza di un'onda percepita da un osservatore che si trova in movimento o in quiete rispetto alla sorgente delle onde, anch'essa in movimento o in quiete. L'effetto Doppler totale può quindi derivare dal moto di entrambi, ed ognuno di essi è analizzato separatamente.
Oggi è molto facile constatare l'effetto Doppler: basta ascoltare la differenza nel suono emesso dalla sirena di un mezzo di soccorso quando si avvicina e quando si allontana. L'effetto è tanto più evidente quanto più il mezzo è veloce e quando l'oggetto o la fonte che emette il suono si trova vicino ad un osservatore.
È importante notare che la frequenza del suono emesso dalla sorgente non cambia nel sistema di riferimento solidale alla sorgente. Quello che cambia è la frequenza nel sistema di riferimento del rilevatore.
L'effetto Doppler può essere riassunto con la seguente formula:

$$\Delta f = -\frac{v}{\lambda_0} \tag{2.12}$$

dove $v$ è la velocità dell'obiettivo relativa al trasmettitore e $\lambda_0$ è la lunghezza d'onda dell'onda trasmessa.

Misurando $\Delta F_D$ in figura 2.4, si ottengono le informazioni necessarie per determinare la velocità dell'oggetto che si sta considerando.

## 2.2 Misuratori di distanza a tempo di volo laser

I sistemi di misura della distanza laser a tempo di volo fecero la loro comparsa nel 1970 presso i *Jet Propulsion Laboratory* di Pasadena (California). La precisione dei primi prototipi era di qualche centimetro per misure comprese tra 1 e 5 metri. Per permettere di misurare le distanze rilevate in più direzioni sono stati introdotti i laser scanner, il loro principio di funzionamento è molto semplice e consiste

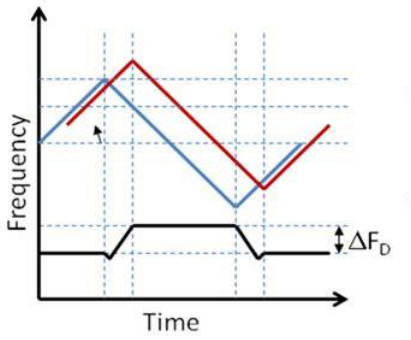


**Figura 2.4:** Forme d'onda per un radar FMCW con un obiettivo in movimento

nel montare l'emettitore laser e il ricevitore su un asse rotante. In questo modo è possibile effettuare la scansione delle distanze su un piano, solitamente con una limitazione sul movimento angolare del motore.
I sensori presenti in commercio sono numerosi e con caratteristiche differenti, per poter realizzare un confronto significativo tra i vari prodotti sarebbe necessario impostare degli esperimenti e valutare le prestazioni di ogni sensore. In questo caso si vuole effettuare solo una breve panoramica dei sistemi disponibili paragonando le caratteristiche tecniche indicate dal costruttore.
I dati significativi per il confronto sono:

- Angolo di apertura della scansione.
- Risoluzione angolare.
- Range di misura.
- Accuratezza della misura.
- Frequenza di scansione.
- Lunghezza d'onda.
- Classe di sicurezza del laser.
- Costo.

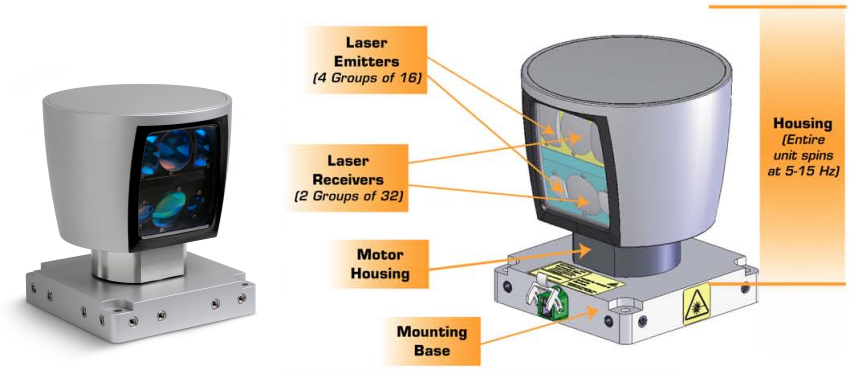


**Figura 2.5:** High Definition Lidar Velodyne (HDL-64ES2)

Nelle tabelle 2.1, 2.2 e 2.3 sono riportate le caratteristiche dei modelli analizzati. I primi due prodotti analizzati, prodotti da Velodyne, sono sensori al cui interno si trovano un numero elevato di sorgenti laser e di ricevitori, 64 nel modello HDL-64ES2 e 32 nel modello HDL-32E (Figura 2.5). Grazie a queste caratteristiche riescono a fornire una precisione ed una quantità di dati (>1.333 million points per second) difficilmente ottenibili con altri tipi di sensori. Queste caratteristiche e le conseguenti prestazioni fanno elevare il loro costo.
Tutti gli altri modelli sono sensori LIDAR veri e propri al cui interno si trova una sola sorgente laser e uno specchio che viene messo in movimento in modo da coprire il range di visione. Le loro caratteristiche sono più o meno simili e il costo varia in base ad esse.

Nella tabella 2.3 sono riportate anche le caratteristiche della luce emessa e la classe di sicurezza del laser usato dal dispositivo.

| Modello | Produttore | Range Angolare | Risoluzione Angolare |
|---|---|---|---|
| HDL-64ES2 | Velodyne | 360° | 0.4° |
| HDL-32E | Velodyne | 360° | 1.25° |
| LD-LRS | SICK | 360° | 0.062°÷1.5° |
| LD-MRS | SICK | 85°(4 layers) 110°(2 layers) | 0.125°÷0.5° |
| LD-OEM | SICK | 300° | 0.125°÷1.5° |
| URG-04LX-UG01 | HOKUYO | 240° | 0.36° |
| UTM-30LX | HOKUYO | 270° | 0.25° |
| LS4 | SIEMENS | 190° | 0.36° |
| LUX 2010 | IBEO | 85°(4 layers) 110°(2 layers) | 0.125° |

**Tabella 2.1:** Caratteristiche tecniche di alcuni scanner laser (1)

| Modello | Range di Misura | Accuratezza | Frequenza della Scansione |
|---|---|---|---|
| HDL-64ES2 | 50m÷120m | <2cm | 5÷10Hz |
| HDL-32E | 100m | <2cm | 5÷10Hz |
| LD-LRS | 250m | ±38mm | 5÷10Hz |
| LD-MRS | 250m | ±300mm | 12.5÷50Hz |
| LD-OEM | 250m | ±38mm | 5÷15Hz |
| URG-04LX-UG01 | 4m | ±30mm | 10Hz |
| UTM-30LX | 30m | ±30mm | 40Hz |
| LS4 | 50m | 5mm | 25÷40Hz |
| LUX 2010 | 200m | 10cm | 12.5/25/50Hz |

**Tabella 2.2:** Caratteristiche tecniche di alcuni scanner laser (2)

| Modello | Lunghezza d'onda | Classe del laser | Costo |
|---|---|---|---|
| HDL-64ES2 | 905nm | Class 1 | $ 85.000 |
| HDL-32E | 905nm | Class 1 | $ 42.200 |
| LD-LRS | 905nm | Class 1 | $ 14.400 |
| LD-MRS | 905nm | Class 1 | $ 14.000 |
| LD-OEM | 905nm | Class 1 | |
| URG-04LX-UG01 | 785nm | Class 1 | $ 1.200 |
| UTM-30LX | 905nm | Class 1 | $ 5.500 |
| LS4 | 905nm | Class 1 | |
| LUX 2010 | 905nm | Class 1 | |

**Tabella 2.3:** Caratteristiche tecniche di alcuni scanner laser (3)

Il significato delle classi di sicurezza è il seguente:

- **Classe 1:** intrinsecamente sicuri, si possono osservare ad occhio nudo senza subire alcun danno. I laser scanner considerati hanno tutti laser appartenenti a questa classe perché sono dispositivi che, per come sono montati, è facile vadano a colpire anche persone.

- **Classe 2:** non intrinsecamente sicuri ma non creano notevoli problemi in quanto i normali tempi di reazione dell'occhio umano fanno si che la retina non rimanga esposta per un tempo sufficiente da risultare pericoloso.

- **Classe 3A:** il laser non deve essere osservato.

- **Classe 3B:** la visione del fascio non è sicura.

- **Classe 4:** comprende i laser più potenti e pericolosi, come quelli utilizzati nel taglio dei metalli.

Da questa breve panoramica si nota che i costi dei sensori lidar tipicamente usati sono sicuramente molto maggiori di quella del dispositivo Kinect di Microsoft, che quindi risulta una valida alternativa a livello economico.

## 2.3 Esempi dei sensori utilizzati

I prototipi di vetture con guida automatica esistenti in questo momento utilizzano la seguente tecnologia:

- **Radar** Le vetture di fascia alta sono già equipaggiate con radar in grado di tracciare gli oggetti vicini. Per esempio, il Distronic Plus Mercedes, un sistema di prevenzione degli incidenti, ha dei sensori sul paraurti posteriore che innescano un allarme quando rilevano la presenza di un oggetto nell'angolo cieco della vettura.

- **Mantenimento della corsia** Videocamere montate sul parabrezza sanno riconoscere i confini delle corsie, registrando il contrasto tra la superficie stradale e le linee di demarcazione. Se il veicolo esce involontariamente dalla corsia, il conducente viene avvisato per mezzo di leggere vibrazioni del


**Figura 2.6:** Night View di Mercedes

volante.

- **LIDAR** Con 64 laser che girano a oltre 900 rotazioni al minuto, il sistema Lidar di Google genera una point cloud che offre all'auto una visione a 360°.

- **Videocamera a infrarossi** Il sistema di assistenza Night View di Mercedes usa due fanali per irradiare un raggio a infrarossi invisibile e non riflettente sulla strada dinanzi a sé. Una videocamera montata sul parabrezza rileva la segnatura IR e mostra sul display del cruscotto l'immagine illuminata, con i pericoli in evidenza (Figura 2.6).

- **Visione stereo** Un prototipo di Mercedes usa due videocamere sul parabrezza per costruire un'immagine 3D, in tempo reale, della strada e degli eventuali ostacoli.

- **GPS/Misura inerziale** Una vettura autoguidata deve sapere dove va. Google nel progetto *Google Driverless Car* utilizza un sistema di posizionamento di Applanix, insieme alla propria tecnologia di mappe e Gps.

- **Wheel Encoder** Sensori sulle ruote misurano la velocità dell'auto del progetto di Google mentre procede nel traffico.

# Capitolo 3

# Kinect come sensore di profondità

Il dispositivo Kinect (inizialmente conosciuto con il nome Project Natal) è una periferica di gioco per la console XBOX 360 di Microsoft, realizzata inizialmente come semplice dispositivo di controllo dei movimenti per applicazioni video ludiche. Tuttavia, grazie al costo contenuto e alle diverse possibilità di applicazioni offerte, la periferica sta attualmente riscontrando successo anche in altri ambiti. L'idea è quella di utilizzare il dispositivo Kinect come sensore di profondità in un sistema automotive.

## 3.1 Storia

Kinect è stato annunciato al pubblico il 1 giugno 2009 durante la conferenza stampa della Microsoft all'E3 2009 (*Electronic Entertainment Expo*) con il nome Project Natal, poi rinominato Kinect alla presentazione ufficiale all'E3 2010. La periferica è in vendita dal 4 novembre 2010 in America e dal 10 novembre in Europa, ed è possibile usarlo su un qualsiasi modello di XBOX 360.


**Figura 3.1:** Microsoft Kinect


**Figura 3.2:** Posizionamento dei componenti sul dispositivo

L'hardware del Kinect si basa su tecnologie della 3DV, una compagnia israeliana specializzata in tecnologie di riconoscimento dei movimenti tramite videocamere digitali che Microsoft ha prima finanziato e poi acquisito nel 2009, e sul lavoro della israeliana PrimeSense, che ha poi dato in licenza la tecnologia a Microsoft. Il software del Kinect è stato, invece, sviluppato internamente ai Microsoft Game Studios.

La NPD Group, società specializzata in inchieste di mercato, ha rivelato che Microsoft ha venduto due milioni e cinquecentomila Kinect nei primi venticinque giorni di commercializzazione. Microsoft ha confermato che in data 31 dicembre 2010 risultavano vendute 8 milioni di unità quindi ben al di là delle previsioni dell'azienda che contava di vendere un totale di 5 milioni di unità entro il 2011. Il 10 gennaio 2012 risultavano venduti 18 milioni di unità.

## 3.2 Hardware

Il Kinect [4] è una periferica dotata di telecamera RGB, sensore di profondità a raggi infrarossi composto da un proiettore e da una telecamera sensibile alla stessa banda (Figure 3.2 e 3.3). La telecamera RGB ha una risoluzione di 640x480 pixel, mentre quella a infrarossi usa una matrice di 320x240 pixel.

Il dispositivo dispone anche di un array di microfoni utilizzato dal sistema per la calibrazione dell'ambiente in cui si trova. In questo modo, attraverso l'analisi della riflessione del suono sulle pareti e sull'arredamento, è possibile eliminare i rumori di fondo ed il riverbero causato dai suoni del gioco in moda da riconoscere correttamente i comandi vocali.

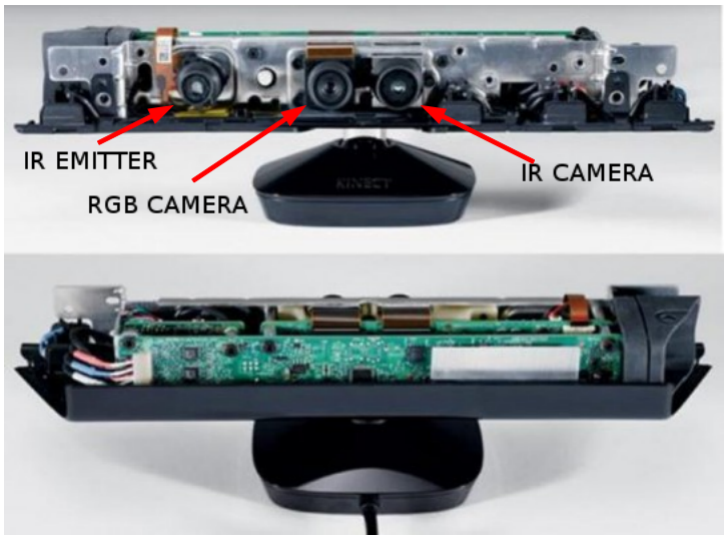


**Figura 3.3:** Kinect senza copertura

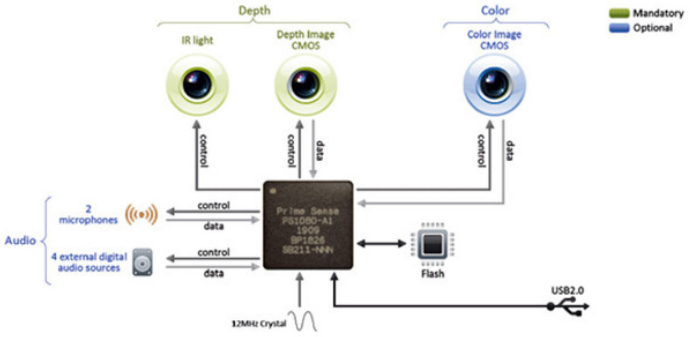


**Figura 3.4:** Schema generale di funzionamento del sensore Kinect

La barra del Kinect è motorizzata lungo l'asse verticale e segue i movimenti dei giocatori, orientandosi nella posizione migliore per il riconoscimento dei movimenti.

La periferica permette all'utente di interagire con la console senza l'uso di alcun controller da impugnare, ma solo attraverso i movimenti del corpo, i comandi vocali o attraverso gli oggetti presenti nell'ambiente. Quello che ne deriva è una vera e propria rappresentazione 3D dell'ambiente in tempo reale. In figura 3.4 si può osservare il principio di funzionamento del sensore.

Nella tabella 3.1 sono riassunte le caratteristiche generali del dispositivo Kinect.

Per alimentare la periferica servono ben 12 W mentre le porte USB sono in grado

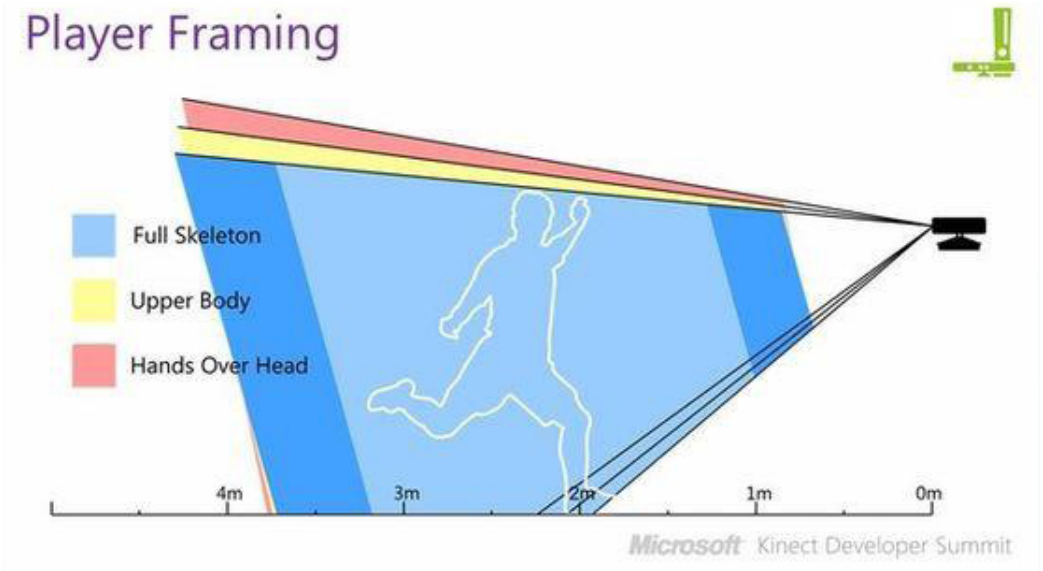


**Figura 3.5:** Raffigurazione della profondità di gioco

| | |
|---|---|
| Sensore | - Camere sensibili al colore e all'infrarosso<br>- Microfono<br>- Tilt motor per permettere alla periferica di spostarsi<br>- Compatibile con tutte le console XBOX 360 |
| Campo visivo (Figura 3.5) | - Orizzontale: 57°<br>- Verticale: 43°<br>- Capacità di inclinazione: 27°<br>- Profondità: 1.2 m ÷ 3.5 m |
| Trasferimento dati | - 320x240 16-bit profondità @ 30 frames/sec<br>- 640x480 32-bit colore @ frames/sec<br>- 16-bit audio @ 16 kHz |
| Tracking dello scheletro | - Fino a 6 persone, inclusi 2 giocatori attivi<br>- Fino a 20 movimenti per ogni giocatore attivo<br>- Riconoscimento dell'Avatar dell'utente |
| Audio | - Possibilità di effettuare le *Chat Party*<br>- Cancellazione dell'eco<br>- Riconoscimento di più voci |

**Tabella 3.1:** Caratteristiche generali del Kinect

di fornire in media 2.5 W di potenza, quindi il Kinect necessita anche di un cavo di alimentazione supplementare. Questo particolare è molto importante se si volesse montare il dispositivo su un robot mobile.


**Figura 3.6:** Disassemblaggio del Kinect

La scheda madre del Kinect (scheda A nella Figura 3.7) ha 6 chip. Da sinistra a destra sono rispettivamente montati:

- Stereo ADC con microfono preamplificato (Wolfson Microelectronics WM8737G)
- N-Channel PowerTrench MOSFET (Fairchild Semiconductor FDS8984)
- Controller USB 2.0 hub (NEC uPD720114)
- SPI flash H1026567 XBOX1001 X851716-005 Gepp
- SoC per il controller dell'interfaccia della camera a colori (Marvell AP102)
- SDRAM DDR2 512 megabit (Hynix H5PS5162FF)

Nella scheda B (Figura 3.7) sono montati:

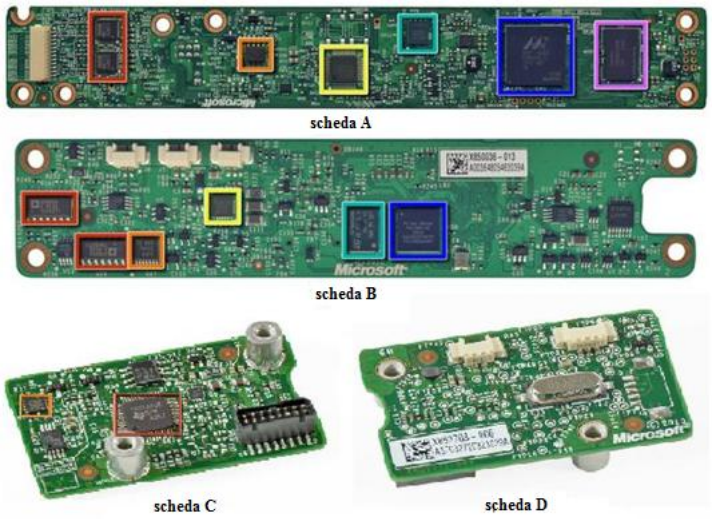


**Figura 3.7:** Componenti interni del sensore Kinect

- Due CMOS Rail-to-Rail amplificatore d'uscita (Analog Devices AD8694)

- Un campionatore e convertitore A/D 8 bit ad 8 canali, con interfaccia I2C (TI ADS7830I)

- Allegro Microsystems A3906, Low Voltage Stepper e Single/Dual DC Motor Driver

- Una memoria Flash 1Mb x 8 oppure 512Kb x 16 (ST Microelectronics M29W800DB)

- Un processore d'immagini SoC Sensor (PrimeSense PS1080-A2)

Le schede C e D (Figura 3.7) mostrano i due lati della stessa scheda che dispone di un controller audio USB frontale e centrale (TI TAS1020B) ed infine sul lato sinistro della scheda un accelerometro (Kionix MEMS KXSD9).

## 3.3 Cenni di stereovisione

La stereovisione è una tecnica di ottica inversa che permette di ottenere un'informazione di profondità da una coppia di immagini provenienti da due fotocamere

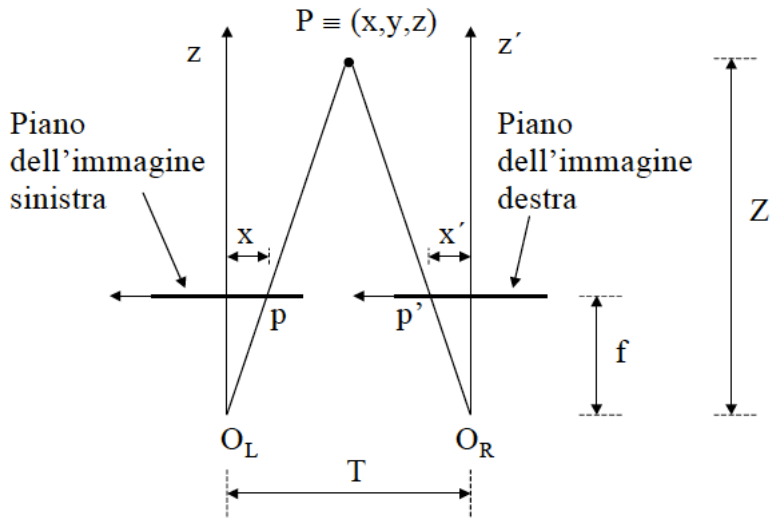


**Figura 3.8:** Geometria stereo per due fotocamere

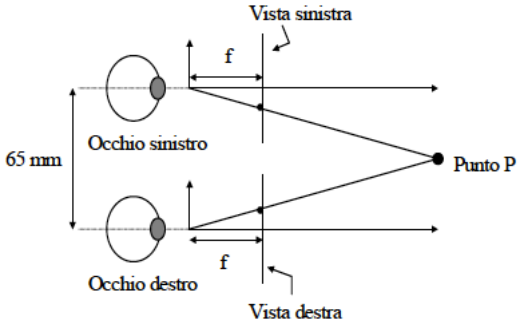


**Figura 3.9:** Sistema visivo umano

che inquadrano la stessa scena da due differenti posizioni. Un singolo punto fisico nello spazio tridimensionale della scena viene fatto corrispondere univocamente ad un pixel di ciascuna immagine.

Avendo a disposizione le due immagini prodotte dalle due fotocamere, localizzando la coppia di pixel $[p, p']$ che corrispondono allo stesso punto $P$ nello spazio 3D (Figura 3.8), è possibile ricavare le coordinate $(x, y, z)$ del punto tridimensionale $P$.

Per analisi stereo si intende il processo di misurazione di distanza di un oggetto, basato sul confronto di due o più immagini dell'oggetto stesso acquisite simultaneamente. Questo processo si rifà alla percezione della terza dimensione che avvertiamo con i nostri occhi, deriva appunto dal fatto che i due bulbi oculari forniscano due immagini leggermente diverse della scena che osserviamo (Figura

3.9). Queste immagini, sommando i loro effetti, producono il senso di profondità.

Il problema principale in questo processo è quello di trovare la corrispondenza tra gli elementi delle varie immagini disponibili. Una volta che questa corrispondenza è stata trovata, la distanza dall'oggetto può essere calcolata tramite la semplice ottica geometrica.
Le tecniche di correlazione sono una parte importante dell'analisi stereo e cercano proprio di trovare punti corrispondenti fra le diverse immagini massimizzando una qualche misura di similarità.

## 3.4 Funzionamento della periferica

Il Kinect deduce la posizione del corpo dell'utilizzatore mediante un processo che si compone dei seguenti due passi:

1. viene costruita una mappa di profondità mediante l'utilizzo di luce strutturata creata dall'emettitore a infrarossi;

2. viene dedotta la posizione del corpo utilizzando algoritmi di tracking implementati nel software sviluppato da Microsoft.

### 3.4.1 Mappa di profondità

In un sistema di stereovisione, una mappa di profondità può essere vista come una matrice $M$ di interi di dimensioni $w$ x $h$, dove $w$ e $h$ sono rispettivamente la larghezza e l'altezza dell'immagine. Ogni intero $d$ presente nelle celle della matrice rappresenta la distanza (espressa in pixel) tra il pixel di riferimento $p$ ed il corrispondente pixel $p'$, appartenente alla seconda immagine, nel momento in cui le due immagini vengono sovrapposte. Più è alto il valore della differenza $d$, più il punto tridimensionale, rappresentato univocamente dalla coppia di pixel $[p, p']$, si trova vicino alle due camere. Quanto appena affermato è vero solo a condizione che gli obiettivi delle due camere siano allineati e lievemente distanziati (Figura 3.10).
Definiamo ora come $a_1$ e $a_2$ i punti proiezione di $A$ rispettivamente sulla prima e sulla seconda immagine; analogamente $b_1$ e $b_2$ per il punto $B$. Assunto che tutti i punti $a_1$, $a_2$, $b_1$ e $b_2$ abbiano lo stesso valore di ordinata, allora tutti i punti giacciono tutti sulla medesima linea orizzontale.
Si calcolano i valori della differenza $d$ per i punti $A$ e $B$ nel seguente modo:

$$d(A) = a_1 - a_2$$
$$d(B) = b_1 - b_2$$

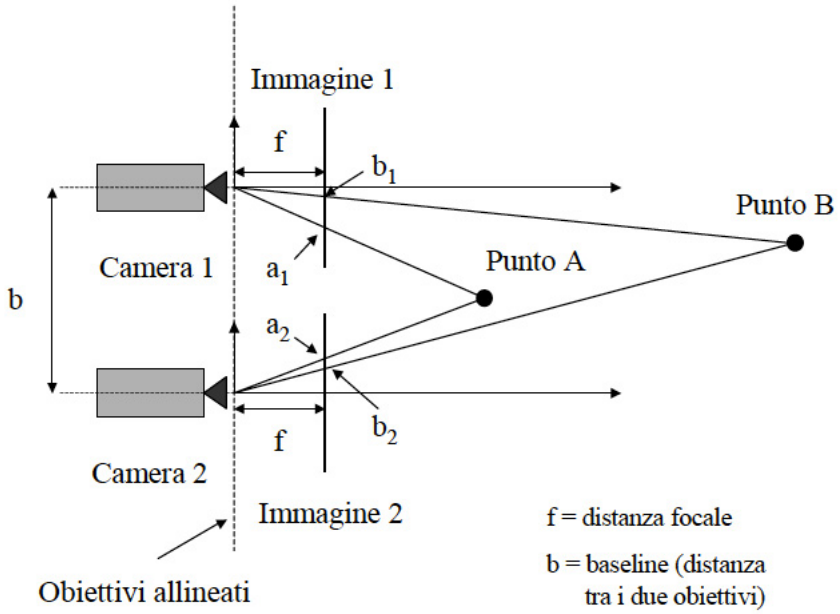


**Figura 3.10:** Sistema di camere con obiettivi allineati

come si poteva immaginare, si trova che $d(A) > d(B)$ essendo $a_1 > b_1$ e $a_2 < b_2$.
È importante sottolineare che bisogna considerare soltanto i valori di ascissa dei quattro punti, in conseguenza dell'ipotesi di allineamento verticale (Figura 3.11). In altre parole, se si considerano le due immagini come matrici di pixel, i punti $a_1$ e $a_2$ possono essere rappresentati in base ai rispettivi valori di riga e di colonna. Il pixel $a_1$ avrà coordinate $(i, j)$ mentre $a_2$ si troverà in posizione $(i', j')$.
La differenza $d$ tra $a_1$ e $a_2$ sarà semplicemente la distanza tra le posizioni $(i, j)$ e $(i', j')$ quantificata mediante un intero che indichi la differenza in pixel tra le due posizioni della coppia di punti. Naturalmente la differenza $d$ può essere calcolata solo per punti della scena che sono visibili in entrambe le immagini; un punto visibile in un'immagine ma non nell'altra si dice essere occluso.
Una volta che la matrice $M$ è stata completamente riempita, ovvero quando per ogni pixel dell'immagine di riferimento è stato trovato il corrispettivo pixel nella seconda immagine ed è stata calcolata la differenza tra i due, si è ottenuto una mappa di profondità della scena che si sta analizzando. Una mappa di profondità può essere vista come una matrice bidimensionale in cui i valori sono riconducibili alla distanza del punto proiettato dal suo corrispettivo reale.
È possibile visualizzare una mappa di profondità come un'immagine dove il valore intero $d$ corrisponde all'intesità luminosa del pixel. In questo modo si ottiene un'immagine su scala di grigi della profondità degli oggetti della scena.
Si può dedurre che la mappa di profondità può essere utilizzata come strumento per la valutazione tridimensionale dell'ambiente osservato.

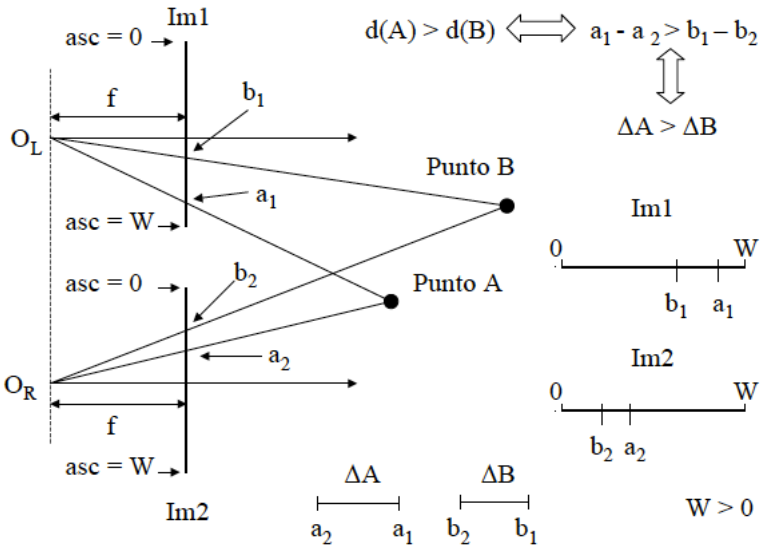


**Figura 3.11:** Calcolo della differenza $d$

Il Kinect ottiene informazioni 3D dalla scena analizzata creando appunto al suo interno una mappa di profondità. Normalmente questa mappa si ottiene tramite un sistema di stereovisione, ma il Kinect non è un sistema di questo tipo essendo dotato solo di una Color Camera, una Depth Camera e di un emettitore di raggi infrarossi.

La soluzione adottata è la seguente: l'emettitore ad infrarossi proietta nell'ambiente una grande quantità di spot luminosi la cui distribuzione a prima vista sembra casuale. Il pattern emesso è visibile spegnendo le luci e inquadrando l'ambiente con una camera digitale (Figura 3.12).

Passare da tale pattern all'informazione 3D è un'operazione matematica ben definita: la disposizione dei punti del pattern è nota al software del Kinect, che al suo interno ha memorizzato come il pattern dovrebbe essere, visto dal sensore IR, se fosse proiettato su una superficie posta ad una distanza ben definita e perfettamente parallela al piano della Depth Camera; questo rappresenta il pattern di riferimento. Per ognuno dei punti proiettati si può calcolare la distanza dal sensore triangolando la sua posizione attuale con quella che avrebbe assunto nel pattern di riferimento, in maniera del tutto simile a quello che viene fatto in un sistema di sterovisione.

Risultato dell'elaborazione del pattern proiettato è un'immagine di profondità grezza, nel senso che i valori di profondità di un punto non sono valori espressi in una qualche unità di misura. È dunque necessaria una procedura di calibrazione che metta in corrispondenza i valori grezzi con valori espressi in scala metrica.

Per riconoscere quale punto si sta analizzando viene considerata (procedura di correlazione) la disposizione dei 64 punti vicini che per ogni punto del pattern è

**Figura 3.12:** Pattern infrarosso emesso dal Kinect

diversa e ben definita. I punti più grossi e luminosi visibili all'interno del pattern e le zone più chiare e più scure servono a facilitare questa operazione di riconoscimento.
Quando gli spot luminosi proiettati impattano sugli oggetti della scena viene a crearsi una deformazione della distanza tra uno spot e l'altro in base all'angolazione delle superfici, in quanto la fotocamera inquadra la proiezione bidimensionale della distribuzione tridimensionale degli spot. In base alla densità degli spot luminosi ci si può fare un'idea anche riguardo l'angolazione di una superficie inquadrata.

La figura 3.13 mostra graficamente quanto spiegato precedentemente, visualizzando come i sensori ottici diano ognuno il proprio contributo fornendo al chip PrimeSense PS1080-A2 i dati necessari alla realizzazione di un'immagine contenente informazioni di profondità relative alla scena osservata; tale immagine contiene anche un certo numero di informazioni relative alla distorsione dei punti rispetto alla loro posizione ideale.
Ecco quindi spiegato in che modo il Kinect determina la distanza degli oggetti nella scena e la loro conformazione.


**Figura 3.13:** Schema della procedura di creazione dell'immagine di depth

### 3.4.2 Tracking

Ottenuta un'immagine di depth della scena osservata, il successivo passo di elaborazione consiste in un'operazione di tracking interamente a carico del software; questa operazione consiste nell'identificazione del numero, della posizione e delle giunture scheletriche degli esseri umani presenti all'interno della scena.
Il termine tracking indica la capacità di un calcolatore di riconoscere la posizione e l'orientamento di determinati oggetti attraverso l'analisi di una sequenza di immagini. Trattando il Kinect è più corretto parlare di *object tracking*, inteso come l'abilità di un calcolatore di riuscire a riconoscere oggetti e a tracciarne il movimento degli stessi attraverso l'analisi di una sequenza di immagini, al limite una sola come nel nostro caso.
Il tracking di Microsoft [5] è frutto di 500.000 campioni di dati registrati riguardanti i diversi comportamenti umani (ballo, spostamento, saluto, ecc.). Di questi, una parte (circa 10.000) sono stati considerati rilevanti e sono stati usati per insegnare al software a riconoscere la posizione dei punti chiave delle scheletro umano.

I dati del tracking vengono elaborati in tempo reale e forniti alla macchina a cui è stato abbinato il dispositivo per essere utilizzati. La figura 3.14 riassume lo schema di elaborazione delle informazioni della scena osservata con l'aggiunta del passo riguardante il tracking.
L'approccio utilizzato dal Kinect per effettuare la cosiddetta *pose recognition in parts*, ovvero la predizione delle posizioni 3D ed il tracciamento del movimento delle giunture scheletriche del corpo umano in real time partendo da una singola

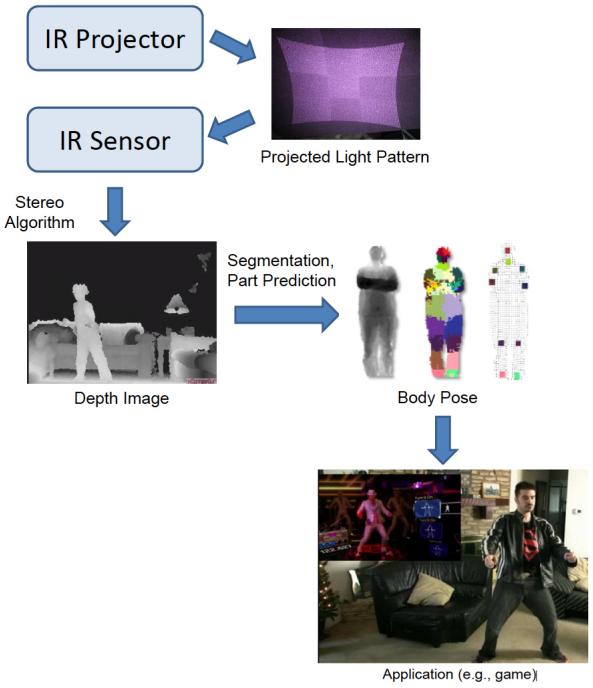


**Figura 3.14:** Schema delle operazioni di elaborazione effettuate dal Kinect sulla scena inquadrata

immagine di depth, si basa sulle moderne tecniche nell'ambito dell'*object recognition* basate sul principio di suddivisione dell'oggetto in parti. Le tecniche di riconoscimento sviluppate in questo ambito assicurano livelli soddisfacenti di efficienza computazionale e robustezza.
L'unico input al software è costituito dall'immagine di depth (in bianco e nero) prodotta dal chip PrimeSense, dalla quale si provvede ad eliminare qualsiasi elemento di sfondo al fine di isolare la fisionomia del controller umano. Su questa immagine viene adattato un modello statistico, cioè l'immagine di depth del corpo viene probabilisticamente suddivisa in parti tali da ricoprirla densamente (Figura 3.15). Alcune di queste parti sono risapute contenere dei particolari giunti scheletrici di interesse (Figura 3.16), mentre altre sono utili solo a riempire gli spazi tra le varie parti del tipo precedente o potrebbero essere usate per prevedere articolazioni aggiuntive.
Questa rappresentazione trasforma il complesso problema del tracking in uno facilmente risolvibile con opportuni algoritmi di classificazione. Ne consegue che le parti identificate dovrebbero essere abbastanza piccole da consentire di individuare con precisione i giunti scheletrici, ma non troppo piccole perché un gran numero di porzioni da analizzare andrebbe inevitabilmente a peggiorare le prestazioni del classificatore. Durante la fase di classificazione i pixel dell'immagine vengono considerati e classificati singolarmente in modo che le singole operazioni di analisi siano parallelizzabili all'interno della GPU (*Graphics Processing Unit*).

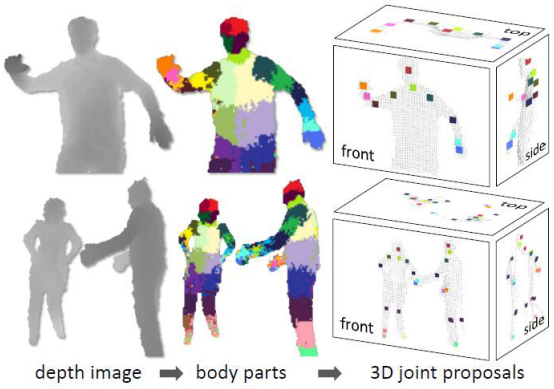


**Figura 3.15:** Fase della pose recognition a partire da una singola immagine di depth

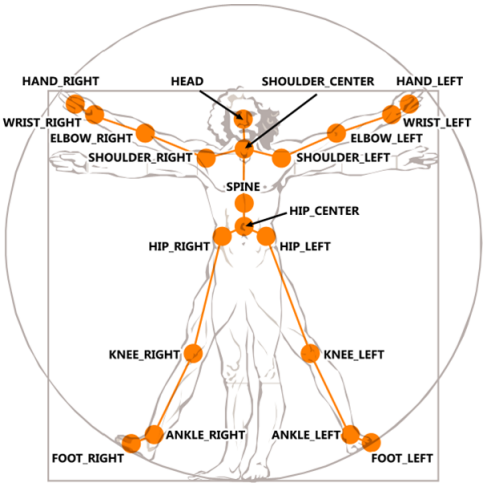


**Figura 3.16:** Schema delle giunture scheletriche ricercate dal Kinect

Il classificatore classificherà i pixel sulla base di determinate features, che nel caso in esame coinvolgono semplici confronti di profondità di offset associati singolarmente ai pixel. In particolare per ogni pixel il calcolo delle features avviene nel modo seguente:

$$f_\theta(I, x) = d_I\left(x + \frac{u}{d_I(x)}\right) - d_I\left(x + \frac{v}{d_I(x)}\right) \tag{3.1}$$

dove:

- $d_I(x)$ è la profondità del pixel $x$ nell'immagine $I$;

- $\theta(u, v)$ descrive gli offset $u$ e $v$.

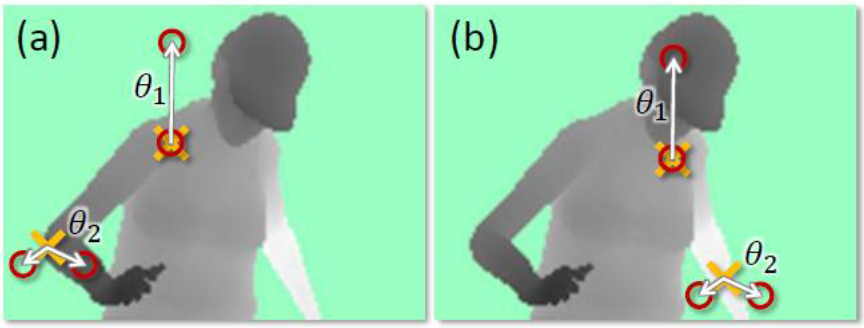


**Figura 3.17:** Features dell'immagine di depth

La normalizzazione dell'offset $\left(\frac{1}{d_I(x)}\right)$ garantisce invarianza rispetto alla profondità, ciò significa che per un dato punto sul corpo un offset fissato nello spazio determinerà se il pixel è vicino o lontano dalla camera. Se un pixel $x'$ di offset si trova sul fondo o al di fuori dei limiti dell'immagine, il calcolo della profondità $d_I(x')$ restituirà un valore grande positivo.

Ad esempio, nell'immagine 3.17 le croci gialle rappresentano i pixel sui quali viene operata la classificazione mentre i cerchi rossi denotano invece i pixel di offset. Sulla base di quanto detto, in (a) il calcolo dell'equazione 3.1 per le due features $\theta_1$ e $\theta_2$ restituirà valori grandi in conseguenza della grande differenza di profondità stimata per i due offset associati al pixel. In (b) invece per le stesse features si otterrà un valore molto minore, questo perché nel caso di $\theta_1$ non ci sono pixel di offset che si trovano al di fuori dei limiti dell'immagine, mentre nel caso di $\theta_2$ perché tutti i pixel di offset si trovano al di fuori dei limiti dell'immagine. Se ne deduce che la feature $\theta_1$ potrebbe essere utilizzata per identificare a che altezza del corpo è localizzato il pixel, mentre la feature $\theta_2$ è utile per trovare sottili strutture verticali quali ad esempio le braccia.

La scelta delle features utilizzate è fortemente condizionata dalla necessità di garantire un'elevata efficienza computazionale: ognuna di esse necessita la lettura di massimo tre pixel dell'immagine e lo svolgimento di al più cinque operazioni matematiche.

Individualmente queste features non forniscono un'indicazione significativa circa la parte del corpo cui un pixel appartiene, ma se combinata in una foresta decisionale risultano sufficienti ad eliminare ogni ambiguità. Foreste e alberi decisionali randomizzati sono strutture di comprovata velocità ed efficacia per la risoluzione di problemi di classificazione e sono implementabili efficacemente su una GPU.

Una foresta non è altro che un insieme di $T$ alberi decisionali, ognuno dei quali rappresenta un classificatore che predice la probabilità di un pixel di appartenere a ciascuna delle parti. Ad ogni nodo di un albero corrisponde una coppia formata da una feature $f_\theta$ e da un valore di soglia $\tau$. Per classificare un pixel $x$ di un'immagine $I$ si valuta ripetutamente l'equazione 3.1 iniziando dal nodo radice di ogni

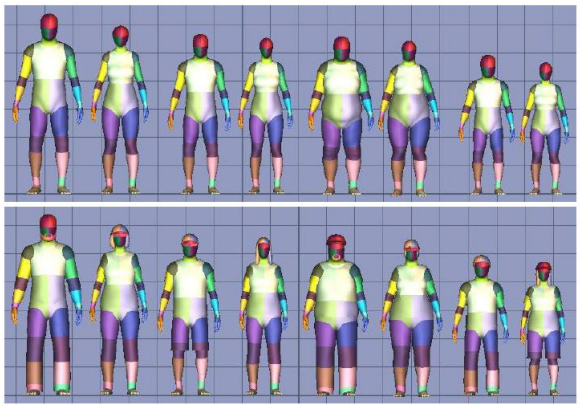

**Figura 3.18:** Esempi di body models precaricati nel motion capture database del Kinect

albero e seguendo le diramazioni sottostanti sulla base dei confronti con i valori di $\tau$ indicati nei nodi. Così facendo si giunge ad uno dei nodi foglia, in ognuno dei quali è memorizzato una distribuzione di probabilità relativa all'appartenenza del pixel considerato alle varie parti del corpo etichettate. Per derivare la classificazione finale viene calcolata una media tra tutte le distribuzioni di probabilità ricavate navigando i $T$ alberi decisionali.
Ogni albero viene addestrato utilizzando un diverso numero di immagine scelte a caso; un sottoinsieme casuale di 2.000 pixel viene scelto in ogni immagine dell'insieme per garantire una distribuzione uniforme su tutte le parti del corpo.
In altre parole, per ogni nodo viene scelto un insieme di features che vengono testate singolarmente per capire quale risulta più discriminante; quest'ultima viene selezionata ed assegnata al nodo. Il processo di selezione viene ripetuto ricorsivamente su ogni nodo che non sia un nodo foglia, utilizzando solo i pixel di esempio presenti nel nodo considerato, cioè su un sottoinsieme dell'iniziale insieme di pixel di esempio ricavato prendendo 2.000 pixel da ogni immagine dell'insieme di training selezionato per l'albero in questione. Il processo ricorsivo si blocca nel momento in cui un nodo contiene un numero troppo basso di pixel di esempio oppure si è raggiunta una prefissata profondità massima.
Per il training del software sono state generate un gran numero di immagini realistiche di depth che rappresentano esseri umani di varia stazza e dimensioni posti in varie pose. Queste immagini sono state raccolte in un motion capture database per poter essere utilizzate nelle comparazioni con quelle ottenute di volta in volta nell'analisi della scena osservata. Il database contiene i modelli 3D di una ventina di body models (Figura 3.18) e circa 500.000 frame che li vedono impegnati in alcune centinaia di scene di guida, ballo, corsa, navigazione nei menù, ecc. Viene lasciato alla componente di classificazione software il compito di riconoscere altre eventuali possibili pose generalizzando quelle già conosciute.

Il risultato della classificazione dei pixel è una *texture map* sulla quale sono ben

evidenziate le parti e che può essere utilizzata per lo skin dei personaggi ad ogni rendering. Riproiettando le parti su più piani (corrispondenti a diversi punti di vista) si cerca di calcolarne le modalità di distribuzione spaziale al fine di produrre una proposta della posizione tridimensionale delle giunture scheletriche di interesse (Figura 3.16).
Il Kinect effettua il riconoscimento delle parti del corpo deducendo le informazioni dall'analisi dei singoli pixel di un'immagine di depth, queste informazioni vengono poi aggregate per essere considerate nel loro insieme al fine di valutare la posizione delle giunture. In particolare viene usato un approccio locale basato sull'algoritmo *mean shift*, cioè un algoritmo per la ricerca di massimi locali. L'idea sulla quale si basa l'algoritmo consiste nel trovare all'interno della finestra di ricerca, centrata in un determinato punto $x$, un nuovo punto $x'$ che sia più vicino, rispetto a $x$, al massimo locale ed iterare questa procedura finchè non si raggiunge il massimo locale.
Si definisce un estimatore della densità di una parte del corpo come:

$$f_c(\widehat{x}) \propto \sum_{i=1}^{N} w_{ic}\ exp\left(-\left\|\frac{\widehat{x}-\widehat{x}_i}{b_c}\right\|^2\right) \tag{3.2}$$

dove:

- $\widehat{x}$ è una coordinata nello spazio 3D;
- $N$ è il numero di pixel dell'immagine;
- $w_{ic}$ è un peso assegnato al pixel;
- $\widehat{x}_i$ è la proiezione del pixel dell'immagine nello spazio data la profondità $d_I(x_i)$;
- $b_c$ è una larghezza di banda dedotta per la parte di corpo considerata.

La quantità $w_{ic}$ tiene conto sia della probabilità che il pixel appartenga ad una certa parte, sia della superficie del pixel stesso:

$$w_{ic} = P(c\backslash I, x_i) \cdot d_I(x_i)^2 \tag{3.3}$$

Questo assicura che le stime di densità siano invarianti rispetto alla profondità, il

che produce un significativo miglioramento dell'accuratezza nella predizione della posizione delle giunture.
Secondo quanto affermato da Microsoft, un'implementazione ottimizzata di questo algoritmo di tracking permette l'analisi di un frame in 5ms utilizzando la GPU in dotazione sulla Xbox360. Questo tempo di elaborazione è almeno di un ordine di grandezza più veloce rispetto a quello di tutti gli altri algoritmi di tracking attualmente utilizzati.

La forza della soluzione adottata da Microsoft per il body tracking deriva dall'aver trattato il problema della *pose estimation* come uno di object recognition utilizzando una rappresentazione delle parti del corpo studiata per la localizzazione spaziale dei giunti scheletrici con un basso costo computazionale ed un alto grado di accuratezza.

## 3.5 Librerie open source per il dispositivo

Per poter lavorare con il dispositivo di Microsoft si è fatto affidamento al lavoro svolto dal progetto OpenKinect (*http://openkinect.org*). OpenKinect è una comunità aperta di persone interessate ad utilizzare l'hardware Xbox Kinect su PC ed altri dispositivi, realizzando librerie gratuite ed open source.
L'obiettivo principale del progetto è attualmente il software **libfreenect**.

Il software libfreenect, utilizzato in questa tesi, include tutto il necessario per attivare, inizializzare e comunicare i dati con l'hardware del Kinect. Questo include i drivers e un API multipiattaforma che funziona su Windows, Linux e OS X. L'API avrà associazioni ed estensioni con i seguenti linguaggi:

- C
- C++
- .NET
- Java
- Python

- C Synchronous Interface

In questa tesi tutti i programmi di interfaccia con il dispositivo sono stati scritti in linguaggio C.

## 3.6 Calibrazione del dispositivo

### 3.6.1 Modello di una fotocamera

#### Modello semplificato

Si introduce un sistema di riferimento destrorso ($X$,$Y$,$Z$) per lo spazio tridimensionale centrato in $C$ e con l'asse $Z$ coincidente con l'asse ottico. Quest'ultimo prende il nome di sistema di riferimento standard della fotocamera [6][7].

Si introduce anche un sistema di riferimento ($u$,$v$) per il piano $R$ centrato nel

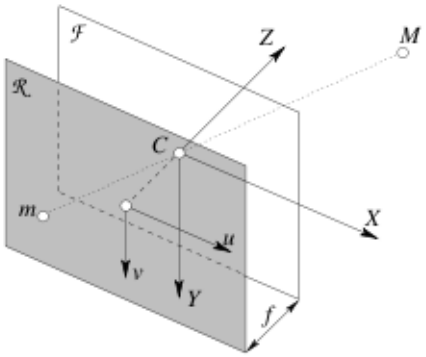


**Figura 3.19:** Modello geometrico della fotocamera

punto principale e con assi $u$ e $v$ orientati rispettivamente come $X$ ed $Y$ (Figura 3.19).
Consideriamo un punto $M$ di coordinate ($x$,$y$,$z$) nello spazio 3D e sia $m$ di coordinate ($u$,$v$) la sua proiezione sul piano $R$ attraverso $C$.

Mediante semplici considerazioni sulla similitudine dei triangoli (Figura 3.20), si può scrivere la seguente relazione:

$$\frac{f}{z} = \frac{-u}{x} = \frac{-v}{y} \tag{3.4}$$

cioè:

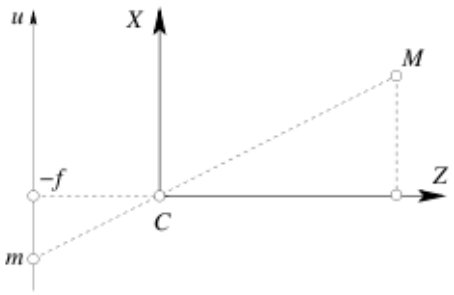


**Figura 3.20:** Vista semplificata del modello della fotocamera

$$\begin{cases} u = \dfrac{-f}{z}x \\ \\ v = \dfrac{-f}{z}y \end{cases} \tag{3.5}$$

Questa è una *proiezione prospettica*. La trasformazione dalla coordinate 3D a quelle 2D è non lineare ma può divenire lineare considerando le coordinate omogenee.

Invece di rappresentare i punti nel piano tramite coppie di coordinate $(x,y)$, si possono rappresentare con terne di coordinate omogenee $(u,v,w)$ collegate alle precedenti dalle relazioni $x = \dfrac{u}{w}$ e $y = \dfrac{v}{w}$.
Le terne proporzionali rappresentano lo stesso punto, per questo motivo si chiamano omogenee. Quando $w \neq 0$ la terna rappresenta un punto effettivo, mentre se $w = 0$ la terna rappresenta un punto all'infinito; la terna (0,0,0) è esclusa.

Siano

$$\mathbf{m} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \text{e} \qquad \mathbf{M} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.6}$$

le coordinate omogenee di $m$ ed $M$. Si nota che ponendo la terza coordinata ad 1, si sono esclusi i punti all'infinito. L'equazione prospettica si riscrive:

$$z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -f_x \\ -f_y \\ z \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.7}$$

Passando alla notazione matriciale:

$$z\mathbf{m} = P\mathbf{M} \tag{3.8}$$

La matrice $P$ rappresenta il modello geometrico della fotocamera e viene chiamata *matrice della fotocamera* o *matrice di proiezione prospettica*($MPP$).

### Modello generale

Un modello realistico di una fotocamera, che descriva la trasformazione da coordinate 3D a coordinate pixel, oltre che della trasformazione prospettica, deve tener conto di:

- pixelizzazione, forma e dimensione della matrice CCD e sua posizione rispetto al centro ottico.

- trasformazione rigida tra la fotocamera e la scena;

***Parametri intrinseci*** La pixelizzazione viene presa in considerazione mediante una trasformazione che tiene conto della traslazione del centro ottico e la riscalatura degli assi $u$ e $v$:

$$\begin{cases} u = k_u \dfrac{-f}{z} x + u_0 \\ v = k_v \dfrac{-f}{z} y + v_0 \end{cases} \tag{3.9}$$

dove ($u_0$,$v_0$) sono le coordinate del punto principale, $k_u$ ($k_v$) è l'inverso della dimensione efficace del pixel lungo la direzione $u$ ($v$), le sue dimensioni fisiche sono pixel/m.
La matrice di proiezione prospettica diventa:

$$P = \begin{bmatrix} -fk_u & 0 & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = K[I|0] \tag{3.10}$$

dove:

$$K = \begin{bmatrix} -fk_u & 0 & u_0 \\ 0 & -fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

Poniamo $\alpha_u = -fk_u$ e $\alpha_v = -fk_v$; si tratta della lunghezza focale espressa rispettivamente in pixel orizzontali e verticali. I *parametri intrinseci*, codificati nella matrice $K$, sono dunque i seguenti quattro: $\alpha_u$,$\alpha_v$,$u_0$,$v_0$.
Il modello più generale prevede anche un ulteriore parametro $\theta$, l'angolo tra gli assi $u$ e $v$ (normalmente $\theta = \frac{\pi}{2}$):

$$K = \begin{bmatrix} -fk_u & -fk_u \cos\theta & u_0 \\ 0 & -fk_v / \sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

***Parametri estrinseci*** Per tenere conto del fatto che, in generale, il sistema di riferimento assoluto non coincide con il sistema di riferimento standard della fotocamera, bisogna introdurre una trasformazione rigida che lega i due sistemi di riferimento.
Si introduce dunque un cambio di coordinate costituito da una rotazione $R$ seguita da una traslazione $t$, e si indica con $\mathbf{P_c}$ le coordinate di un punto nel sistema di riferimento standard della fotocamera e con $\mathbf{P}$ le coordinate dello stesso punto nel sistema di riferimento assoluto.

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.13}$$

La matrice di rotazione è ortogonale, cioè $R^T R = RR^T = I$.

$$\mathbf{P_c} = R(\mathbf{P} - t) \tag{3.14}$$

Data una matrice ortogonale di rotazione ed un vettore di traslazione, la matrice dei *parametri estrinseci* è:

$$M_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -R_1^T t \\ r_{21} & r_{22} & r_{23} & -R_2^T t \\ r_{31} & r_{32} & r_{33} & -R_3^T t \end{bmatrix} \tag{3.15}$$

Se si considera un sistema costituito da due o più fotocamere, normalmente si prenderà come sistema di riferimento assoluto quello relativo ad una fotocamera. I parametri estrinseci di questa fotocamera saranno costituiti da una matrice identità come matrice di rotazione e da un vettore di traslazione nullo.

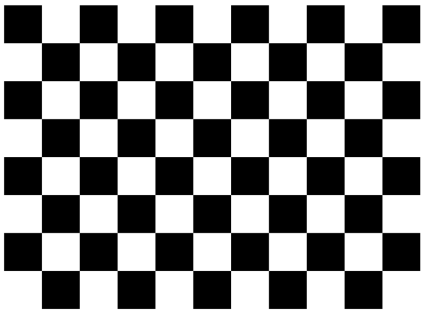

**Figura 3.21:** Scacchiera di riferimento

### 3.6.2 Calibrazione Kinect

Per ogni punto acquisito dal dispositivo come dato di profondità dalla Depth Camera è possibile ricavare la posizione 3D reale utilizzando la matrice di calibrazione della telecamera ed associare ad ogni punto il colore reale dell'oggetto presente nella scena usando l'immagine RGB acquisita dalla Color Camera del Kinect. Questa è una procedura che richiede un processo di calibrazione fra le due telecamere.
Per fare questa calibrazione è stato usato RGBDemo, un tool ideato da Nicolas Burrus, cofondatore di Manctl, una compagnia che si occupa di software per scanner 3D low-cost, *http://labs.manctl.com/rgbdemo/*.
RGBDemo è un software open source che si propone di fornire un semplice toolkit per iniziare a lavorare con i dati del Kinect e sviluppare programmi autonomi di computer vision.
Per eseguire la calibrazione devono essere eseguiti i seguenti passi:

1. Costruire un pattern di riferimento utilizzando una scacchiera (Figura 3.21) stampata su un foglio di carta e incollata su un pezzo di cartone. Non è necessario tagliare il cartone in eccesso attorno al foglio di carta.

2. Acquisire alcune immagini della scacchiera usando l'applicazione *rgbd-viewer*. Le immagini devono essere acquisite in modalità *Dual IR/RGB* (abilitando l'opzione nel menù *Capture*). Di default le immagini vengono salvate nella directories *grab1/SERIAL/view<N>* (dove *<N>* è sostituito dal numero progressivo dell'acquisizione) che contiene i file raw, *raw/color.png*, *raw/depth.raw* e *raw/intensity.raw* che corrispondono all'immagine a colori, all'immagine di depth e all'immagine IR normalizzata in scala di grigi.
   Per ottenere una calibrazione ottimale bisogna seguire alcune indicazioni:

- coprire con la scacchiera, nelle varie acquisizioni, la maggior area dell'immagine possibile, prestando attenzione alla copertura degli angoli;
- cercare di tenere la scacchiera il più vicino possibile alla fotocamera per ottenere una precisione migliore;
- per la calibrazione stereo le informazioni di depth non sono richieste, perciò è possibile coprire il proiettore IR ottenendo vicino alla camera una stima migliore dei parametri intrinseci della camera IR e dei parametri stereo. L'algoritmo di calibrazione determinerà automaticamente quali immagini possono essere usate per la calibrazione della depth;
- posizionare la scacchiera con diversi angoli;
- solitamente si acquisisce un set di 30 immagini per mediare i possibili errori;
- tipicamente l'errore di riproiezione è minore di 1 pixel. Se il valore ottenuto è significativamente più alto, significa che probabilmente la calibrazione è fallita.

3. Eseguire il programma di calibrazione:

   *build/bin/calibrate-kinect-ir –pattern-size 0.025 –input grab1/SERIAL*

   La dimensione del pattern corrisponde alla dimensione in metri di un quadrato della scacchiera. Bisognerà inserire 0.025 (25mm) per la scacchiera stampata su un foglio A4.
   Il programma genererà il file *kinect_calibration.yml* contenente i parametri per il viewer, e due file *calibration_rgb.yaml* e *calibration_depth.yaml* per la compatibilità con ROS (*Rugged Operating System*).

L'operazione di calibrazione restituisce i parametri intrinseci delle due fotocamere e le matrici di rotazione e traslazione che descrivono la posizione di una fotocamera rispetto all'altra.
Queste informazioni sono contenute nel file *kinect_calibration.yml*. La struttura dei file generati dal programma di calibrazione è visibile in appendice A.

Per ogni fotocamera si ottengono i parametri intrinseci:

$$\begin{pmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{pmatrix}$$

Dove $(f_x, f_y)$ sono le lunghezze focali, per la proiezione prospettica, mentre $(C_x, C_y)$ sono le coordinate, in pixel, del centro dell'immagine.
Questi dati sono definiti nel file come *rgb_intrinsics* e *depth_intrinsics*:

$$\text{rgb\_intrinsics} = \begin{pmatrix} 554.6184 & 0 & 304.2246 \\ 0 & 551.2557 & 270.0465 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{depth\_intrinsics} = \begin{pmatrix} 621.0659 & 0 & 317.7947 \\ 0 & 617.2708 & 235.2383 \\ 0 & 0 & 1 \end{pmatrix}$$

I parametri intrinseci sono i parametri necessari a collegare le coordinate di un pixel dell'immagine con le coordinate corrispondenti nel sistema di riferimento della camera. Per ogni pixel $(x_d, y_d)$ della depth camera può essere proiettato nello spazio 3D usando le seguenti formule:

$$\begin{aligned} x_{3D} &= (x_d - C_{x_{depth}}) \cdot depth(x_d, y_d)/f_{x_{depth}} \\ y_{3D} &= (y_d - C_{y_{depth}}) \cdot depth(x_d, y_d)/f_{y_{depth}} \\ z_{3D} &= depth(x_d, y_d) \end{aligned} \tag{3.16}$$

Oltre ai parametri intrinseci delle due fotocamere nel file troviamo anche i parametri estrinseci, cioè quelli che legano la posizione di una fotocamera rispetto all'altra; viene calcolata la matrice di rotazione e il vettore di traslazione:

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} 0.99998 & -0.00157 & -0.00460 \\ 0.00155 & 0.99998 & -0.00427 \\ 0.00460 & 0.00426 & 0.99998 \end{pmatrix}$$

$$T = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} 0.02516 \\ -0.00038 \\ -0.00113 \end{pmatrix}$$

Con queste due matrici è possibile modificare ciascun punto 3D nel sistema di riferimento della Depth Camera in modo da portarlo nel sistema di riferimento della Color Camera potendo così riproiettarlo sull'immagine a colori assegnandoli in questo modo il colore corrispondente.

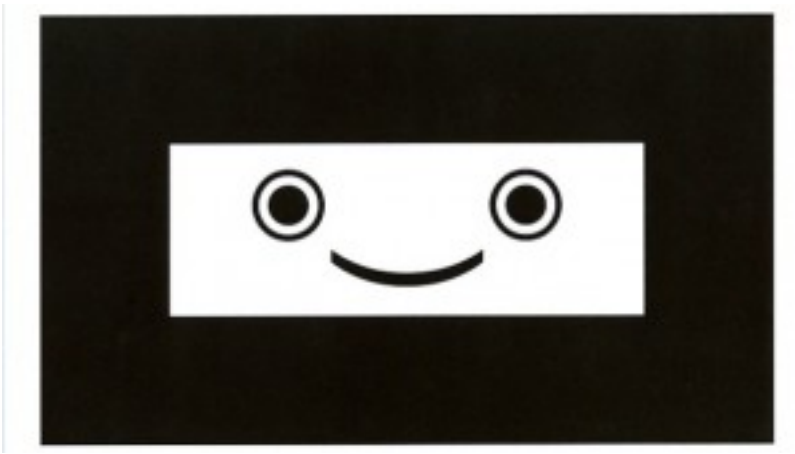

**Figura 3.22:** Microsoft Calibration Card

$$
\begin{aligned}
z &= 1.0/[depth(x_d, y_d) \cdot (-0.0030711016) + 3.3309495161] \\
x_{3D} &= (x_d - C_{x_{depth}}) \cdot z / f_{x_{depth}} \\
y_{3D} &= (y_d - C_{y_{depth}}) \cdot z / f_{y_{depth}} \\
P3D &= [x; y; z] \\
P3D' &= R' \cdot P3D - T' \\
x &= (P3D'(1) \cdot f_{x_{RGB}}) / P3D'(3) + C_{x_{RGB}} \\
y &= (P3D'(2) \cdot f_{y_{RGB}}) / P3D'(3) + C_{y_{RGB}}
\end{aligned}
\tag{3.17}
$$

Nell'equazioni 3.17 si calcolano $(x, y)$, cioè le coordinate in pixel nell'immagine a colori relative al punto $(x_d, y_d)$ dell'immagine di depth, passando attraverso lo spazio 3D. $P3D$ rappresenta un punto nel piano 3D con il riferimento alla Depth Camera mentre $P3D'$ è lo stesso punto spostato nel sistema di riferimento della Color Camera. Il valore $z$ rappresenta la profondità nel modello 3D del punto considerato calcolata in metri.

L'operazione di calibrazione deve essere eseguita per ogni dispositivo in quanto non si ha la certezza che durante il montaggio le due telecamere siano state montate esattamente con la stessa posizione reciproca, distanza e angolazione fra di esse. Chiaramente si parla di differenze minime ma che potrebbero portare ad errori nel modello 3D. Queste differenze potrebbero portare anche ad errori nel riconoscimento della persona, per questo motivo è stato creato un processo di calibrazione che usa la Microsoft Calibration Card (Figura 3.22), disponibile in ogni custodia di gioco per Kinect.

## 3.7 Acquisizione dati dal dispositivo

Per acquisire i dati dal dispositivo sono state usate due funzioni contenute nella libreria libfreenect.

**int freenect_sync_get_video(void **video, uint32_t *timestamp,**
**int index, freenect_video_format fmt);**

dove:

- video: puntatore ad un buffer video con le dimensioni richieste (640x480x3);
- timestamp: puntatore associato a timestamp;
- index: indice del dispositivo (0 se è il primo);
- fmt: formato valido.

**int freenect_sync_get_depth(void **depth, uint32_t *timestamp,**
**int index, freenect_depth_format fmt);**

dove:

- video: puntatore ad un buffer depth con le dimensioni richieste (640x480);
- timestamp: puntatore associato a timestamp;
- index: indice del dispositivo (0 se è il primo);
- fmt: formato valido.

Queste funzioni scrivono i dati della Color Camera e della Depth Camera nei buffer corrispondenti; i dati rimangono validi fino a quando le funzioni non vengono richiamate, quindi bisogna creare delle copie se i dati devono essere usati in un secondo momento. Entrambe le funzioni restituiscono un valore diverso da zero in caso di errore.
Un esempio dell'utilizzo di queste funzioni è mostrato nell'appendice B (Paragrafo B.1).

# Capitolo 4

# Kinect + fotocamera

Per valutare le prestazioni del Kinect si è pensato di acquisire una vista laterale con una fotocamera digitale aggiuntiva (Figura 4.1). Tale informazione permette di misurare la qualità del modello 3D generando una versione sintetica della vista laterale, mediante un'operazione di warping, e confrontandola con quella realmente acquisita. La similarità fra queste due immagini risulta tanto maggiore quanto accurato è il modello 3D utilizzato nel warping.
L'acquisizione del modello 3D tramite Kinect e della vista laterale avviene in maniera pseudo-simultanea, in quanto i due dispositivi vengono comandati sequenzialmente.

**Figura 4.1:** Sistema Kinect e fotocamera

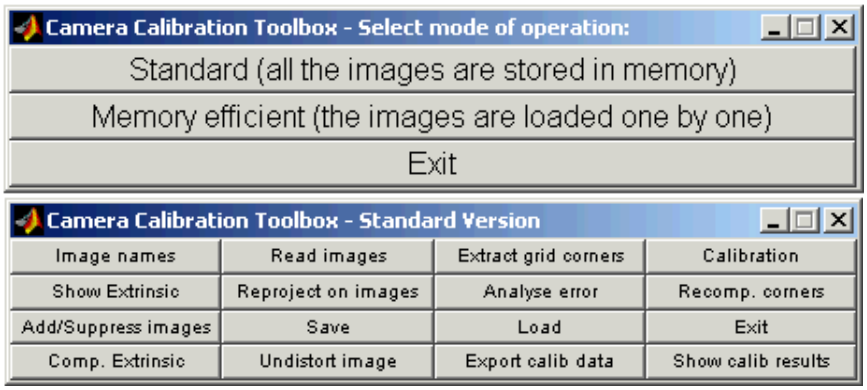


**Figura 4.2:** Interfaccia utente del Camera Calibration Toolbox for Matlab

## 4.1 Calibrazione stereo

Nei sistemi multicamera un'operazione fondamentale è data dalla calibrazione dei sensori (vedi paragrafo 3.6 a pag. 35). Tale operazione permette di collegare i punti o i pixel di immagini appartenenti ad altri sensori. In particolare, l'operazione di calibrazione ricostruisce l'equazione di geometria predittiva che definisce come punti tridimensionali della realtà vengono proiettati nelle varie immagini.
Nel lavoro svolto, è stata effettuata una calibrazione stereo cioè una procedura che lega la posizione della fotocamera rispetto alla posizione del Kinect, in particolare con la camera colore del Kinect.
Questa procedura di calibrazione è stata effettuata utilizzando un tool di Matlab, *Camera Calibration Toolbox for Matlab*, creato e reso disponibile da Jean-Yves Bouguet, disponibile sul sito internet *www.vision.caltech.edu/bouguetj/calib_doc/*.

Una volta scaricato e installato correttamente il toolbox può essere eseguito usando la funzione *calib_gui* (o *calib*) che mostrerà una finestra in cui scegliere fra i due modi di utilizzo del toolbox (Figura 4.2, in alto): standard o memory efficient. Nel metodo standard tutte le immagini usate per la calibrazione sono caricate in memoria una volta e non vengono più lette dal disco; questo minimizza il numero totale di accessi al disco ed aumenta le prestazioni del processo su tutte le immagini. Comunque, se la grandezza delle immagini è elevata o è il loro numero ad essere elevato, si può incorrere nel messaggio di errore *OUT OF MEMORY*. In questo caso la versione memory efficient del toolbox deve essere usata; in questo modo di funzionamento ogni immagine è caricata una alla volta e non viene mai salvata permanentemente nella memoria.
Entrambe le scelte portano ad una stessa finestra di scelta (Figura 4.2, in basso); selezionando l'opzione *Images names* e inserendo dove richiesto il basename delle immagini di calibrazione e il formato delle immagini stesse, tutte le immagini sono caricate in memoria nelle variabili $I_1$, $I_2$, $I_3$, ..., $I_n$. Il numero delle immagini è salvato nella variabile *n_ima*.

L'insieme del set di immagini caricate e che verranno usate per la calibrazione

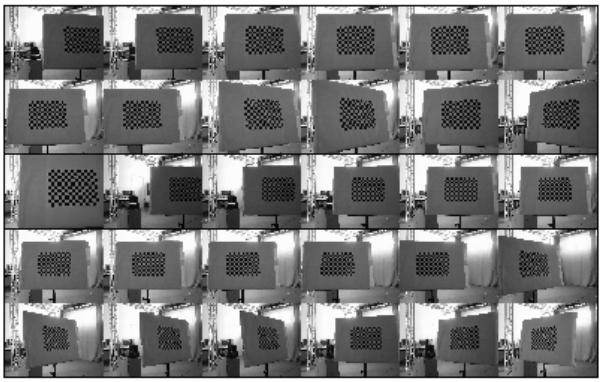

**Figura 4.3:** Immagini usate per la calibrazione

viene mostrato dal toolbox (Figura 4.3). Le immagini usate devono mostrare la medesima scacchiera di riferimento ma posta in posizioni e angolazioni diverse.
Scegliendo l'opzione *Extract grid corners* viene chiamata una funzione che permetterà di individuare l'esatta posizione della scacchiera nell'immagine, per fare ciò bisogna selezionare i quattro vertici esterni della scacchiera (Figura 4.4) per tutte le immagini e il toolbox in automatico ricaverà tutti i vertici dei quadrati che compongono la scacchiera.
Una volta completata la procedura per tutte le immagini bisogna far eseguire al toolbox la calibrazione scegliendo l'opzione *Calibration* dalla finestra di interfaccia. Questa operazione viene eseguita in maniera automatica e restituisce i parametri intrinseci della fotocamera che si sta considerando; questo risultato può essere salvato in un file, *Calib_Results.mat*, selezionando l'opzione *Save*.
Naturalmente questa operazione deve essere eseguita anche per la seconda fotocamera che stiamo considerando. Nel nostro caso deve essere eseguita la calibrazione prima con le immagini acquisite con il Kinect e successivamente con quelle acquisite dalla fotocamera digitale.

Dopo la calibrazione di entrambi i dispositivi si avranno a disposizione due file di calibrazione che vengono rinominati in *Calib_Results_left.mat* e *Calib_Results_right.mat* in base alla posizione reciproca dei due dispositivi. Questi due file servono per eseguire la calibrazione stereo dei dispositivi.

Per iniziare la calibrazione stereo si usa il comando *stereo_gui* che aprirà una finestra di interfaccia (Figura 4.5); selezionando l'opzione *Load left and right calibration files* vengono caricati i file di calibrazione ottenuti con la precedente procedura.
A questo punto selezionando l'opzione *Run stereo calibration* il toolbox calcolerà la calibrazione stereo del sistema preso in considerazione. In pratica il toolbox

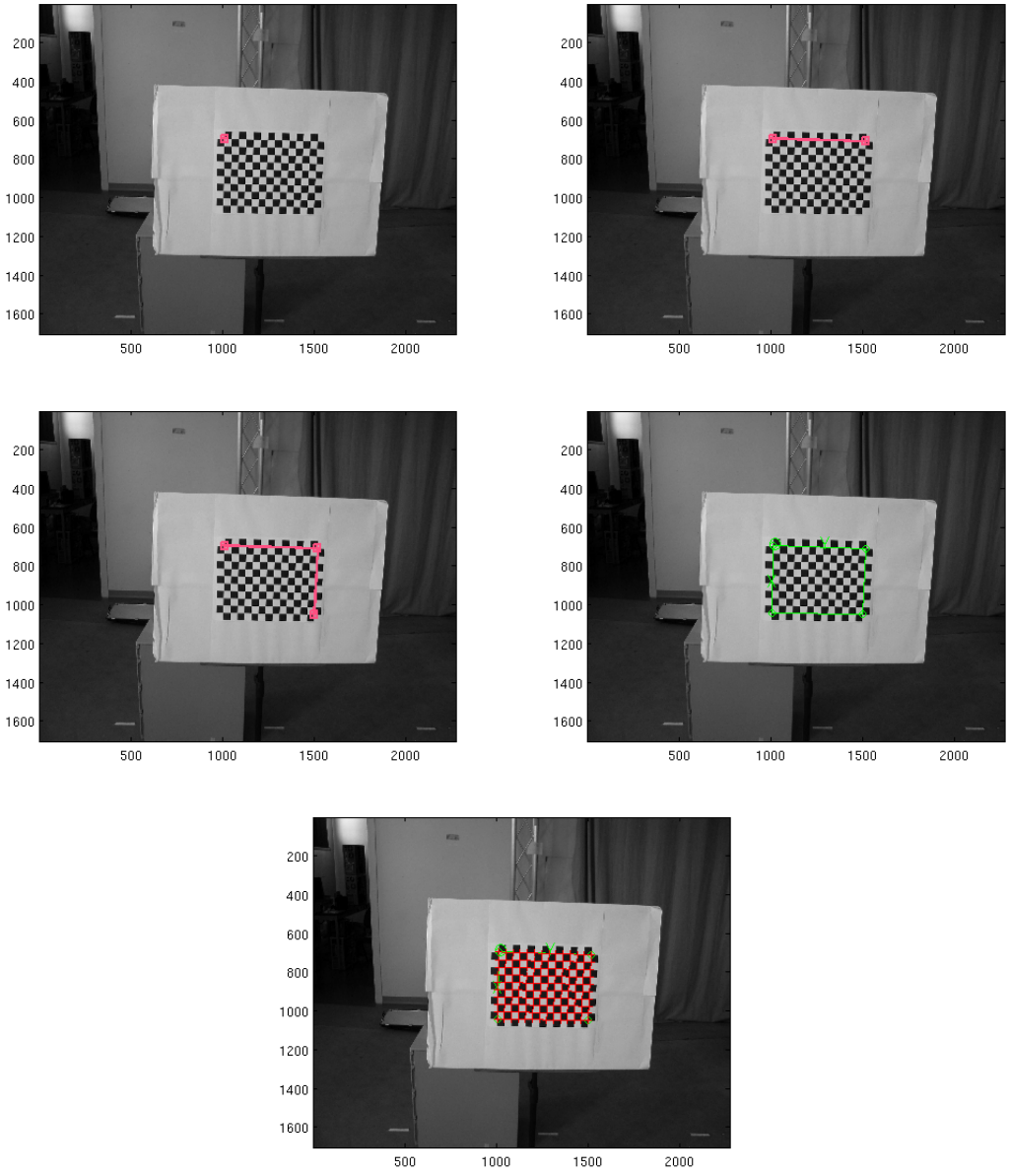


**Figura 4.4:** Estrazione degli angoli della scacchiera del Camera Calibration Toolbox for Matlab

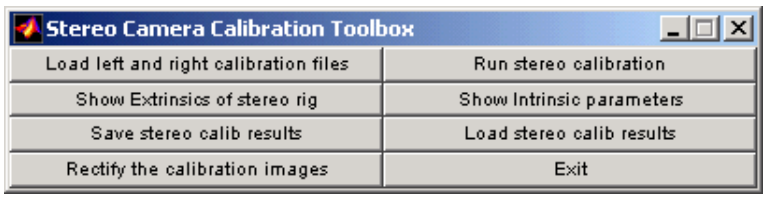


**Figura 4.5:** Interfaccia utente del Camera Calibration Toolbox for Matlab, Stereo Calibration

fornirà il vettore *om* ed il vettore di traslazione $T$, la matrice di rotazione $R$ può essere ricavata dal vettore *om* attraverso il comando Matlab *rodrigues*.

$$R = rodrigues(om) \tag{4.1}$$

Dopo questa operazione siamo in possesso dei seguenti parametri:

- parametri intrinseci della telecamera ad infrarossi del Kinect (Depth Camera);
- parametri intrinseci della telecamera a colori del Kinect (Color Camera);
- calibrazione stereo fra le due telecamere del Kinect ($R$ e $T$);
- parametri intrinseci della fotocamera digitale aggiuntiva;
- calibrazione stereo fra la telecamera a colori del Kinect e la fotocamera digitale ($R_{stereo}$ e $T_{stereo}$).

Avendo a disposizione questi dati possiamo fare le seguenti operazioni:

1. proiettare ogni punto della Depth Camera nello spazio 3D, creando un modello 3D, di tipo *nuvola di punti* (*point cloud*), in cui manca ancora l'informazione sul colore;
2. rimappare ogni punto del modello 3D creato al punto 1 sull'immagine a colori del Kinect assegnando così il colore corrispondente ai dati di profondità (si usano $R$ e $T$);
3. riproiettare ogni dato ottenuto, contenente sia l'informazione di profondità sia quella di colore, nello spazio 3D. Si ottiene un modello 3D contenente tutte le informazioni ottenute dal dispositivo Kinect;
4. rimappare ogni punto del modello 3D ottenuto al punto 3 sull'immagine a colori della fotocamera (si usano $R_{stereo}$ e $T_{stereo}$). In questa situazione si potrà calcolare la qualità del modello 3D utilizzando l'immagine della fotocamera come riferimento.

**Figura 4.6:** Acquisizione 1

**Figura 4.7:** Acquisizione 2

## 4.2 Acquisizioni immagini

Con lo stesso metodo di acquisizione usato per catturare le immagini utilizzate nella calibrazione sono state acquisite una serie di immagini con diversi soggetti e con diverse condizioni di illuminazione. I punti del modello 3D, generato con i dati del Kinect, vengono proiettati sulla vista laterale in modo da poterli confrontare con i dati della fotocamera che vengono considerati i dati di riferimento.

### 4.2.1 Acquisizione immagini in ambiente interno

Le prime acquisizioni (Figure da 4.6 a 4.13) sono state effettuate in un ambiente interno con situazioni di luce controllata. Si può notare che le silhouette degli oggetti inquadrati sono riconoscibili con l'aggiunta di rumore sui bordi, legato alla particolare risoluzione e struttura del pattern infrarosso proiettato sulla scena.
Si può quindi affermare che in queste condizioni si possono usare i dati della Depth Camera del Kinect per ricostruire la scena osservata andando a creare una mappa di profondità utilizzabile per il rilevamento degli ostacoli.

**Figura 4.8:** Acquisizione 3

**Figura 4.9:** Acquisizione 4

**Figura 4.10:** Acquisizione 5

**Figura 4.11:** Acquisizione 6

**Figura 4.12:** Acquisizione 7

**Figura 4.13:** Acquisizione 8

### 4.2.2 Acquisizione immagini in ambiente esterno

Le successive acquisizioni (Figure da 4.14 a 4.16) sono state effettuate in ambiente esterno, quindi in una situazione di luce non controllata e con forte presenza di radiazione elettromagnetica alla stessa frequenza del segnale infrarosso usato dal dispositivo. Nei dati di profondità si perde ogni riferimento alle forme degli oggetti inquadrati nella scena a causa della luce solare che viene catturata dalla fotocamera a infrarossi.
Nelle condizioni di illuminazione considerate i dati acquisiti dal sensore Kinect

**Figura 4.14:** Acquisizione 9

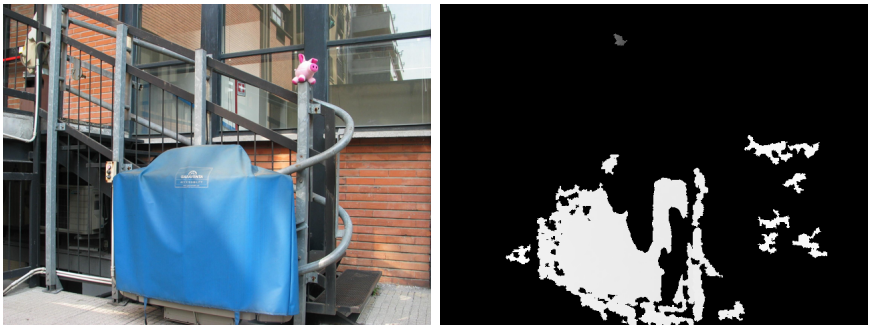

**Figura 4.15:** Acquisizione 10

**Figura 4.16:** Acquisizione 11

non sono molto utili per la creazione di un modello 3D della scena in quanto si nota che la gran parte dell'informazione non viene acquisita in maniera corretta dal dispositivo.

**Figura 4.17:** Acquisizione 12

**Figura 4.18:** Acquisizione 13

**Figura 4.19:** Acquisizione 14

### 4.2.3 Acquisizione immagini in ambiente interno con luce riflessa

Nella situazione di ambiente interno è stata considerata la situazione particolare in cui la luce solare proveniente dall'esterno arriva direttamente o indirettamente al dispositivo.
Nelle acquisizioni effettuate in questa condizione(Figure 4.17 e 4.18) si nota un peggioramento della situazione rispetto alle altre acquisizioni in ambiente interno e al limite, come si vede nella seconda immagine (Figura 4.18), si ha la perdita quasi totale delle informazioni di profondità. Questo porta a concludere che il dispositivo non permette una stima accurata della profondità non solo in presenza di luce solare diretta ma anche di quella indiretta, come quella riflessa dal muro bianco presente nelle scene considerate.
Si può quindi affermare che anche in ambiente interno il Kinect ha dei limiti di utilizzo dovuti alla radiazione solare che acquisisce.

### 4.2.4 Acquisizione immagini in assenza di luce

L'ultima situazione considerata è stata quella di quasi totale assenza di luce, per fare ciò l'acquisizione è stata effettuata in un ambiente interno in cui si è evitato

che la luce esterna filtrasse all'interno.
Il risultato (Figura 4.19) mostra che il dispositivo, utilizzando un emettitore e un ricevitore a infrarossi, è in grado di acquisire modelli tridimensionali degli oggetti con buona precisione. Infatti l'assenza di radiazione luminosa gioca a favore della Depth Camera in quanto la componente di rumore esterna viene ridotta.

### 4.2.5 Funzionalità del dispositivo nelle diverse condizioni di acquisizione

Considerando tutte le acquisizioni effettuate si può sostanzialmente descrivere le funzionalità del dispositivo nei diversi scenari di utilizzo. Si può affermare che:

- in condizioni di illuminazione controllata il dispositivo funziona in maniera corretta, creando una mappa di profondità pressoché completa.

- in condizioni di illuminazione non controllata, all'esterno o comunque con la radiazione solare che colpisce il Kinect in maniera diretta o anche indiretta, il dispositivo va in contro ad un funzionamento non corretto fornendo una ricostruzione errata della mappa di profondità in cui una grande quantità di dati risulta mancante in quanto il dispositivo risulta accecato dalla luce solare.

- in condizioni di assenza di illuminazione il dispositivo funziona in maniera corretta per quanto riguarda la definizione della mappa di profondità in quanto per fare ciò utilizza la camera ad infrarossi che analizza il pattern infrarosso emesso dal dispositivo stesso. L'assenza di luce visibile non permette di avere i dati sulla Color Camera ma non influisce sui dati della Depth Camera che come detto lavora su lunghezze d'onda nell'infrarosso.

## 4.3 Prestazioni del dispositivo

Per valutare le prestazioni del dispositivo (ovvero la qualità del modello 3D ricostruito) si è cercato di trovare una relazione tra due diversi parametri:

- MSE (*Mean Squared Error*);

- Valutazione di consistenza della segmentazione di un'immagine.

### 4.3.1 MSE

In statistica, la *distorsione quadratica media* (*Mean Squared Error* o MSE), è un parametro largamente utilizzato per misurare l'accuratezza di uno stimatore confrontando il valore vero con quello stimato.
Il valore di MSE può essere calcolato come:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \widehat{x}_i)^2 \tag{4.2}$$

Nel caso del Kinect questo calcolo viene fatto in maniera indiretta sui dati di colore prendendo come dati di riferimento quelli della fotocamera digitale.
Come specificato in precedenza, i dati della depth map ottenuta possono essere rimappati sull'immagine prodotta dalla Color Camera del Kinect tramite un'operazione di warping. Per ogni pixel al quale è associato un valore di profondità valido, è possibile generare un punto 3D le cui coordinate sono ricavate dal valore di depth e dai parametri intrinseci ed estrinseci stimati. Il colore associato è quello del pixel corrispondente nell'immagine colore.
Dal modello 3D generato (che solitamente in letteratura viene chiamato *nuvola di punti* o *point cloud*), è possibile proiettare i punti 3D sul piano immagine della fotocamera laterale.
Le operazioni descritte in precedenza (warping) possono essere effettuate solo se il valore di depth fornito in partenza è valido.
A questo punto si hanno a disposizione due immagini RGB che si riferiscono alla stessa telecamera (quella laterale).
Si calcola l'MSE partendo dall'equazione generale 4.2:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(S(i) - S_w(i))^2 \tag{4.3}$$

dove:

- $S(i)$ è il dato dell'immagine della fotocamera digitale;

- $S_w(i)$ è il dato colore del Kinect riportato sulla fotocamera.

Considerando $P$ come l'insieme di tutti i punti che compongono l'immagine, l'insieme $N$ su cui viene calcolato l'MSE è un sottoinsieme di $P$. Infatti le fotocamere sono poste in posizioni diverse e quindi non inquadrano perfettamente la stessa scena e di conseguenza esistono dei punti della telecamera laterale non visibili al

sensore Kinect. Questo porta, dopo aver rimappato i dati del Kinect sul piano della fotocamera, ad una immagine a cui mancano dei dati; l'MSE viene calcolato sul sottoinsieme dei pixel che possiedono l'informazione del Kinect.
Il calcolo dell'MSE, sul sottoinsieme di pixel $N$, in realtà viene fatto su ciascuna componente RGB separatamente:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{((S_R(i) - S_{wR}(i))^2 + (S_G(i) - S_{wG}(i))^2 + (S_B(i) - S_{wB}(i))^2)}{3}\right) \tag{4.4}$$

dove:

- $S_R$ e $S_{wR}$ rappresentano la componente rossa delle due immagini;

- $S_G$ e $S_{wG}$ rappresentano la componente verde delle due immagini;

- $S_B$ e $S_{wB}$ rappresentano la componente blu delle due immagini.

Il valore di MSE diminuisce al migliorare delle prestazioni, infatti, in un sistema ideale privo di errori e di disturbi, il dato di colore del Kinect sarebbe riproiettato esattamente sul pixel corrispondente della fotocamera e i valori dei dati RGB sarebbero esattamente gli stessi fornendo così un valore di MSE pari a 0.

### 4.3.2 Consistenza della segmentazione

L'idea da cui si è partiti per trovare un parametro utilizzabile per valutare le prestazioni del dispositivo è quella di avere l'immagine a colori e l'immagine di depth (mappa di profondità data dalla Depth Camera) riproiettata su quella a colori, dividere in segmenti una delle due immagini e valutare la consistenza di ciascun segmento considerando l'altra immagine, usando quindi un metodo di valutazione unsupervised.

Il vantaggio di una metrica unsupervised sulla segmentazione è costituito dal fatto di non richiedere un'immagine di riferimento segmentata manualmente. Questo vantaggio è indispensabile in applicazioni di segmentazione general-purpose, soprattutto in quelle implementate in sistemi real-time, dove una grande varietà di immagini devono essere segmentate.

**Segmentazione** La segmentazione di un'immagine è il processo di partizione di un'immagine in regioni significative. Lo scopo della segmentazione, in generale, è segmentare un'immagine, ovvero dividere l'immagine in regioni di pixel contigui caratterizzati da specifiche di colore (o di profondità se lo si fa sulle depth map) simili.
Pertanto ciascun pixel in una regione è simile agli altri della stessa regione per una qualche proprietà o caratteristica. Regioni adiacenti sono significativamente differenti rispetto alla caratteristica considerata. Se i pixel hanno caratteristiche simili, probabilmente la segmentazione individua gli oggetti presenti nella scena.
Esistono tre tipi principali di algoritmo per segmentare un'immagine: basati sull'istogramma, basati sulla crescita/divisione delle regioni, basati sulla tecnica del rilassamento.

*Haralick* e *Shapiro* [8] proposero quattro criteri per definire una buona segmentazione:

1. le regioni devono essere uniformi e omogenee rispetto ad una o più caratteristiche;

2. regioni adiacenti devono avere differenze significative rispetto alla caratteristica per la quale sono uniformi;

3. l'interno delle regioni dovrebbe essere semplice e privo di buchi;

4. i confini delle regioni dovrebbero essere semplici, non irregolari ed essere spazialmente accurati.

I primi due criteri esaminano le caratteristiche degli oggetti nell'immagine, perciò vengono chiamati *Characteristic Criteria*, invece gli ultimi due criteri sono basati su come ciascuna regione sarà considerata come un singolo oggetto da una persona, perciò vengono chiamati *Semantic Criteria*.

La metrica che si è deciso di utilizzare è quella che, in letteratura, è normalmente denominata $F_{RC}$ [9]. Essa si basa su un criterio di valutazione che tiene conto sia dell'omogeneità globale all'interno del segmento sia della differenza fra i diversi segmenti.
La scelta è caduta su questa metrica in quanto in letteratura [10] viene descritta come una delle più performanti a confronto con le altre metriche unsupervised.

Inoltre $F_{RC}$ è una metrica molto bilanciata rispetto alla sotto-segmentazione e alla sovra-segmentazione, con solo lievi e trascurabili biases in un modo o nell'altro.

Una volta eseguita la segmentazione, la metrica $F_{RC}$ può essere calcolata separando i due criteri su cui si basa:

(a) la parte della metrica $F_{RC}$ che valuta l'uniformità all'interno di un segmento è basata sul criterio 1:

$$D_{intra} = \frac{1}{N} \sum_{i=1}^{N} \left( eT_i \cdot \frac{n_i}{w \cdot h \cdot 255} \right) \tag{4.5}$$

dove:

- $N$: numero di segmenti in cui è stata divisa l'immagine;
- $n_i$: numero di pixel appartenenti al segmento i-esimo;
- $eT_i$: errore assoluto medio relativo alle tre componenti di colore dei pixel del segmento i-esimo:

$$eR_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \left( I_{R_j} - I_{RM_i} \right) \tag{4.6}$$

$$eG_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \left( I_{G_j} - I_{GM_i} \right) \tag{4.7}$$

$$eB_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \left( I_{B_j} - I_{BM_i} \right) \tag{4.8}$$

$$eT_i = \sqrt{\frac{eR_i^2 + eG_i^2 + eB_i^2}{3}} \tag{4.9}$$

$I_{R_j}$, $I_{G_j}$ e $I_{B_j}$ sono le componenti di colore del pixel j-esimo;
$I_{RM_i}$, $I_{GM_i}$ e $I_{BM_i}$ sono le componenti di colore medie dei pixel appartenenti al segmento i-esimo;

- $w$ e $h$: dimensioni dell'immagine.

(b) la parte della metrica $F_{RC}$ che valuta la differenza fra i segmenti si basa sul criterio 2:

$$D_{inter} = \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{k=1}^{N-1} \left( \frac{D_{ispar_{ik}}}{I_{M_i} + I_{M_k}} \right) \cdot \frac{n_i}{w \cdot h} \right] \tag{4.10}$$

dove:

- $N$: numero di segmenti in cui è stata divisa l'immagine;

- $n_i$: numero di pixel appartenenti al segmento i-esimo;

- la sommatoria interna è limitata a $N - 1$ perché non si deve considerare il segmento che in quel momento è considerato nella sommatoria esterna;

- $D_{ispar_{ik}}$: calcolo della differenza fra due segmenti diversi:

$$D_{ispar_{ik}} = \sqrt{\frac{(I_{RM_k} - I_{RM_i})^2 + (I_{GM_k} - I_{GM_i})^2 + (I_{BM_k} - I_{BM_i})^2}{3}} \tag{4.11}$$

  $I_{RM_i}$, $I_{GM_i}$ e $I_{BM_i}$ sono le componenti di colore medie dei pixel appartenenti al segmento i-esimo;
  $I_{RM_k}$, $I_{GM_k}$ e $I_{BM_k}$ sono le componenti di colore medie dei pixel appartenenti al segmento k-esimo;

- $I_{M_i}$ e $I_{M_k}$: medie delle tre componenti di colore dei due segmenti considerati:

$$I_{M_i} = \sqrt{\frac{I_{RM_i}^2 + I_{GM_i}^2 + I_{BM_i}^2}{3}} \tag{4.12}$$

$$I_{M_k} = \sqrt{\frac{I_{RM_k}^2 + I_{GM_k}^2 + I_{BM_k}^2}{3}} \tag{4.13}$$

- $w$ e $h$: dimensioni dell'immagine.

Questi due criteri possono essere uniti per ottenere un valore della metrica $F_{RC}$ che indica la bontà della segmentazione:

$$val = \frac{D_{inter} - D_{intra}}{2} \tag{4.14}$$

Il risultato finale della metrica aumenta al migliorare delle prestazioni, infatti per come sono stati definiti i parametri che compongono la metrica, $D_{inter}$ aumenta al migliorare delle prestazioni mentre $D_{intra}$ diminuisce al migliorare delle prestazioni.

Avendo a disposizione due tipi di immagini diverse, si possono eseguire due segmentazioni differenti che possono essere più o meno compatibili fra di loro. Nel caso in cui esse siano fortemente simili è possibile dedurre che il modello generato presenti una perfetta coerenza fra i dati cromatici e i dati geometrici.
Le segmentazioni considerate sono:

- segmentazione sull'immagine a colori; non si tiene conto delle coordinate dei pixel ma solo dei dati RGB che forniscono informazione sul colore. In questo caso si segmenta considerando che pixel vicini che hanno lo stesso colore appartengano allo stesso oggetto e quindi abbiano dei valori di profondità simili.

- segmentazione sull'immagine di depth; anche in questo caso non si considerano le coordinate dei pixel ma solo i dati di profondità forniti dal Kinect. Nella pratica si divide il range di dati in diversi piani i quali conterranno ciascuno valori simili di profondità.

Per eseguire la segmentazione in maniera pratica si è prima utilizzato l'algoritmo proposto da Felzenszwalb e Huttenlocher [11], il cui codice scritto in C++ è disponibile in internet (*www.cs.brown.edu/∼pff/segment/*). Tale algoritmo produce una nuova immagine di dimensioni uguali a quella di input in cui tutti i pixel appartenenti ad una certa regione hanno lo stesso colore (scelto a caso e diverso da regione a regione).
Questo algoritmo ha due problemi fondamentali, uno legato alla pesantezza computazionale, l'altro legato al fatto che non è possibile eseguire la segmentazione come si desidera, cioè senza considerare le coordinate dei pixel.
Per ovviare a questi problemi, soprattutto al secondo, si è deciso di usare un algoritmo K-means [12], cioè un algoritmo di clustering che segue una procedura iterativa cercando di minimizzare la varianza totale intra-cluster. Inizialmente vengono create $K$ partizioni e ad ognuna di esse vengono assegnati i punti d'ingresso o in maniera casuale o usando alcune informazioni euristiche. L'algoritmo

calcola il centroide di ogni gruppo e successivamente crea delle nuove partizioni associando ogni punto di ingresso al cluster il cui centroide è più vicino ad esso. Quindi vengono calcolati i centroidi per i nuovi cluster e così via finché l'algoritmo non converge.
Questo algoritmo è implementato dal comando Matlab *kmeans* che ha la seguente struttura:

$$\boldsymbol{IDX} = \mathbf{kmeans}(\boldsymbol{X},\boldsymbol{k})$$

La variabile $X$ è una matrice $n$ x $p$ contenente i punti da dividere in $k$ cluster. Le righe di $X$ rappresentano i punti mentre le colonne rappresentano le variabili da considerare. Nel nostro caso, se si vorrà segmentare l'immagine a colori, $X$ dovrà essere una matrice $n$ x 3 che conterrà le informazioni RGB di ciascun punto mentre, se si vorrà segmentare l'immagine di profondità, $X$ dovrà essere una matrice $n$ x 1 che conterrà per ciascun punto solo l'informazione di profondità. Il comando *kmeans* restituisce un vettore *IDX* di dimensioni $n$ x 1 contenente l'indice del cluster a cui ogni punto appartiene.

### 4.3.3 Procedura di valutazione delle prestazioni

Per valutare le prestazioni del dispositivo sono state acquisite una serie di set di immagini in diverse condizioni di illuminazione e con diversi soggetti, ogni set è composto da una decina di acquisizioni identiche.
Vengono prese in considerazione più immagini per ogni scena considerata in modo da poter valutare anche cosa cambia in termini di prestazioni mediando più immagini insieme, cercando quindi di eliminare alcuni errori presenti nei dati forniti dal dispositivo.
I passi da seguire una volta acquisite le immagini sono:

- mediare, eventualmente, i dati di profondità delle acquisizioni considerate;
- proiettare ciascun punto della Depth Camera nello spazio 3D e conseguentemente riproiettarlo sul piano della Color Camera;
- riproiettare ciascun dato ottenuto, formato sia dall'informazione di profondità sia da quella di colore, nello spazio 3D per poi rimapparlo sul piano della fotocamera digitale di riferimento;
- modificare l'illuminazione dei dati in modo da renderla molto simile, idealmente uguale, a quella dell'immagine di riferimento;

- calcolare il valore di MSE e PSNR, che rappresentano la prima parte della metrica che si vuole considerare;

- eseguire la segmentazione sia sull'immagine a colori sia su quella di depth e calcolare in entrambi i casi la consistenza della segmentazione effettuata. Le immagini da considerare devono essere entrambe riferite al piano della Color Camera. I dati ottenuti rappresentano la seconda parte della metrica considerata.

L'equalizzazione che si esegue sull'illuminazione viene fatto in modo da avere dei risultati più corretti, in modo da non tenere conto della possibile differenza di illuminazione vista dal kinect e dalla fotocamera di riferimento e della possibile differenza di colore nelle immagini dovuta ai sensori diversi dei dispositivi usati. In pratica l'equalizzazione e di conseguenza il calcolo dell'MSE viene fatto sulla componente di luminanza della terna YUV che non è altro che la media delle tre componenti RGB.
L'operazione di equalizzazione è stata effettuata utilizzando il comando Matlab histeq($I, HGRAM$) che modifica l'immagine $I$ in modo che abbia un istogramma simile a quello fornito, cioè quello dell'immagine di riferimento, quindi $HGRAM$ è l'istogramma dell'immagine della fotocamera di riferimento.

Nella figura 4.20 si nota che le due immagini a colori che dovrebbero essere confrontate mostrano un'illuminazione diversa e anche i colori non sono proprio gli stessi. Si può notare che il passaggio aggiuntivo dell'equalizzazione, effettuato sull'immagine che dal Kinect è riportata sul piano della fotocamera, porta a due immagini, costituite dalla sola informazione di luminanza, con le stesse caratteristiche e quindi il confronto che si esegue sarà indipendente dalla differenza di illuminazione e dai due diversi tipi di sensori utilizzati nei due dispositivi.
Per il calcolo dell'MSE l'equazione 4.4 si modifica in:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(I(i) - I_w(i))^2 \tag{4.15}$$

dove:

- $I(p)$ è il dato di luminanza della fotocamera digitale;

- $I_w(p)$ è il dato di luminanza dell'immagine colore del Kinect riportata sulla fotocamera ed equalizzata.

**Figura 4.20:** Fasi della procedura di equalizzazione

Oltre al MSE è stato calcolato il PSNR (*peak signal-to-noise ratio*), questo indice di qualità delle immagini è definito come il rapporto tra la massima potenza di un segnale e la potenza di rumore. Poiché molti segnali hanno una gamma dinamica molto ampia, il PSNR è solitamente espresso in termini di scala logaritmica di decibel.
Maggiore è il valore del PSNR maggiore è la somiglianza con l'immagine di riferimento, nel senso che si avvicina maggiormente ad essa da un punto di vista percettivo umano.

$$PSNR = 20 \cdot log_{10}\left(\frac{MAX\{I\}}{\sqrt{MSE}}\right) = 10 \cdot log_{10}\left(\frac{255^2}{MSE}\right) \tag{4.16}$$

In realtà nella procedura di equalizzazione i dati relativi all'immagine assumono valori compresi nel range 0÷1 e non più nel range 0÷255, quindi l'equazione 4.16 deve essere modificata nel seguente modo:

$$PSNR = 20 \cdot log_{10}\left(\frac{1}{MSE}\right) \tag{4.17}$$

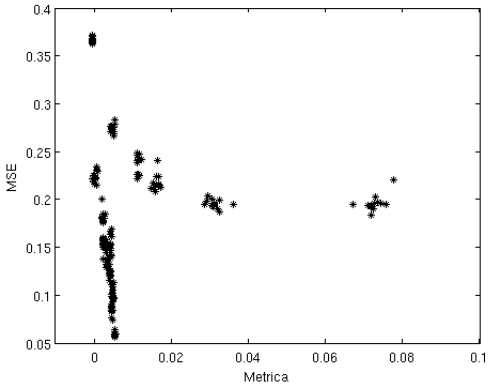


**Figura 4.21:** Risultati della metrica valutata sulla segmentazione dell'immagine a colori

### 4.3.4 Risultati

In questo paragrafo si andranno ad analizzare i risultati ottenuti nelle acquisizioni effettuate secondo la procedura descritta nella sezione precedente. Per fare questa analisi è stato usato un sistema costituito, oltre dal Kinect, da una fotocamera laterale con risoluzione pari alla metà di quella della Color Camera del Kinect. La fotocamera laterale è stata usata come camera di riferimento nel calcolo dell'MSE. Come spiegato precedentemente le segmentazioni possibili sono due e quindi i risultati devono essere considerati separatamente.

#### Segmentazione dell'immagine a colori

Nella figura 4.21 si possono osservare i risultati ottenuti considerando la metrica valutata sulla segmentazione dell'immagine a colori.
Osservando i dati si possono riconoscere due andamenti differenti, uno lineare e uno simile ad un esponenziale. Analizzando le scene acquisite si osserva che appartengono all'andamento esponenziale i dati delle scene che presentano forti riflessioni, provocate da finestre o da specchi.

La figura 4.22 mostra i due andamenti osservati analizzando i dati ottenuti.
L'andamento teorico che si sarebbe pensato di ottenere era un andamento decrescente, infatti l'MSE diminuisce mentre la metrica che valuta la segmentazione aumenta al crescere delle prestazioni. I dati sono coerenti con l'andamento teorico con però due andamenti distinti in base alla presenza o meno di riflessioni nella scena inquadrata.

Per verificare gli andamenti ottenuti si è considerato un nuovo sistema costituito

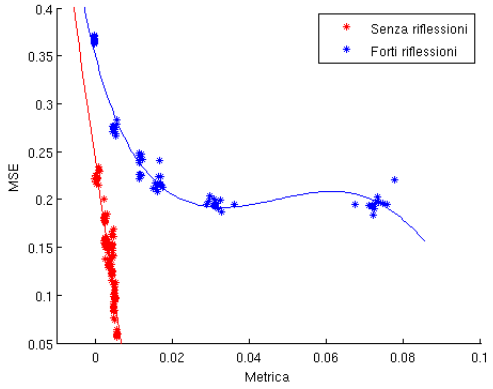


**Figura 4.22:** Interpolazione dei dati ottenuti considerando la metrica valutata sulla segmentazione dell'immagine a colori

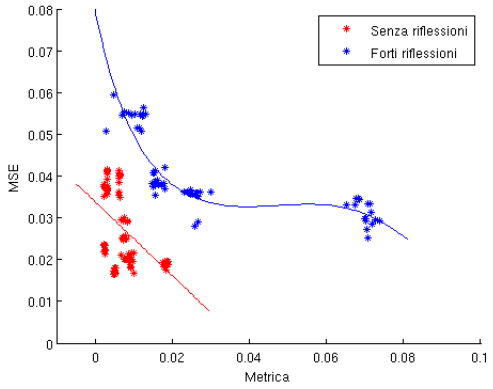


**Figura 4.23:** Risultati della metrica valutata sulla segmentazione dell'immagine a colori del secondo sistema considerato

dal Kinect e da una fotocamera laterale diversa; i risultati ottenuti sono visibili in figura 4.23.
Si nota che anche in questo caso si ottengono due andamenti differenti ma sempre decrescenti, simili a quelli ottenuti considerando il primo sistema. Si ottiene un andamento simile ad un esponenziale per i dati provenienti da scene che presentano riflessioni, e un andamento lineare decrescente per i dati provenienti da scene che non presentano riflessioni.

## Segmentazione dell'immagine di profondità

Nella figura 4.24 si possono osservare i risultati ottenuti considerando la metrica valutata sulla segmentazione dell'immagine di profondità di entrambi i sistemi considerati precedentemente.
Analizzando i dati, anche considerando separatamente le scene che presentano

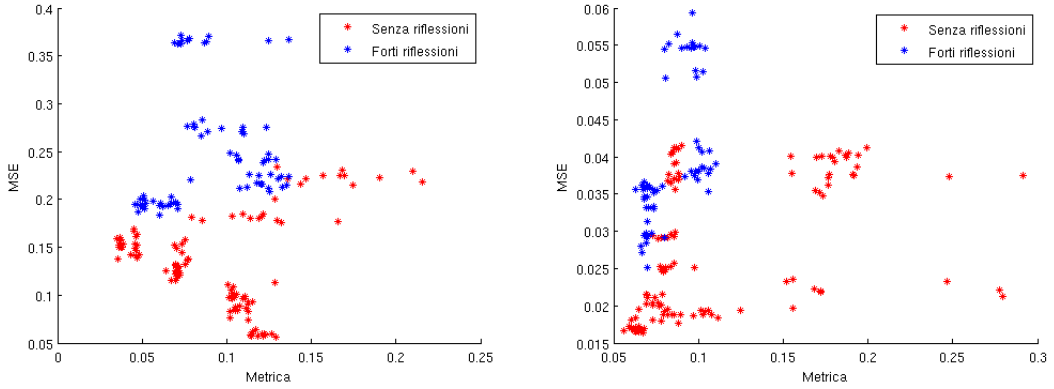


**Figura 4.24:** Risultati della metrica valutata sulla segmentazione dell'immagine di profondità dei due sistemi considerati

riflessioni e le scene che non le presentano, non si nota un particolare andamento che metta in relazione i dati considerati.
Questo è dovuto al fatto di valutare la segmentazione dell'immagine di profondità utilizzando l'immagine a colori. L'immagine di profondità presenta gli oggetti con i contorni frastagliati a causa del rumore che affligge il Kinect. I contorni dei segmenti creati sono anch'essi irregolari e quindi quando i segmenti vengono valutati utilizzando l'immagine a colori, che presenta contorni regolari, si ottengono valori della metrica che non hanno un andamento preciso.

## Analisi dei dati

Avendo analizzato i dati ottenuti si può concludere che solo la metrica che valuta la segmentazione dell'immagine a colori utilizzando l'immagine di profondità può essere utilizzata per valutare le prestazioni del dispositivo, ovvero la qualità del modello 3D ricostruito.
Conoscere gli andamenti visti in precedenza (Figure 4.22 e 4.23), ci permette di fare una stima dell'MSE, quindi della qualità dell'informazione, senza utilizzare un'immagine di riferimento proveniente da una fotocamera laterale. Infatti data la segmentazione dell'immagine a colori del Kinect e la corrispondente immagine di profondità si può usare la metrica precedentemente descritta e, una volta ottenuto un risultato, usarlo per stimare l'MSE attraverso gli andamenti trovati.

# Capitolo 5

# Soluzione con 2 Kinect

Si è valutato nel capitolo precedente la possibilità di generare un modello 3D a partire dai dati acquisiti da un sensore Microsoft Kinect. Tuttavia il modello risultante risulta essere fortemente incompleto a causa della presenza di rumore che non permette una stima accurata e dalla possibilità che molte occlusioni siano presenti. È possibile ottenere un modello più accurato e ridurre il numero di punti mancanti utilizzando un sistema multisensore. In particolare, in questo capitolo verrà analizzata la possibilità di affiancare due sensori Kinect creando un sistema stereo composto da due telecamere colore e due telecamere profondità. Tale soluzione non è priva di problemi da risolvere.

Il primo problema che deve essere affrontato è quello dell'interferenza che si crea quando i due dispositivi sono attivi nello stesso momento, infatti in questa condizione il singolo dispositivo vede sia il proprio pattern infrarosso sia quello emesso dall'altro dispositivo. Questo porta ad errori nella valutazione della profondità di alcuni punti.

## 5.1 Valutazione dell'interferenza

L'interferenza del secondo kinect varia a seconda dell'angolazione a cui i due Kinect sono posti. Per valutare questa interferenza e come varia sono state fatte delle acquisizioni di un Kinect con un altro dispositivo attivo posto ad angolazioni differenti distribuite lungo un angolo retto.
Per cercare di mitigare l'interferenza sono state eseguite trenta acquisizioni per ogni angolazione e poi i dati di profondità sono stati mediati.

Osservando i risultati ottenuti (Figure 5.1, 5.2, 5.3) si nota come l'interferenza si veda sotto forma di zone in cui i dati di profondità sono sicuramente errati. Nell'ultima acquisizione (Figura 5.3, angolo di 90°) si potrebbe pensare che l'interferenza non agisca ma analizzando bene i dati si nota che lateralmente il dato

**Figura 5.1:** Interferenza con angolo di 0°(sinistra) e 26.4°(destra)

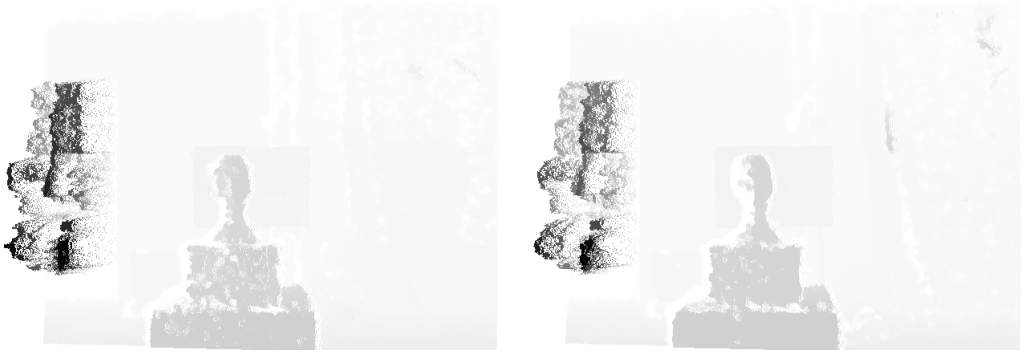

**Figura 5.2:** Interferenza con angolo di 50.3°(sinistra) e 72.5°(destra)

**Figura 5.3:** Interferenza con angolo di 90°

di profondità manca proprio dove punta il secondo dispositivo.

## 5.2 Ricostruzione zone soggette a interferenza

Avendo visto che l'interferenza dovuta ad un secondo Kinect non può essere risolta con la semplice media aritmetica, si è cercato di trovare un altra procedura che possa portare a ricostruire i dati di profondità corrotti dall'interferenza.
La procedura che è stata seguita è la seguente:

- fondere le varie acquisizioni utilizzando un sistema di media più intelligente, cioè un algoritmo che prima di mediare i dati a disposizione ne valuti la consistenza ed elimini i dati sicuramente errati;

- analizzare i dati di profondità in modo da riconoscere le zone in cui i dati sono corrotti a causa dell'interferenza del secondo dispositivo e quindi eliminarli;

- ricostruire i dati per le zone soggette a interferenza attraverso un'interpolazione dei dati ritenuti corretti.

### 5.2.1 Media alternativa

L'idea alternativa che è stata seguita per mediare i dati delle varie acquisizioni consiste nell'analizzare i dati che si hanno per ogni singolo pixel ed eliminare quelli che si rivelano errati.
Per fare ciò si è costruito un algoritmo che controlla la differenza fra il valore massimo e quello minimo e se questa è inferiore ad una soglia, che nel nostro caso è stata fissata a 10, considera tutti i dati come omogenei ed esegue una semplice media matematica. Nel caso che la differenza calcolata superi il valore della soglia, l'algoritmo divide i dati di profondità in due gruppi che vengono definiti $A$ e $B$. Nell'insieme $A$ trovano posto il primo valore di profondità e tutti quelli ad esso simili, cioè quelli che hanno un valore di profondità che si discosta meno di una soglia dal primo valore inserito nell'insieme. Tutti i dati che non risultano simili vengono inseriti nell'insieme $B$.
Se quando tutti i dati sono stati divisi l'insieme $A$ contiene più elementi dell'insieme $B$, allora si considerano i dati dell'insieme $A$ come i valori corretti e viene eseguita la media su di essi. Se, al contrario, è l'insieme $B$ a contenere più elementi l'algoritmo dovrà eseguire una nuova iterazione considerando come dati di

profondità solo quelli presenti nell'insieme $B$. Questa nuova iterazione è necessaria in quanto i dati presenti nell'insieme $B$ sono sicuramente non simili con quelli presenti nell'insieme $A$, ma nessuno può garantire che siano omogenei fra di loro. Nel caso si abbiano solo due dati di profondità ed uno di essi è pari a 1023 (valore massimo di profondità), quest'ultimo può essere scartato in quanto è stato osservato che i valori elevati di profondità sono poco corrispondenti con la realtà.
Il codice dell'algoritmo appena descritto è presente nell'appendice B (Paragrafo B.2).

### 5.2.2 Analisi dei dati

Per trovare le zone in cui i dati di profondità sono corrotti dall'interferenza bisogna proiettarli sull'immagine a colori del Kinect in modo da utilizzare i dati di quest'ultima come riferimento.
È necessario eseguire una segmentazione (vedi paragrafo 4.3.2 a pag. 57) sull'immagine a colori.

Nel nostro caso la segmentazione è fatta considerando il colore di ciascun pixel e le sue coordinate utilizzando l'algoritmo di Felzenszwalb. Se la segmentazione viene fatta in modo corretto nel singolo segmento troveremo i pixel che hanno un colore simile e quindi che presumibilmente apparterranno allo stesso oggetto e quindi avranno dei valori di profondità simili (Figura 5.4).

Il passo successivo consiste nell'analizzare un segmento alla volta. All'interno del singolo segmento si va a costruire un istogramma dei dati di profondità, trovato il picco fondamentale si considera una soglia pari al 50% del numero di dati uguali al valore della fondamentale. Si considera la differenza tra il numero di elementi pari al valore della fondamentale meno il numero degli elementi pari a ciascun valore che risulta sopra la soglia.
Se la differenza massima che viene calcolata è inferiore a 20 (valore ottimale trovato dopo alcune prove) si può affermare che il segmento presenta un solo valore fondamentale e quindi tutti i dati di profondità in esso contenuti possono essere considerati validi.
Se invece la differenza massima è maggiore di 20, il segmento presenta un secondo picco significativo e quindi il passo successivo è quello di dividere i valori di profondità in due gruppi, il primo conterrà i valori che si possono considerare contenuti nella campana del picco fondamentale, mentre il secondo gruppo quelli relativi al picco secondario. I pixel che hanno un valore di profondità pari ad un valore contenuto nel primo gruppo possono essere considerati validi mentre gli altri pixel sono quelli che presumibilmente sono soggetti all'interferenza e quindi vengono posti a 0 perché non validi.
Nella figura 5.5 è possibile vedere il risultato di questo processo.

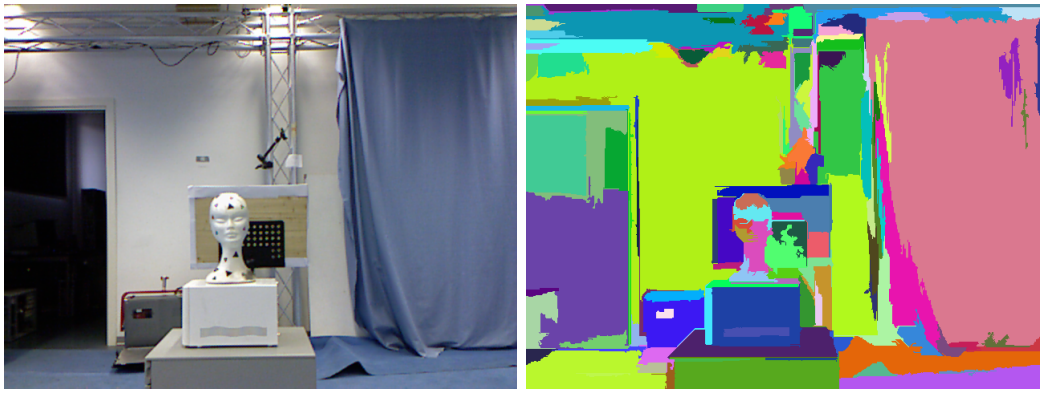

**Figura 5.4:** Immagine a colori e la sua segmentazione

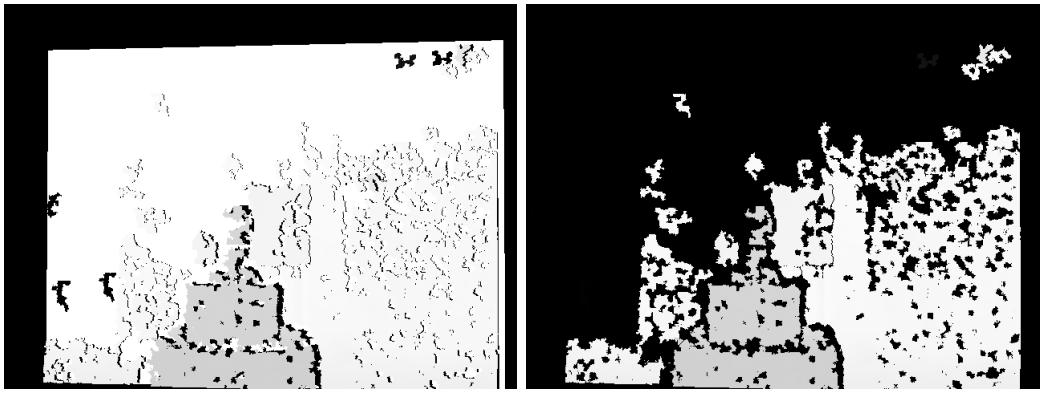

**Figura 5.5:** Dati di profondità prima (sinistra) e dopo (destra) l'analisi dei dati

La funzione Matlab che è stata creata per eseguire questa analisi sui dati è descritta nell'appendice B (Paragrafo B.3).

### 5.2.3 Interpolazione

Per cercare di ricostruire i dati cancellati, in quanto errati, si ricorre ad un'interpolazione sui dati presenti in un singolo segmento ipotizzando che all'interno del segmento stesso la superficie sia regolare.
È stato usato il comando Matlab regress$(d, X)$ che restituisce un vettore di coefficienti stimato con una regressione multilineare della risposta in $d$ dei predittori $X$.
Si è usata un interpolazione lineare quindi del tipo:

$$d = b_1 \cdot x + b_2 \cdot y + b_3 \tag{5.1}$$

Al comando Matlab regress$(d, X)$ dobbiamo fornire il vettore $d$ contenente i punti

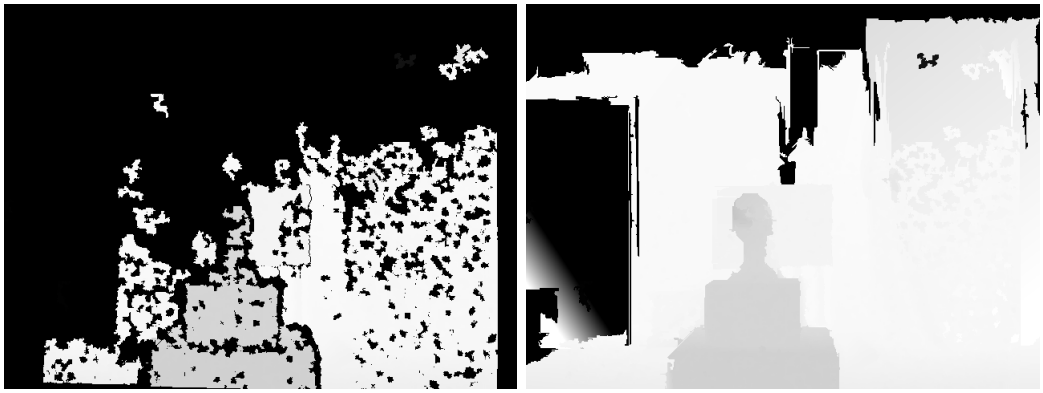

**Figura 5.6:** Dati di profondità prima (sinistra) e dopo (destra) l'interpolazione dei dati

noti all'interno del segmento considerato, cioè tutti i punti che a questo momento risultano avere un dato di profondità diverso da 0. In $X$ invece dovremo dare il valore dei moltiplicatori relativi ai punti in $d$, nella pratica le coordinate le coordinate $x$ e $y$ dei punti noti all'interno del segmento.

$$d = \begin{bmatrix} depth_1 \\ depth_2 \\ depth_3 \\ \vdots \\ depth_n \end{bmatrix} \quad X = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \tag{5.2}$$

L'equazione 5.2 mostra la struttura che devono avere i dati che devono essere forniti al comando Matlab regress; $n$ è il numero di punti che hanno un valore di profondità diverso da 0 all'interno del segmento considerato.

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{5.3}$$

Questa operazione restituisce come risultato un vettore (5.3) dei valori che devono essere inseriti nell'equazione cercata (5.1).
Una volta che si ha a disposizione l'equazione del piano che interpola i punti, si possono trovare i valori di profondità per i punti in cui mancano inserendo le coordinate di tali punti.
La figura 5.6 mostra il risultato di questa operazione di interpolazione.
La funzione Matlab che è stata creata per eseguire questa interpolazione è descritta nell'appendice B (Paragrafo B.4).

## 5.3 Creazione modello 3D

Avendo a disposizione i valori di profondità e la calibrazione del Kinect, che ci permette di proiettare questi punti nello spazio, è possibile creare un modello 3D costituito dalla nuvola dei punti a disposizione.
Per fare ciò si può creare un file con estensione ply (*Polygon File Format*) che rappresenta un formato in cui si possono salvare i dati di un modello 3D semplicemente scrivendo al suo interno l'elenco dei punti che compongono il modello con le loro caratteristiche.
Il modello generale per scrivere questo tipo di file è il seguente:

ply
format ascii 1.0
element vertex $N$
property float x
property float y
property float z
property float nx
property float ny
property float nz
property uchar diffuse_red
property uchar diffuse_green
property uchar diffuse_blue
end_header

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | $-y_1$ | $-z_1$ | 0,0 | 0,0 | 0,0 | $R_1$ | $G_1$ | $B_1$ |
| $x_2$ | $-y_2$ | $-z_2$ | 0,0 | 0,0 | 0,0 | $R_2$ | $G_2$ | $B_2$ |
| $x_3$ | $-y_3$ | $-z_3$ | 0,0 | 0,0 | 0,0 | $R_3$ | $G_3$ | $B_3$ |
| | $\vdots$ | | | | | | | |
| $x_N$ | $-y_N$ | $-z_N$ | 0,0 | 0,0 | 0,0 | $R_N$ | $G_N$ | $B_N$ |

Il parametro $N$ presente nella terza riga del file di testo deve essere sostituito con in numero di punti che si vanno ad elencare all'interno del file stesso. Il valore di $z$ è pari al dato di profondità mentre i valori $x$ e $y$ non sono le coordinate del pixel in questione ma invece sono i valori che si ottengono dalla proiezione dei punti nello spazio 3D.

I risultati che si ottengono, leggendo il file con un programma adatto (si è usato MeshLab), sono visibili nelle figure 5.7 e 5.8, nella prima sono stati usati solo i dati che sono stati ritenuti corretti dopo la procedura di analisi dei dati. Nel secondo caso (Figura 5.8) sono stati usati i dati ottenuti dopo l'interpolazione, seguendo però un semplice accorgimento cioè quello di analizzare i dati ottenuti ed eliminare quelli che risultano fuori dal range di profondità ($0 \div 1023$) e quelli che si discostano molto dal valore medio dei dati di profondità del segmento con-

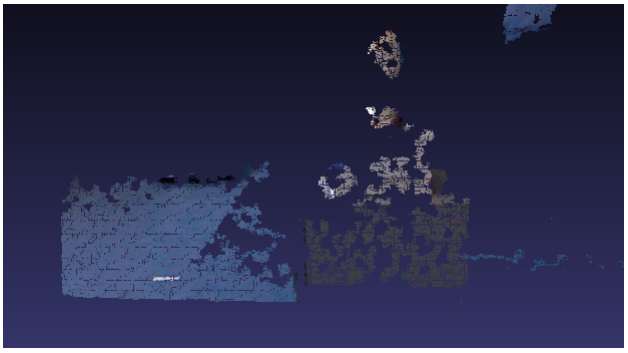

**Figura 5.7:** Modello 3D

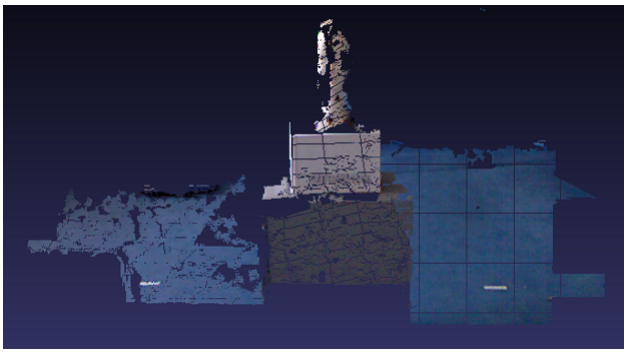

**Figura 5.8:** Modello 3D dati interpolati

siderato. Questa procedura di eliminazione dei dati che si discostano molto dal valore medio è necessaria perché è stato usato un metodo di interpolazione del primo ordine che quindi può commettere errori nelle zone in cui i dati sono molto diradati e nelle zone in cui sono presenti superfici curve.

## 5.4 Modello 3D con 2 Kinect

Il secondo Kinect fino ad ora è stato visto come una fonte di interferenza ma è possibile considerarlo come una fonte di dati aggiuntiva, utile per creare un modello 3D più completo e più accurato.

Naturalmente per poter usare i dati del secondo Kinect il sistema costituito dai due dispositivi deve essere calibrato prima dell'utilizzo in modo da poter calcolarne i parametri estrinseci ($R_{Stereo}$ e $T_{Stereo}$). Quando si ha la calibrazione del sistema stereo (Figura 5.9) si seguono le seguenti operazioni:

- proiettare i dati del primo Kinect nello spazio 3D utilizzando i parametri intrinseci del dispositivo;

**Figura 5.9:** Visione del sistema con 2 Kinect

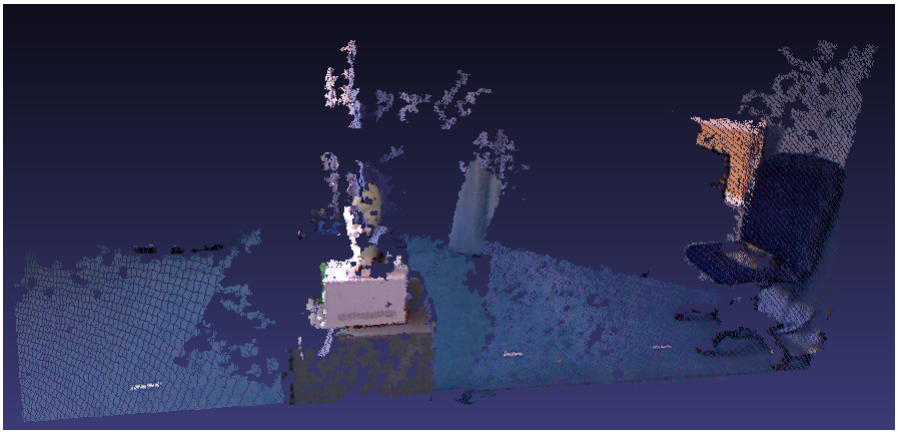

**Figura 5.10:** Modello 3D con i dati di 2 Kinect

- proiettare nello spazio 3D i dati del secondo Kinect con lo stesso procedimento;

- trasformare i punti nello spazio relativi al secondo dispositivo tramite la matrice di rotazione ($R$) e il vettore di traslazione ($T$) in modo che risultino nello stesso sistema di riferimento utilizzato dal primo dispositivo;

Quello che si ottiene è una nuvola di punti costituita dai dati di entrambi i dispositivi.

Il risultato che si ottiene fondendo i dati dei due Kinect è quello mostrato in figura 5.10.

### 5.4.1 Valutazione del modello 3D

La prima cosa che deve essere valutata è se esiste un miglioramento del modello 3D quando si aggiungono i dati del secondo Kinect. Sicuramente il numero di punti che compongono il modello aumenta, ma quello che si vuole capire è se c'è un miglioramento della precisione con cui il modello è costruito.
Per fare questo si devono considerare separatamente i due casi, cioè:

- modello 3D costituito dai dati provenienti solo dal primo Kinect;
- modello 3D costituito dai dati provenienti da entrambi i dispositivi.

Per cercare di avere una valutazione su un numero di punti significativo si ricorre ad una interpolazione dei dati forniti dai dispositivi. Per fare ciò si deve seguire la seguente procedura:

1. riproiettare i dati del modello considerato su una camera di riferimento. Si userà quella del secondo Kinect per cui anche in questo caso dovranno esseri usati i parametri della calibrazione stereo;
2. l'interpolazione è necessaria dal momento che molti dei punti riproiettati si trovano su coordinate non intere dell'immagine. Considerando la differenza di colore e di profondità si possono unire due punti che apparterranno con molta probabilità allo stesso oggetto.
   Nell'analisi bisogna tener conto anche della densità dei modelli 3D.

Per fare l'analisi sui due modelli si è calcolato l'MSE (paragrafo 4.3.1 a pag. 55) dei punti che si sono ottenuti dall'interpolazione.
Per ottenere un risultato più preciso si è ricorso ad un equalizzazione dei dati in modo che abbiano la stessa luminosità dell'immagine di riferimento. Per fare ciò si è utilizzato il comando Matlab histeq($I, HGRAM$) che modifica l'immagine $I$ fornita in modo che abbia un istogramma simile a quello fornito, cioè quello dell'immagine di riferimento, quindi $HGRAM$ è l'istogramma dell'immagine a colori del secondo Kinect.
Il comando histeq lavora solo con immagini in due dimensioni quindi bisogna trasformare le immagini formate dai dati e anche quella di riferimento da una matrice RGB a una matrice di due dimensioni; per arrivare a questo risultato si è considerato il solo dato di luminanza di una matrice YUV che non è altro che la media delle tre componenti RGB ed è il parametro che contiene la maggior parte

| PROVA 1 | Kinect 1 | Kinect 1 + 2 |
|---|---|---|
| MSE | 0,0986 | 0,0238 |
| PSNR | 10,0592 | 16,2304 |
| N | 67.646 | 141.906 (67.646 + 74.260) |
| PROVA 2 | Kinect 1 | Kinect 1 + 2 |
| MSE | 0,1380 | 0,0821 |
| PSNR | 8,6012 | 10,8586 |
| N | 94.712 | 155822 (94.712 + 61.110) |
| PROVA 3 | Kinect 1 | Kinect 1 + 2 |
| MSE | 0,0447 | 0,0184 |
| PSNR | 13,4977 | 17,3535 |
| N | 92.271 | 179.561 (92.271 + 87.290) |

**Tabella 5.1:** Risultati valutazione modelli 3D

dell'informazione.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(I(p) - I_w(p))^2 \tag{5.4}$$

Il calcolo dell'MSE, e conseguentemente anche del PSNR, nei due casi è stato fatto solo sui pixel che in entrambi i casi mostrano un dato diverso da 0, questo permette di eseguire il calcolo sullo stesso numero di punti in modo che i due MSE trovati siano perfettamente confrontabili.

$$PSNR = 20 \cdot log_{10}\left(\frac{MAX\{I\}}{\sqrt{MSE}}\right) = 10 \cdot log_{10}\left(\frac{1}{MSE}\right) \tag{5.5}$$

Nel calcolo del PSNR (5.5) bisogna mettere 1 al numeratore in quanto dopo l'equalizzazione rappresenta il valore massimo che la luminanza dell'immaggine può assumere. Se non si esegue l'equalizzazione bisogna usare il valore massimo che la luminanza poteva assumere, cioè 255, ricordando di elevarlo al quadrato.

I risultati ottenuti, mostrati nella tabella 5.1, mostrano che costruendo il modello 3D con i dati di entrambi i Kinect si ottiene un miglioramento rispetto al modello costruito usando un solo dispositivo (l'MSE diminuisce e quindi il PSNR aumenta) inoltre aumenta anche la densità del modello stesso.

# Capitolo 6

# Conclusioni

In questo lavoro di tesi si è analizzato il dispositivo Kinect di Microsoft partendo dall'idea di poterlo utilizzare come sensore di profondità in un veicolo autonomo.

Analizzando le varie acquisizioni effettuate si è potuto osservare il diverso funzionamento del dispositivo in presenza di differenti condizioni di illuminazione. Si è notato che il dispositivo, essendo stato progettato come applicazione ludica da utilizzare in ambiente interno, funziona in maniera corretta in condizioni di illuminazione controllata. In ambiente esterno, con forte presenza di luce solare, il Kinect risulta accecato e si può notare che una grande quantità di dati di profondità è mancante perché la luce solare interferisce sulla stessa frequenza in cui lavora l'emettitore a infrarossi. Questa perdita di dati avviene anche quando il Kinect si trova in un ambiente interno ma viene colpito dalla luce solare che proviene dall'esterno, in maniera diretta o anche indiretta dopo essere stata riflessa da una superficie, ad esempio da un muro bianco.
Queste considerazioni portano alla conclusione che il dispositivo Kinect può essere usato come sensore di profondità solo in condizioni di illuminazione controllata.
Considerando tutte le acquisizioni effettuate si è notato anche che in totale assenza di luce i dati di profondità vengono comunque acquisiti dal dispositivo in maniera corretta. Quindi si può pensare di utilizzare il dispositivo come sensore di profondità su un veicolo che ha la necessità di muoversi in condizioni di totale assenza di luce.

Considerando il sistema costituito dal Kinect e da una fotocamera laterale usata come riferimento, si è trovata una relazione tra due parametri che caratterizzano le acquisizioni, l'MSE calcolato grazie all'immagine della fotocamera laterale ed il valore di una metrica unsupervised che considera la segmentazione dell'immagine a colori del Kinect valutandone la consistenza utilizzando l'immagine di profondità.
L'andamento trovato può essere usato come discriminatore in un'applicazione real-time, infatti avendo a disposizione i dati del Kinect può essere eseguita la

segmentazione dell'immagine a colori e una volta calcolata la metrica, presentata in questa tesi, si potrà avere una stima dell'MSE e quindi della bontà dell'acquisizione effettuata. Questa stima può servire per scegliere se utilizzare i dati forniti dal dispositivo o effettuare una nuova acquisizione mediando poi i dati con l'intento di migliorarne la qualità.
Naturalmente per poter utilizzare questa stima in applicazioni real-time deve essere implementata una versione della segmentazione meno costosa a livello computazionale.

Considerando il sistema costituito da una coppia di Kinect si è potuto osservare l'interferenza creata dai due emettitori infrarossi contemporaneamente attivi. Si è osservato come l'interferenza varia in base all'angolazione con cui sono posizionati i due dispositivi e si è cercato di trovare una soluzione.
Attraverso il procedimento descritto in questa tesi si è riusciti a riconoscere i dati di profondità corrotti dall'interferenza e che quindi devono essere scartati. Inoltre si è riusciti a recuperare alcuni dati mancanti grazie ad un'operazione di interpolazione sui dati considerati validi.
Queste operazioni di analisi e di interpolazione permettono di creare un modello 3D contenente informazioni più corrette rispetto a quello che si otterrebbe con i dati del Kinect senza considerare il problema dell'interferenza.

Il sistema costituito dai due dispositivi Kinect è stato osservato anche come un vero sistema di visione stereo, considerando il secondo Kinect non solo come fonte di interferenza ma come un vero e proprio dispositivo che fornisce dei dati di profondità validi.
Si ha a disposizione due serie di dati che, analizzate entrambe per eliminare l'interferenza, permettono di ottenere un modello 3D più preciso.
Il modello 3D creato dai dati di entrambi i dispositivi è una *nuvola di punti* costituita da più dati rispetto a quelli che costituiscono la *nuvola di punti* creata dai dati del solo primo Kinect.
La maggior precisione del modello creato con i dati di entrambi i dispositivi è stata valutata confrontando l'MSE calcolato sui dati dei due modelli 3D (quello creato da entrambi i dispositivi e quello con i soli dati del primo) utilizzando come immagine di riferimento quella prodotta dalla Color Camera del secondo Kinect.
Si può concludere quindi che utilizzando un sistema costituito da due Kinect, sistema che deve essere debitamente calibrato, si ottiene un modello 3D costituito da più punti e più preciso di quello che si otterrebbe considerando solo un singolo dispositivo.

# Appendice A

# Calibrazione Kinect

Il programma di calibrazione del Kinect, RGBDemo, genera tre file contenenti i parametri calcolati.

Il file *kinect_calibration.yml* contiene i parametri per il viewer ed è così composto:

```
%YAML:1.0
rgb_intrinsics: !!opencv-matrix
  rows: 3
  cols: 3
  dt: d
  data: [ 5.5461841183490526e+02, 0., 3.0422459059313053e+02,
      0., 5.5125565919373912e+02, 2.7004649992905178e+02,
      0., 0., 1. ]
rgb_distortion: !!opencv-matrix
  rows: 1
  cols: 5
  dt: d
  data: [ 2.9817240531974626e-01, -1.0009785061688954e+00,
      1.2567536930217064e-03, -2.0343169326131269e-03,
      7.2011667135214730e-01 ]
depth_intrinsics: !!opencv-matrix
  rows: 3
  cols: 3
  dt: d
  data: [ 6.2106589469908863e+02, 0., 3.1779471156750293e+02,
      0., 6.1727076438214544e+02, 2.3523829813881366e+02,
      0., 0., 1. ]
depth_distortion: !!opencv-matrix
  rows: 1
```

```
   cols: 5
   dt: d
   data: [ -1.7882889851247255e-01, 1.2110601253482915e+00,
         4.8522314018299038e-03, 2.3661219219828105e-03,
         -2.9158730518641054e+00 ]
R: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 9.9998818436088266e-01, -1.5704868928830226e-03,
         -4.6005118785440394e-03, 1.5508341673424564e-03,
         9.9998967107278325e-01, -4.2723133232789234e-03,
         4.6071739722681844e-03, 4.2651282121579891e-03,
         9.9998029112044162e-01 ]
T: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [ 2.5161255262803665e-02, -3.7784821691766755e-04,
         -1.1295281949158680e-03 ]
rgb_size: !!opencv-matrix
   rows: 1
   cols: 2
   dt: i
   data: [ 640, 480 ]
raw_rgb_size: !!opencv-matrix
   rows: 1
   cols: 2
   dt: i
   data: [ 640, 480 ]
depth_size: !!opencv-matrix
   rows: 1
   cols: 2
   dt: i
   data: [ 640, 480 ]
raw_depth_size: !!opencv-matrix
   rows: 1
   cols: 2
   dt: i
   data: [ 640, 480 ]
depth_base_and_offset: !!opencv-matrix
   rows: 1
```

```
  cols: 2
  dt: f
  data: [ 7.50000030e-02, 1090. ]
```

I file *calibration_rgb.yaml* e *calibration_depth.yaml* contengono i dati per la compatibilità con ROS (*Rugged Operating System*).

Il file *calibratione_rgb.yaml* è strutturato nel seguente modo:

```
%YAML:1.0
camera_matrix: !!opencv-matrix
  rows: 3
  cols: 3
  dt: d
  data: [ 5.5461841183490526e+02, 0., 3.0422459059313053e+02,
    0., 5.5125565919373912e+02, 2.7004649992905178e+02,
    0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
  rows: 1
  cols: 5
  dt: d
  data: [ 2.9817240531974626e-01, -1.0009785061688954e+00,
    1.2567536930217064e-03, -2.0343169326131269e-03,
    7.2011667135214730e-01 ]
rectification_matrix: !!opencv-matrix
  rows: 3
  cols: 3
  dt: f
  data: [ 1., 0., 0., 0., 1., 0., 0., 0., 1. ]
projection_matrix: !!opencv-matrix
  rows: 4
  cols: 4
  dt: f
  data: [ 5.53212280e+02, 4.39621598e-01, -3.06773804e+02,
    -1.36112652e+01, -2.10809183e+00, -5.50096252e+02,
    -2.72392365e+02, 5.13316095e-01, -4.60051186e-03,
    4.27231332e-03, -9.99980271e-01, 1.12952816e-03,
    0., 0., 0., 1. ]
```

Il file *calibratione_depth.yaml* è strutturato nel seguente modo:

```
%YAML:1.0
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 6.2106589469908863e+02, 0., 3.1779471156750293e+02,
     0., 6.1727076438214544e+02, 2.3523829813881366e+02,
     0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ -1.7882889851247255e-01, 1.2110601253482915e+00,
     4.8522314018299038e-03, 2.3661219219828105e-03,
     -2.9158730518641054e+00 ]
rectification_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: f
   data: [ 1., 0., 0., 0., 1., 0., 0., 0., 1. ]
projection_matrix: !!opencv-matrix
   rows: 4
   cols: 4
   dt: f
   data: [ 6.21065918e+02, 0., -3.17794708e+02, 0.,
     0., -6.17270752e+02, -2.35238297e+02, 0.,
     0., 0., -1., 0.,
     0., 0., 0., 1. ]
```

I dati riportati in questi file sono relativi ad uno dei Kinect usati.

# Appendice B

# Codici sorgente

## B.1 Acquisizione dati dal Kinect

Un esempio di utilizzo delle funzioni descritte nel paragrafo 3.7 (pag. 43) è il seguente, nel quale i dati forniti dal Kinect sono salvati in due file di testo, uno per i dati della Color Camera e uno per quelli della Depth Camera, per poi poterli usare in una successiva analisi.

**Codice B.1:** Programma C per acquisire i dati dal Kinect

```
1           ...
2  #include <libfreenect/libfreenect.h>
3  #include <libfreenect/libfreenect_sync.h>
4           ...
5  unsigned char **data;
6  unsigned short **depth;
7  unsigned int timestamp;
8           ...
9
10 int main()
11 {
12
13    ...
14
15    int flag, frame_num;
16    char n[1024];
17
18    data=(unsigned char**)calloc(1,sizeof(unsigned char*));
19    for (i=0;i<1;i++)
20          data[i]=(unsigned char*)calloc((640*480*3),
21                sizeof(unsigned char);
```

```
22
23    depth=(unsigned short**)calloc(1,sizeof(unsigned short*));
24    for (i=0;i<1;i++)
25          depth[i]=(unsigned short*)calloc(640*480),
26                sizeof(unsigned short);
27
28    ...
29
30    flag = 0;
31    frame_num = 0;
32
33    while(flag==0)
34    {
35       printf("Print \"a\" to save the frame (\"q\" to exit):");
36       scanf("%s",n);
37
38       if(n[0]=='a')
39       {
40          frame_num++;
41
42          // Richiesta dati Kinect
43
44          freenect_sync_get_video((void**)data, &timestamp, 0,
45                FREENECT_VIDEO_RGB);
46          freenect_sync_get_depth((void**)depth, &timestamp, 0,
47                FREENECT_DEPTH_10BIT);
48
49          // Salvataggio dati Color Camera
50
51          for(i=0;i<=(640*480*3);i++)
52             a[i]=(float)data[0][i];
53          outfile=fopen("/immagini/data.txt","a");
54          for(i=0;i<640*480*3;i++)
55             fprintf(outfile,"%f\n",a[i]);
56          fclose(outfile);
57
58          // Salvataggio dati Depth Camera
59
60          for(i=0;i<=(640*480);i++)
61             b[i]=depth[0][i];
62          outfile=fopen("/immagini/depth.txt","a");
63          for(i=0;i<640*480;i++)
```

```
64              fprintf(outfile,"%d\n",b[i]);
65            fclose(outfile);
66
67            ...
68
69        }
70        else
71        {
72            if(n[0]=='q')
73            {
74                flag = 1;
75            }
76        }
77    }
78
79    outfile=fopen("/home/daniele/Scrivania/immagini/num.txt","w");
80    fprintf(outfile,"%d",frame_num);
81    fclose(outfile);
82
83    for (i=0;i<1;i++)
84        free(data[i]);
85    free(data);
86    for (i=0;i<1;i++)
87        free(depth[i]);
88    free(depth);
89 }
```

I file *data.txt* e *depth.txt* contengono i dati forniti dal dispositivo mentre il file *num.txt* contiene il numero di acquisizioni effettuate.
Utilizzando l'opzione "a" nel comando *fopen* si posiziona il puntatore alla fine del file in modo da non sovrascrivere i dati delle acquisizioni precedenti.
La necessità di salvare i dati provenienti dal Kinect in file di testo si presenta per due semplici ragioni, salvare le acquisizioni per un uso futuro ed eseguire un analisi con un altri strumenti, ad esempio Matlab.

In Matlab si usano una serie di isruzioni che caricano i dati leggendoli dai file di testo creando matrici con le giuste dimensioni.

**Codice B.2:** Codice Matlab per caricare i dati del Kinect

```
1 [n] = textread('num.txt','%d');
2 [data] = textread('data.txt','%f');
```

```
3  [depth] = textread('depth.txt','%d');
4
5  mdata = zeros(480,640,3);
6  w = 1;
7  for i=1:480
8     for j=1:640
9           mdata(i,j,1) = data(w);
10          mdata(i,j,2) = data(w+1);
11          mdata(i,j,3) = data(w+2);
12
13          w = w + 3;
14    end
15 end
16
17 mdepth = zeros(480,640);
18 w = 1;
19 for i=1:480
20    for j=1:640
21          mdepth(i,j) = depth(w);
22
23          w = w + 1;
24    end
25 end
```

I dati della prima acquisizione sono stati salvati nelle variabili *mdata* e *mdepth*. Per utilizzare i dati delle altre acquisizioni bisogna spostarsi all'interno delle variabili *data* e *depth*, che contengono tutte le acquisizioni effettuate, considerando che il numero totale di acquisizioni è dato dalla varabile *n*.

## B.2 Media alternativa

L'idea di media alternativa usata per fondere più acquisizioni della stessa scena si basa sull'analisi dei dati che si hanno per ogni singolo pixel eliminando quelli che si rivelano sicuramente sbagliati.
Il codice usato parte dalla variabile *mdepth_vet* che è una matrice di dimensioni 640x480x$q$ dove $q$ è il numero di acquisizioni considerate.

**Codice B.3:** Codice Matlab per fondere più acquisizioni della stessa scena

```
1  mdepth = zeros(480,640);
```

```
2
3  for i=1:480
4    for j=1:640
5      if ((max(mdepth_vet(i,j,:)))-(min(mdepth_vet(i,j,:))))<10
6        mdepth(i,j) = round(mean(mdepth_vet(i,j,:)));
7      else
8        if q==2
9          if mdepth_vet(i,j,1)==1023
10           mdepth(i,j) = mdepth_vet(i,j,2);
11         else
12           if mdepth_vet(i,j,2)==1023
13             mdepth(i,j) = mdepth_vet(i,j,1);
14           else
15             mdepth(i,j) = 0;
16           end
17         end
18       else
19         D = zeros(1,q);
20         for z=1:q
21           D(z) = mdepth_vet(i,j,z);
22         end
23         flag = 1;
24         while flag==1
25           A = D(1);
26           B = [];
27           for z=2:size(D,2)
28             if abs(D(1)-D(z))<10
29               A(end+1) = D(z);
30             else
31               B(end+1) = D(z);
32             end
33           end
34           if size(A,2)>size(B,2)
35             mdepth(i,j) = round(mean(A(:)));
36             flag = 0;
37           else
38             if size(B,2)==1
39               mdepth(i,j) = 0;
40               flag = 0;
41             else
42               D = zeros(1,size(B,2));
43               D(:) = B(:);
```

```
44                  end
45                end
46              end
47            end
48          end
49        end
50  end
```

La variabile *mdepth* contiene il risultato della media effettuata sulle $q$ acquisizioni considerate.

## B.3 Analisi dei dati

Per trovare i dati di profondità corrotti dall'interferenza causata dalla presenza di un secondo Kinect si è usata la seguente funzione:

**[*D*,*count*,*So*] = find_error(*data*,*depth*)**

- variabili di input:
  - *data*: matrice dell'immagine a colori del Kinect;
  - *depth*: matrice dei dati di profondità riproiettati nel sistema di riferimento della Color Camera;
- variabili di output:
  - *D*: matrice dei dati di profondità in cui sono stati cancellati i dati corrotti;
  - *count*: numero di segmenti trovati;
  - *So*: segmentazione dell'immagine a colori.

**Codice B.4:** Codice Matlab per cancellare i dati corrotti da interferenza

```
1 function [D, count, So]=find_error(data, depth)
2
3 data_d = double(data);
4 Sdata = data_d(:,:,1)+data_d(:,:,2)*256+data_d(:,:,3)*256*256;
5
6 imwrite(data/256, 'data.ppm', 'ppm');
```

```
7
8  system('./segment 0.5 100 200 data.ppm data_seg.ppm');
9
10 S = imread('data_seg.ppm');
11
12 S = double(S);
13 Sb = S(:,:,1)+S(:,:,2)*256+S(:,:,3)*256*256;
14 vet_val_S = unique(Sb(:));
15
16 So = zeros(480,640);
17 count = 0;
18 for col=vet_val_S'
19   ind = find(Sb(:)==col);
20   So(ind) = count;
21   count = count+1;
22 end
23
24 D = zeros(480,640);
25
26 for i=0:(count-1)
27   ind2 = find(So(:)==i);
28   Cd = depth(ind2);
29   vet_val_Cd = unique(Cd(:));
30
31   N = size(Cd,1);
32   M = size(vet_val_Cd,1);
33
34   % Creo istogramma relativo al singolo segmento
35   isto = zeros(M,2);
36   isto(:,1) = vet_val_Cd;
37   for j=1:M
38     isto_n = size(find(Cd(:)==isto(j,1)));
39     isto(j,2) = isto_n(1);
40   end
41
42   if isto(1,1)==0
43     N1 = N-isto(1,2);
44     M = M-1;
45     isto(1,:) = [];
46   else
47     N1 = N;
48   end
```

```
49
50    if M~=0
51      a = find(isto(:,2)==max(isto(:,2)));a=a(1);
52      th = 0.5*isto(a,2);th=th(1);
53      ind_th = find(isto(:,2)>th);
54      diff_th = zeros(size(ind_th));
55      w = 1;
56      for j=ind_th'
57        diff_th(w) = isto(a,1)-isto(j,1);
58        w = w+1;
59      end
60      if max(diff_th)<20
61        D(ind2) = depth(ind2);
62      else
63        isto2 = isto;
64        isto2(a,2) = 0;
65        b = find(isto2(:,2)==max(isto2(:,2)));b=b(1);
66        A = [];
67        B = [];
68        for z=1:M
69          dist_a = sqrt((isto(a,1)-isto(z,1))^2+...
70            (isto(a,2)-isto(z,2))^2);
71          dist_b = sqrt((isto(b,1)-isto(z,1))^2+...
72            (isto(b,2)-isto(z,2))^2);
73          if dist_a<dist_b
74            A(end+1) = isto(z,1);
75          else
76            B(end+1) = isto(z,1);
77          end
78        end
79        for z=ind2'
80          d = unique(ismember(A,depth(z)));
81          if d(end)==1
82            D(z) = depth(z);
83          end
84        end
85      end
86    end
87  end
```

## B.4 Interpolazione

Per interpolare i dati di profondità in modo da recupare i dati corrotti dall'interferenza si è usata la seguente funzione:

**[*D_int*] = interpolazione(*D*,*count*,*So*)**

- variabili di input:
  - *D*: matrice dei dati di profondità in cui sono stati cancellati i dati corrotti;
  - *count*: numero di segmenti trovati;
  - *So*: segmentazione dell'immagine a colori;
- variabili di output:
  - *D_int*: matrice dei dati di profondità interpolati.

**Codice B.5:** Codice Matlab per interpolare i dati di profondità

```
1  function [D_int]=interpolazione(D,count,So)
2
3  D_int = D;
4
5  for i=0:(count-1)
6    ind = find(So(:)==i);
7    D_so = zeros(480,640);
8    D_so(ind) = D(ind);
9    ind2 = find(D_so(:)~=0);
10   if size(ind2,1)>0
11     X = zeros(size(ind2,1),3);
12     Y = D(ind2);
13     w = 1;
14     for j=ind2'
15       x = floor((j-1)/480)+1;
16       y = mod((j-1),480)+1;
17
18       X(w,1) = x;
19       X(w,2) = y;
20       X(w,3) = 1;
21
22       w = w+1;
23     end
```

```
24
25      B = regress(Y,X);
26
27      Ym = mean(Y);
28
29      for j=ind'
30        if D(j)==0
31          x = floor((j-1)/480)+1;
32          y = mod((j-1),480)+1;
33          Dint = B(1)*x + B(2)*y + B(3);
34          if Dint<0
35            D_int(j) = 0;
36          else
37            if Dint>1023
38              D_int(j) = 0;
39            else
40              if abs(Dint-Ym)<5
41                D_int(j) = Dint;
42                D_so(j) = Dint;
43              end
44            end
45          end
46        end
47      end
48    end
49  end
```

# Bibliografia


[1] J. C. Bebel, B. L. Raskob, A. C. Parker, D. J. Bebel, *Managing Complexity in an Autonomous Vehicle*, in Aerospace and Electronic Systems Magazine, IEEE, 2008.

[2] S. Donati, *Electro-Optical Instrumentation - Sensing and Measuring with Lasers*, Ed. Prentice Hall, 2004.

[3] E. Bava, R. Ottoboni, C. Svelto, *Fondamenti della Misurazione*, Società Editrice Esculapio, 2005.

[4] IHS iSuppli, *The teardown: The Kinect for Xbox 360*, IET Engineering Technology, vol. 6, pp. 94-95, 2011.

[5] J. Shutton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, *Real-Time Human Pose Recognition in Parts from Single Depth Images*, at Computer Vision and Pattern Recognition (CVPR), IEEE, 2011.

[6] R. Szeliski, *Computer Vision: Algorithms and Applications*, Ed. Springer, 2011.

[7] A. Fusiello, *Visione computazionale. Appunti delle lezioni*, pubblicato a cura dell'autore, 2008.

[8] R. Haralick, L. Shapiro, *Survey: Image segmentation techniques*, in Computer Vision, Graphics and Image Processing, vol. 29, pp. 100-132, 1985.

[9] C. Rosenberger, K. Chehdi, *Genetic fusion: Application to multicomponents image segmentation*, at International Conference on Acoustic, Speech and Signal Processing (ICASSP), IEEE, 2000.

[10] H. Zhang, J. E. Fritts, S. A. Goldman, *Image Segmentation Evaluation: A Survey of Unsupervised Methods*, in Computer Vision and Image Understanding, vol. 110, pp. 260-280, 2008.

[11] P. F. Felzenswalb, D. P. Huttenlocher, *Efficient Graph-Based Image Segmentation*, International Journal of Computer Vision, vol. 59, 2004.

[12] J. B. MacQueen, *Some Methods for Classification and Analysis of Multivariate Observations*, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 1967.

[13] S. Milani, G. Calvagno, *Joint Denoising and Interpolation of Depth Maps for MS Kinect Sensors*, at International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2012.

[14] S. B. Gokturk, H. Yalcin, C. Bamji, *A Time-Of-Flight Depth Sensors - System Description, Issue and Solutions*, in Proc. of CVPRW 2004, vol. 3, p. 35, 2004.

[15] B. Huhle, T. Schairer, P. Jenke, W. W. Staíer, *Fusion of range and color images for denoising and resolution enhancement with non-local filter*, Comput. Vis. Image Underst., vol. 114, pp. 1336-1345, 2010.

[16] S. Milani, G. Calvagno, *A depth image coder based on progressive silhouettes*, IEEE Signal Process. Lett., vol. 17, no. 8, pp. 711-714, 2010.

[17] S. Milani, E. Frigerio, M. Marcon, S. Tubaro, *Denoising Infrared Structured Light DIBR Signals Using 3D Morphological Operators*, at 3DTV-CON, 2012.

[18] R. C. Gonzalez, R. E. Woods, *Digital image processing*, IEEE Transactions on Systems, Man and Cybernetics, 2008.