Università degli Studi di Padova
Dipartimento di Tecnica e Gestione dei Sistemi Industriali

Corso di Laurea in Ingegneria Meccanica e Meccatronica

Tesi di Laurea

Studio di sistemi di ricostruzione di scene basato su sensori tridimensionali

A Study on Scene Reconstruction Systems Based on Three-Dimensional Sensors

Laureando:
Alvise De Biasio
Matricola 1073242

Relatore:
Dott. Stefano Ghidoni – DEI Dipartimento di Ingegneria dell'Informazione

Correlatore:
Chiar.ma Prof.ssa Monica Reggiani

Anno Accademico 2015 – 2016

Alla mia famiglia, ai miei amici
e a te caro lettore e cara lettrice

Indice

Prefazione

Questo lavoro è per persone a cui diverte ragionare e che, come capita a me, provano soddisfazione nella scoperta di come funzionano le cose.

Quando ci si addentra nel mondo dell'ingegneria, passato un primo periodo di allenamento alla complessità, si viene affascinati dalla ricerca continua del principio primo, del fattore scatenante, della nuova tecnica ma soprattutto delle eleganti regole che governano i sistemi complessi.

Sensazioni di questo tipo sono all'ordine del giorno quando si opera nel campo della robotica e in particolare per quanto riguarda la Computer Vision. Queste tematiche che a prima vista appaiono tecniche e sofisticate le ritroviamo in realtà in molteplici campi della nostra vita quotidiana. Specificatamente, la Ricostruzione Tridimensionale ha subito una evoluzione che ha posto le basi per lo sviluppo di sistemi articolati molto attuali. In effetti, algoritmi di 3D Reconstruction sono ritrovabili in architetture molto diverse tra loro come ad esempio dispositivi medici, sistemi di automazione e controllo industriali, Computer Graphics and Gaming.

La presente tesi ha lo scopo di esaminare sotto un profilo scientifico quanto accade nei moderni sistemi di ricostruzione 3D. Per ottenere questo obiettivo, la ricerca si è soffermata sui principi logico-matematici su cui si basa l'architettura di riferimento senza perdere di vista lo sviluppo dal punto di vista progettuale. Infine, sono stati presentati i framework di sviluppo più diffusi allo stato attuale.

Esiste ampia letteratura in rete su buona parte dei temi descritti non sempre in maniera organica. Particolare attenzione è stata data nella presentazione della struttura dell'elaborato secondo un percorso autoconsistente, proponendo i concetti sotto una forma intuitiva contestualmente alla caratterizzazione scientifica.

Ringraziamenti

Un ringraziamento particolare a tutti coloro che si avventureranno nella lettura di questa tesi, spero che quanto prodotto possa appassionarvi e/o fornirvi nuovi spunti e chiavi di lettura.

Desidero ringraziare innanzitutto il Professor S. Ghidoni per avermi dato l'opportunità di sviluppare questo primo lavoro di ricerca. Profondo conoscitore della materia e paziente riferimento per tutta la durata del percorso, mi ha fornito preziosi suggerimenti stimolando una naturale attitudine alla curiosità.

A seguire, sono lieto di ringraziare la Prof.ssa M. Reggiani per la sua didattica piacevole e coinvolgente, attraverso la quale ho avuto modo di interagire con la programmazione orientata agli oggetti in C++ e con alcuni tra i più interessanti temi specificatamente alla logica di sviluppo.

Desidero inoltre ringraziare tutte le persone con le quali ho avuto modo di confrontarmi in questo periodo, per l'incoraggiamento e per aver contribuito in modo propositivo alla stesura.

Infine, un grazie di cuore alla mia famiglia, soprattutto a mia mamma, ai miei amici e a tutte le persone a me più care.

Introduzione

Negli ultimi decenni la tecnologia ha assunto un ruolo sempre più preponderante nella vita quotidiana. Lo scaling dei componenti elettronici ha reso possibile l'utilizzo di infrastrutture hardware che permettono di ottenere risultati complessi in tempi rapidi e a costi accessibili.

Sono stati introdotti nel mercato nuovi sistemi, come ad esempio Microsoft Kinect Fusion, per l'elaborazione di immagini tridimensionali più performanti che hanno potuto attingere da un ambiente maturo e multidisciplinare articolato quale la Computer Vision.

La Visione Artificiale – Computer Vision – è una disciplina che si prefigge l'obiettivo di fornire a uno specifico dispositivo l'abilità di estrarre informazioni dalla realtà circostante. Essa sfrutta particolari tecniche di Image Processing e Artificial intelligence per elaborare immagini o sequenze di immagini acquisite attraverso specifici sensori.

Riuscire ad identificare con poche parole i principali domini della Visione Artificiale è complicato perché i campi d'interesse sono molteplici ed in continua espansione. In particolare, spaziano dal puro intrattenimento, alla robotica industriale esplorando problematiche inerenti a sicurezza, medicina, architettura, arte e istruzione. Tali tematiche sono solo alcune tra le più note ma ne esistono molte di ibride nate attraverso una combinazione delle tecniche sopra descritte.

Inoltre, la cosa che ogni volta sorprende è il fatto che ogni innovazione inevitabilmente vada ad influenzare almeno un'altra disciplina. La tecnologia e l'informatica si fondono diventando il "pensiero laterale" che guida l'innovazione globale. Così capita, ad esempio, che gruppi di ricerca e sviluppo creino un sensore così sofisticato da dare nuova linfa al mondo della videosorveglianza o delle console di gioco, che rinnova le regole sugli addestramenti per i piloti con simulatori di volo, oppure crea nuove tecniche innovative per la chirurgia tramite operazioni in artoscopia o la riabilitazione da traumi fisici attraverso un processo assolutamente dinamico e inatteso. Il tutto presentato con della grafica accattivante ed informazioni di natura statistica utili e talvolta indispensabili. Questo capita perché ogni tecnica è sempre più interconnessa e

Figura 1: Computer Vision [14].

l'innovazione si riverbera su aree di business adiacenti.

Allo stato attuale, le principali aree di ricerca sono le seguenti:

- Object Recognition;
- People Surveillance;
- Motion Analysis;
- 3D Reconstruction;
- Fusione e/o combinazione dei precedenti.

I benefici derivanti dall'utilizzo di tecnologie così avanzate sono evidenti. La costanza, l'affidabilità e l'oggettività delle architetture tecnologiche consentono di agevolare l'intervento umano in processi industriali che necessitano di operazioni di controllo multiplo, ripetitivo o basate su superamenti di soglia parametrizzati attraverso gestione di eventi. Condizioni ambientali proibitive caratterizzate ad esempio da temperature critiche, agenti chimici pericolosi o sistemi ad elevata automazione non costituiscono più un rischio per la sicurezza dei lavoratori. Inoltre grazie all'elevata precisione e velocità di controllo è possibile intervenire su oggetti di dimensioni molto piccole eseguendo mansioni complesse.

In particolare i sistemi per l'acquisizione di dati tridimensionali hanno raggiunto recentemente livelli di accuratezza e di praticità molto elevati riuscendo a manipolare un numero sempre più rilevante di informazioni. Questo continuo sviluppo ha permesso di esplorare campi sempre più vasti annullando di volta in volta i problemi di scalabilità e di efficienza tecnologica.

Struttura dell'elaborato

Data la vastità degli argomenti inerenti la 3D reconstruction nel campo della Computer Vision, uno dei problemi fondamentali è stato il circoscrivere le aree di interesse. Di conseguenza, durante le fasi di stesura degli argomenti principali si è scelto di approfondire solo alcune delle ramificazioni in modo tale da rendere l'elaborato autoconsistente.

Scopo principale di questa tesi è studiare dal punto di vista hardware e software i principali sistemi di ricostruzione di scene basati su sensori tridimensionali. Si cercherà di esplorare i principali scenari ponendo attenzione alle metodologie che hanno permesso lo sviluppo su campi di applicabilità specifici. Verificheremo inoltre le problematiche più ricorrenti e le soluzioni più accreditate, sempre considerando che si tratta di processi a forte impatto innovativo e soggette a miglioramenti continui.

Nello specifico, particolare attenzione sarà posta nell'analisi ed elaborazione di immagini 3D acquisibili attraverso Microsoft Kinect, esaminandone le caratteristiche tecniche ma portando nella lettura della presente tesi ad immaginare il dispositivo anche come possibile elemento catalizzatore che permetta uno sviluppo congiunto tra diverse tecnologie, quali ad esempio:

- 3D Reconstruction;
- 3D Printing;
- IoT (Internet of Things);
- Home Automation;
- Medecine and Surgery.

Questa tesi è strutturata come un cammino che porta il lettore ad una visione sempre più esaustiva della tecnica e della materia senza deviare dall'obiettivo di una specializzazione legata alla ricostruzione visuale 3D.

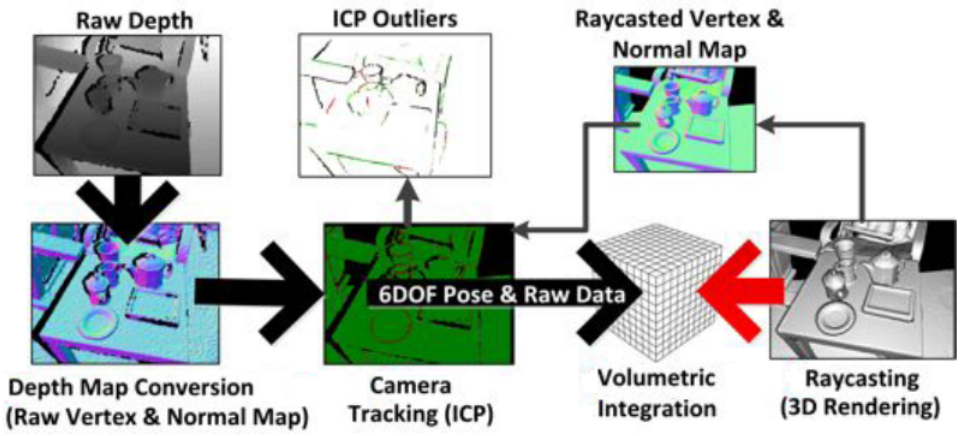

Figura 2: Pipeline del sistema di ricostruzione di scene Microsoft Kinect Fusion [18, 25, 27].

La prima parte ha lo scopo di livellare le competenze di base in forma introduttiva per poi specializzarsi sui rispettivi pilastri di competenza nei quali vengono esaminate le tecniche con maggior dettaglio.

Di seguito i capitoli relativi alla prima parte della tesi:

- Capitolo 1: si esamineranno le specifiche inerenti la codifica dell'informazione;
- Capitolo 2: saranno analizzate nel dettaglio le tecnologie hardware più frequentemente utilizzate nel campo della 3D reconstruction.

La seconda parte, successivamente ad una disamina delle peculiarità legate all'ottimizzazione dell'informazione spaziale, ha lo scopo di focalizzarsi su casi pratici utilizzando uno specifico algoritmo di ricostruzione della scena.

Di seguito i capitoli relativi alla seconda parte della tesi:

- Capitolo 3: saranno descritte le tecniche di filtraggio e interpolazione delle immagini tridimensionali;
- Capitolo 4: sarà presentata un approfondimento relativo all'associazione delle point clouds nello spazio tridimensionale specificatamente a Microsoft Kinect Fusion – Figura 2;
- Capitolo 5: si esamineranno brevemente le librerie e i tool specifici utilizzati nello scenario attuale.

Capitolo 1

Istantanee digitali

La vita quotidiana ci ha abituato ad osservare e ad interagire in un universo 3D. Tale visione si forma all'interno del nostro cervello attraverso un'acquisizione di immagini bidimensionali con una elaborazione di un processo di fusione denominato solitamente triangolazione. Questa abilità innata, che ci appare semplice e naturale, in realtà è molto complessa ed è l'unione di processi chimici e fisici attraverso una modalità simbiotica e di governo.

Di seguito, alcuni concetti preliminari strumentali alla costruzione di una architettura tecnica strutturata che li sviluppa ed utilizza.

1.1 Codifica dell'informazione

Consentire alle macchine di "vedere" e di interagire è sempre stato uno dei sogni dell'uomo e, proprio attraverso queste abilità sono state sviluppate tecnologie impensabili in passato. Le capacità di dispositivi di "percepire" e ricostruire lo spazio hanno permesso la nascita di sistemi che esplorano corpi solidi specializzando di volta in volta il sistema di acquisizione.

Gestire la profondità, lo zooming e/o la rotazione diventano a questo punto delle attività correlate alla modalità di memorizzazione delle immagini. Di conseguenza, tali modalità sono subordinate all'architettura e rappresentano sia il limite che la potenza su problematiche di scalabilità.

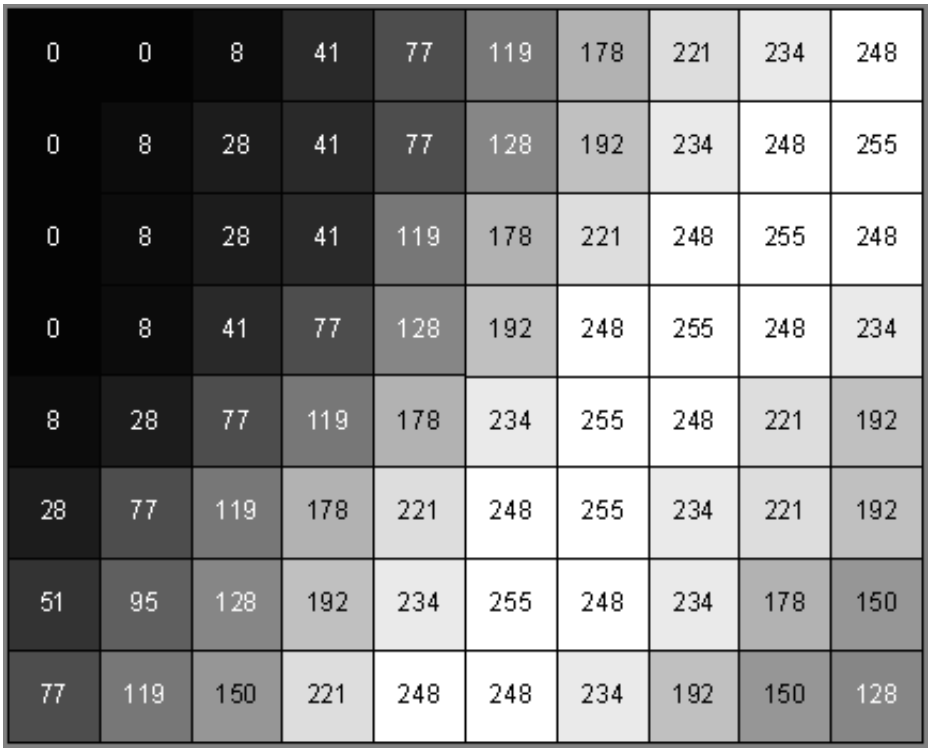

Figura 1.1: Codifica di un immagine di intensità a scala di grigi [17].

Il sistema di acquisizione delle immagini può utilizzare vari metodi di codifica [53] e quelli attualmente più utilizzati nella visione artificiale sono i seguenti:

Intensity Images Sono le immagini a cui siamo solitamente abituati e misurano la quantità di luce impressa in un dispositivo fotosensibile. È possibile fare un'analogia con le fotocamere che utilizzano o utilizzavano pellicole fotosensibili.

La codifica avviene attraverso delle matrici bidimensionali che contengono dei valori compresi tra 0 e 255, nel caso di immagini con scala di grigi – Figura 1.1. Nel caso dell'utilizzo del colore invece, è l'unione di tre matrici distinte, specializzate e in funzione dello spettro RGB.

3D Images In precedenza chiamate Range Images, possono stimare direttamente la struttura tridimensionale della scena decodificando forma e distanza acquisite attraverso specifici sensori come sonar, radar o laser.

Questa tecnica ha delle analogie con il metodo precedente per quanto riguarda la matrice nel sistema di codifica, tuttavia, l'intervallo dei

valori è compreso tra 0 e 4500 perché è strumentale alla memorizzazione in forma spaziale.

RGB-D Tale sistema, rappresenta una evoluzione dei due sistemi precedenti. La tecnica di memorizzazione è sempre su base matriciale ma ha la caratteristica di gestire la profondità assieme al colore come variante. Di conseguenza, tutte le tecniche di filtraggio e i vari algoritmi di interpolazione sono particolarmente agevolati potendo sfruttare metodologie consolidate da best-practices precedenti.

Le metodologie sopra enunciate, sono utilizzate in gran parte dei dispositivi di natura fisica e digitale. Ogni dispositivo ha caratteristiche di natura specifica che vengono specializzate attraverso una calibrazione focalizzata. In particolare, i parametri che caratterizzano la memorizzazione dell'immagine [53] sono i seguenti:

Ottico Caratterizzano l'ottica del sensore e quindi la parte fisica. Possono includere ad esempio il tipo di lente o la lunghezza focale, l'angolo di apertura o il campo di vista. È possibile considerare inoltre che a parità di caratteristiche dichiarate, esistono marchi internazionali specializzati che forniscono prestazioni anche molto differenti tra di loro;

Fotometrico Sono inerenti all'energia che raggiunge il sensore dopo essere stata riflessa dagli oggetti presenti nella scena. Possono includere il tipo di intensità e la direzione di illuminazione, proprietà di riflesso su alcune superfici o effetti della natura del sensore sulla quantità di luce che raggiunge i fotoricettori;

Geometrico Sono correlati alle caratteristiche dell'immagine nella conversione di una rappresentazione spaziale tridimensionale in una codifica bidimensionale. Possono includere l'orientazione della camera nello spazio o le distorsioni prospettiche introdotte dall'image processing.

Questi parametri, sono misurabili ma al tempo stesso sono correlati tra di loro. Di conseguenza, talvolta, la strategia di modifica è dipendente dalla sequenza con cui vengono effettuate le calibrazioni.

Inoltre, qualsiasi immagine, indipendentemente dalla modalità di memorizzazione, è soggetta ad una corruzione denominata rumore. Infatti immagini acquisite a parità di scena, condizioni ambientali e di architettura non sono mai identiche in quanto esistono delle fluttuazioni che introducono degli errori sia nella codifica che inevitabilmente nella parte di elaborazione. Tale rumore, comportando una alterazione dell'informazione acquisita, ha la necessità di essere filtrato attraverso metodi di analisi frequenziali.

Capitolo 2

Acquisizione di dati tridimensionali

Ogni sistema di ricostruzione di scene tridimensionali si basa su specifiche architetture ognuna delle quali ha dei propri punti di forza. In particolare, tali sistemi operano in prevalenza attraverso strutture hardware dedicate oppure attraverso algoritmi software che permettono una estrapolazione della profondità basati su modelli di tipo geometrico.

Scopo di questo capitolo è introdurre l'insieme delle strategie di acquisizione su cui si basano le architetture tecniche. Tali strategie si differenziano tra di loro appartenendo a insiemi di metodi differenti.

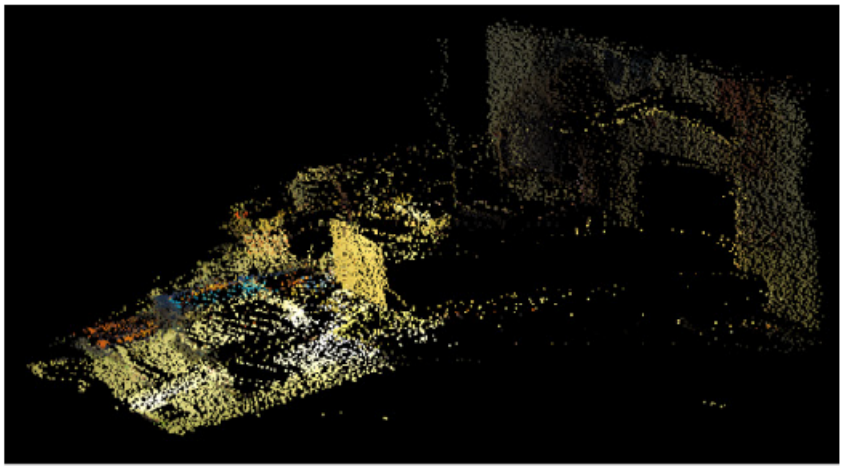

Figura 2.1: Dati tridimensionali raw [16].

Tabella 2.1: Metodi di ricostruzione 3D Shape from X [53].

Shape from	How many images	Method type
Stereo	2 or more	Passive
Motion	A sequence	Active/passive
Focus/defocus	2 or more	Active
Zoom	2 or more	Active
Contours	Single	Passive
Texture	Single	Passive
Shading	Single	Passive

Il primo insieme di metodi che esamineremo, è universalmente conosciuto come Shape from X [53] – Tabella 2.1. Il filo conduttore che caratterizza questa metodologia si basa sulla ricostruzione della struttura tridimensionale della scena a partire da una o più intensity images. Tale sistema ha la peculiarità di gestire alcuni automatismi di armonizzazione a seconda che vengano modificati, più o meno intenzionalmente, uno o più parametri del sistema di acquisizione per determinare la profondità spaziale. Questi metodi sono detti Attivi o Passivi.

Il secondo sistema che avremo modo di esaminare, si basa sull'opportunità di effettuare una scannerizzazione su base tridimensionale [13], di conseguenza ha caratteristiche completamente diverse rispetto alla metodica precedente.

2.1 Shape from Stereo

Per l'essere umano, la percezione tridimensionale della realtà circostante, viene costruita attraverso un'interpretazione computata dal cervello in base a differenze di posizione nella retina, chiamate disparità, tra punti di riferimento fissi. In effetti, la percezione della profondità è dipendente dall'utilizzo di entrambi gli occhi.

Per Stereo Vision si intende una metodologia di ricostruzione di una struttura tridimensionale partendo da due o più immagini dello stesso oggetto acquisite da diversi punti di riferimento [47, 53]. Analogamente a quanto avviene nell'essere umano, per poter applicare una metodologia del genere si rende necessario l'utilizzo di almeno due sensori. Nel caso in cui sia nota con precisione la geometria del sistema di riferimento, è possibile ricostruire la scena in maniera tridimensionale a partire dalle immagini ottenute dal sistema di acquisizione.

Si prenda come riferimento lo schema in Figura 2.2:

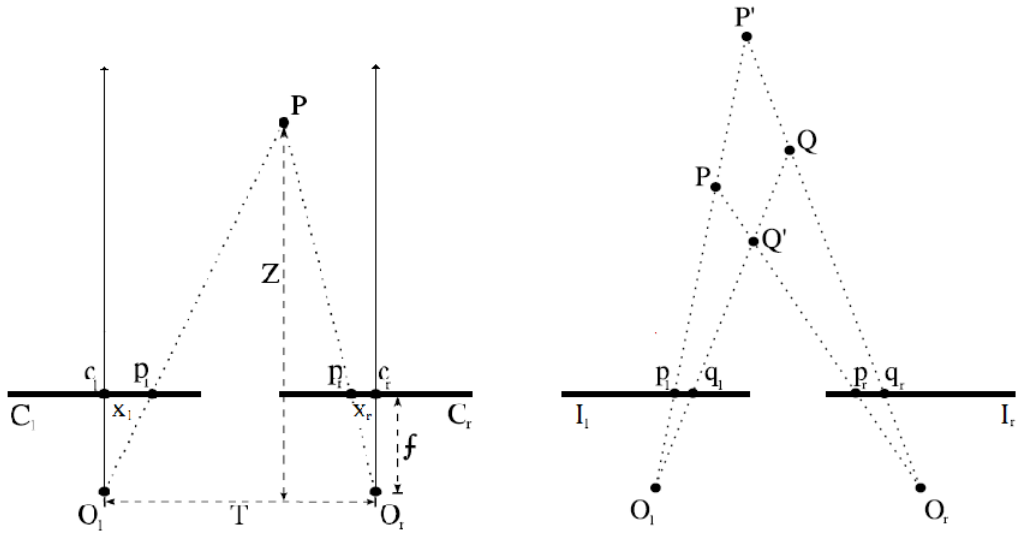

Figura 2.2: Stima della profondità attraverso triangolazione [53].

Il diagramma mostra un ipotetico sistema di acquisizione stereo composto da due telecamere orientate.

Sono detti I_l e I_r i segmenti che rappresentano i piani immagine sinistro e destro e rispettivamente O_l e O_r i centri di proiezione. Il modo in cui la visione stereo determina la posizione nello spazio dei punti P e Q, assunti come riferimento, è detta triangolazione. Intersecando i raggi uscenti dai centri di proiezione, con i punti di riferimento, si ottengono le posizioni relative dei punti di riferimento sul piano immagine.

Scegliendo a coppie i punti (p_l, p_r) e (q_l, q_r) nei piani immagine e intersecando il raggio passante per O_l e p_l con il corrispettivo passante per O_r e p_r si ottiene la posizione del punto P rispetto al sistema di riferimento. Tuttavia, scegliendo a coppie (p_l, q_r) e (q_l, p_r) punti corrispondenti nei piani immagine e intersecando i raggi passanti per O_l e p_l con il corrispettivo passante per O_r e q_r si ottiene la posizione del punto P' rispetto al sistema di riferimento. Nella realtà dei fatti, correlare correttamente le due proiezioni è uno dei problemi principali. In effetti nel caso in cui il matching non sia accurato si determinano degli errori di prossimità che rendono inutilizzabile la stima. Assumendo per semplicità che tale problema sia risolto e cercando di ricostruire la profondità della scena è consequenziale scrivere l'equazione che permette di determinare la triangolazione nella figura sopra riportata.

La distanza T tra i centri di proiezione O_l e O_r è detta baseline del sistema stereo. Se x_l e x_r sono le coordinate dei punti p_l e p_r sui piani immagini rispetto ai riferimenti centrali sugli stessi c_l e c_r, f la distanza focale e Z la

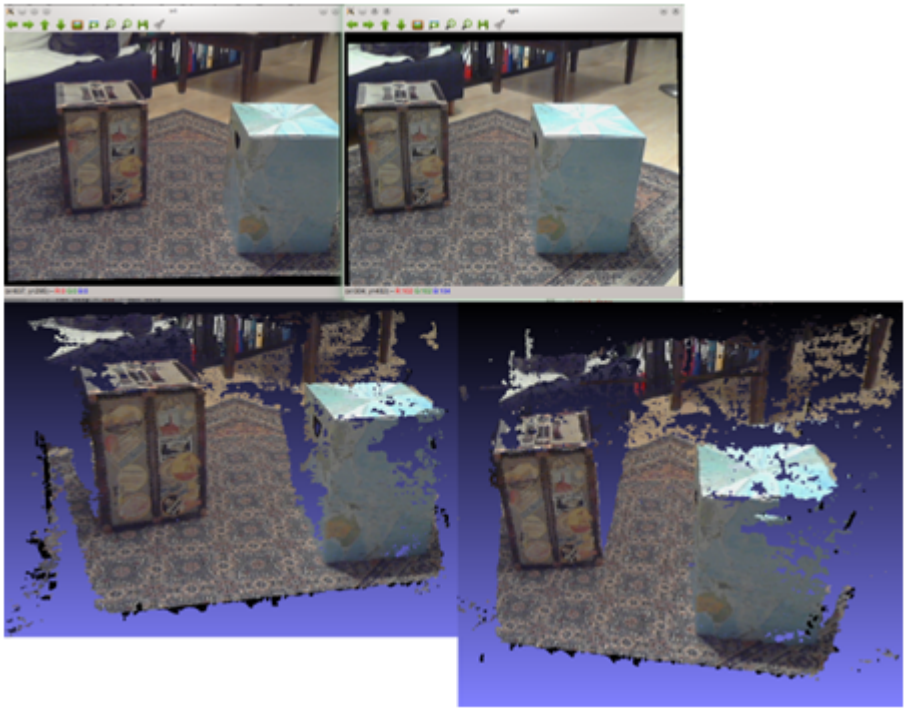

Figura 2.3: Due immagini ottenute con un sistema di acquisizione Stereo e dati tridimensionali risultanti osservati da due punti di riferimento distinti[47].

distanza tra il punto P e la baseline, per le proprietà dei triangoli simili si ottiene:

$$\frac{T - x_l - x_r}{Z - f} = \frac{T}{Z}.$$

Risolvendo rispetto a Z otteniamo:

$$Z = f\frac{T}{d}, \tag{2.1}$$

con $d = x_r - x_l$ disparità, la quale esprime la differenza nelle posizioni della retina tra i corrispondenti punti nei piani immagine.

2.2 Shape from Motion

Questo metodo, deriva come il precedente dallo studio della percezione della profondità umana. In questo caso, la caratteristica prevalente è correlata al movimento. Il metodo Shape from Motion infatti, permette di ricostruire la struttura tridimensionale di una scena estraendo le informazioni dai cambiamenti spaziali e temporali di una sequenza di immagini [53].

Anche questo metodo, similmente allo Shape from Stereo, si basa sul movimento relativo tra il punto di osservazione e la scena. Considerando che questa acquisizione di immagini si basa sull'utilizzo di un solo sensore, la caratteristica di acquisizione ha la necessità di vincolare una ulteriore variante ovvero la temporizzazione. In aggiunta, mentre con la metodica precedente, la sincronizzazione è possibile subito dopo l'acquisizione di entrambe le immagini, in questo caso si rende necessario un elapsed di tolleranza prima di poter effettuare le operazioni di fusione e di determinazione della profondità.

Lo Schema 2.4 raffigura una telecamera orientata e una barra verticale che si sta muovendo a velocità costante verso il punto di osservazione.

È possibile computare il tempo τ che impiega la barra per raggiungere la telecamera solamente a partire dalle informazioni contenute nelle immagini. Identifichiamo con L la dimensione reale della barra, V la sua velocità costante e f la lunghezza focale della telecamera.

Se la posizione della barra è: $D(0) = D_0$ all'istante iniziale, la sua posizione dopo un tempo t sarà:

$$D = D_0 - Vt,$$

da cui

$$\tau = \frac{V}{D}.$$

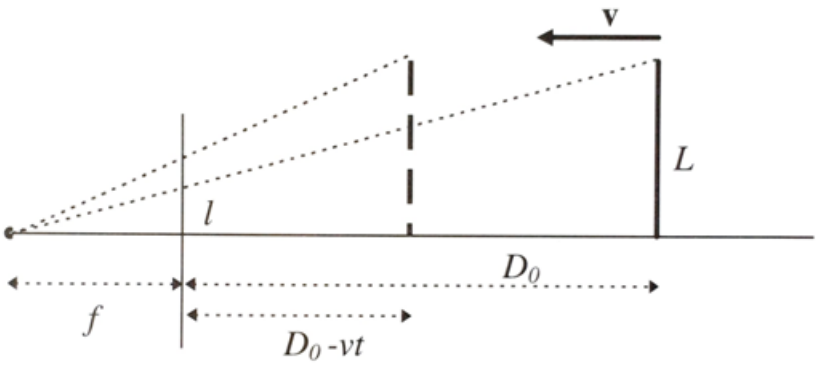

Figura 2.4: Stima della profondità attraverso metodo Shape from Motion [53].

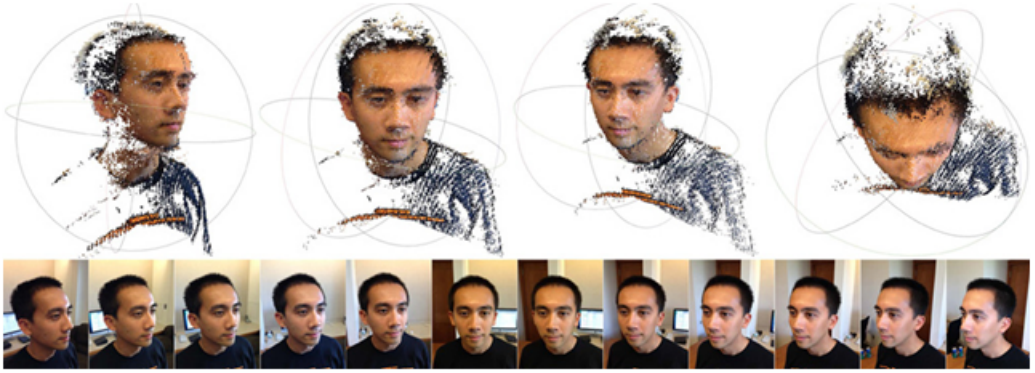

Figura 2.5: Sequenza di immagini ottenute con un sistema di acquisizione Shape from motion e dati tridimensionali risultanti osservati da punti di riferimento distinti [37].

Inoltre sarà:

$$l(t) = f\frac{L}{D},$$

La dimensione apparente della barra al tempo t sul piano immagine.

Se ora computiamo derivando rispetto al tempo $l(t)$, otteniamo:

$$l'(t) = \frac{dl(t)}{dt} = -f\frac{L}{D^2}\frac{dD}{dt} = f\frac{LV}{D^2},$$

da cui

$$\tau = \frac{l'(t)}{l(t)}. \tag{2.2}$$

Sviluppando le equazioni precedenti, è possibile ottenere un campo bidimensionale di vettori (simile al campo di disparità della visione stereo), indicante le velocità relative dei punti di riferimento sul piano immagine, introdotte dal movimento relativo tra il punto di osservazione e la scena osservata.

Considerando alcune approssimazioni, è possibile determinare la riconducibilità a quanto accade per la visione stereo. In particolare, viene stimata la struttura tridimensionale di un oggetto in base a determinati algoritmi che effettuano l'elaborazione su considerazioni di natura analitica o empirica.

2.3 Shape from Shading

Una delle metodiche più interessanti e che ha affascinato figure matematiche di rilievo è lo Shape from Shading [36, 53]. Viene utilizzato in presenza di particolari necessità in campi correlati alla ricerca scientifica e di particolari grandezze che non sarebbero misurabili attraverso strumenti meno sofisticati di questi. Infatti, i metodi di Shape from Shading hanno numerose applicazioni, ad esempio in campo astronomico, in quanto sono utilizzati per ricostruire la superficie dei pianeti attraverso fotografie acquisite dalle sonde spaziali.

Il metodo è particolarmente complesso e si basa su specifici modelli matematici che richiederebbero delle dimostrazioni teoriche che nella presente tesi non verranno affrontate. In particolare, l'algoritmo è fondato su una interazione ipotetica della superficie dell'oggetto con una sorgente di illuminazione tenendo conto delle caratteristiche di diffusione e riflessione proprie della materia da cui è composto. Ipoteticamente, risulta essere la percezione di un osservatore sulla radianza della superficie stessa.

La forma di un qualsiasi oggetto può essere ricostruita attraverso il metodo di Shape from Shading grazie all'esistenza di un'equazione fondamentale che lega l'intensità di un'immagine alla pendenza di una superficie:

$$E(p) = R_{\varrho,I}(n), \tag{2.3}$$

in cui

$$E(p) = L(P)\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4 \alpha, \tag{2.4}$$

$$R_{\varrho,I}(n) = \varrho I^T n. \tag{2.5}$$

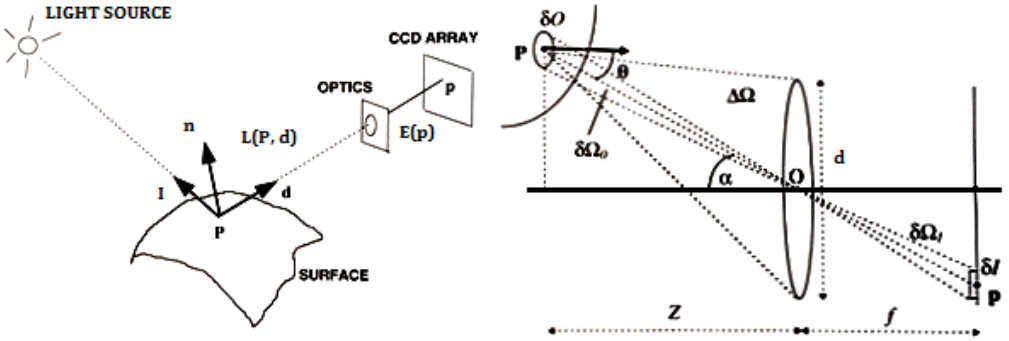

Figura 2.6: Stima della profondità attraverso metodo Shape from shading [53].

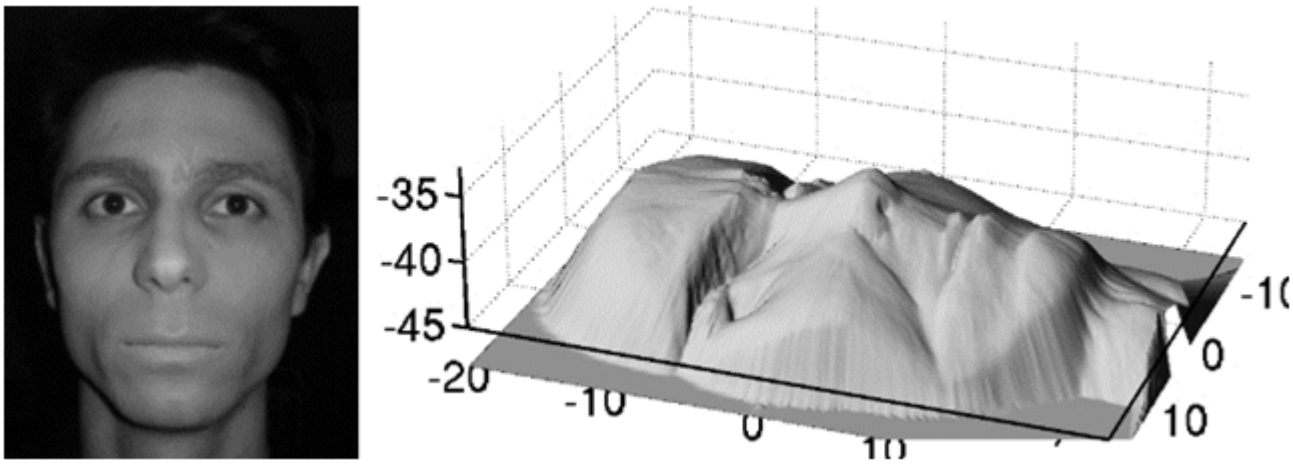

Figura 2.7: Immagine di intensità a scala di grigi e dati tridimensionali ricostruiti attraverso metodo Shape from shading [36].

La duttilità del modello permette di ricavare la struttura tridimensionale di un oggetto, sviluppando il modello sopra riportato, avendo cura di impostare alcune approssimazioni matematiche e numeriche. Si potrebbe pensare di utilizzare tale modello in ogni occasione si renda necessario. Tuttavia, è da considerare che l'accuratezza della stima è direttamente proporzionale alle risorse necessarie per raggiungere risultati concreti. Di conseguenza, viene applicato esclusivamente potendo disporre di sistemi di calcolo sofisticati e per specifiche necessità.

2.4 Shape from Texture

Il seguente è uno dei metodi più utilizzati in quanto è di facile progettualità e tecnica ricostruttiva e consente di ottenere una buona qualità dei risultati. Il concetto sottostante su cui si basa l'architettura per l'acquisizione delle immagini nel corso del tempo è stato sempre più sofisticato permettendo un'ampia diffusione su buona parte dei dispositivi.

Tale metodologia si basa sull'emissione di una luce strutturata, solitamente di natura infrarossa, sulla superficie di un oggetto da ricostruire tridimensionalmente – Figura 2.8. In questo modo, la rifrazione della luce sull'oggetto viene catturata da un sistema di rilevazione il quale permette di determinare la forma spaziale della scena memorizzandola attraverso una triangolazione [32, 53]. In particolare, il significato di Texture relativamente al metodo, indica la ripetizione di un elemento o di un pattern chiamata texel su una superficie. I texels specifici del pattern possono essere di varie tipologie, in particolare, si parla di tipo deterministico, nel caso in cui siano rappresentati attraverso modelli matematici, oppure statistici nel caso in cui il pattern è modificato sfruttando proprietà statistiche definite ad esempio attraverso le proprietà frequenziali nello spazio.

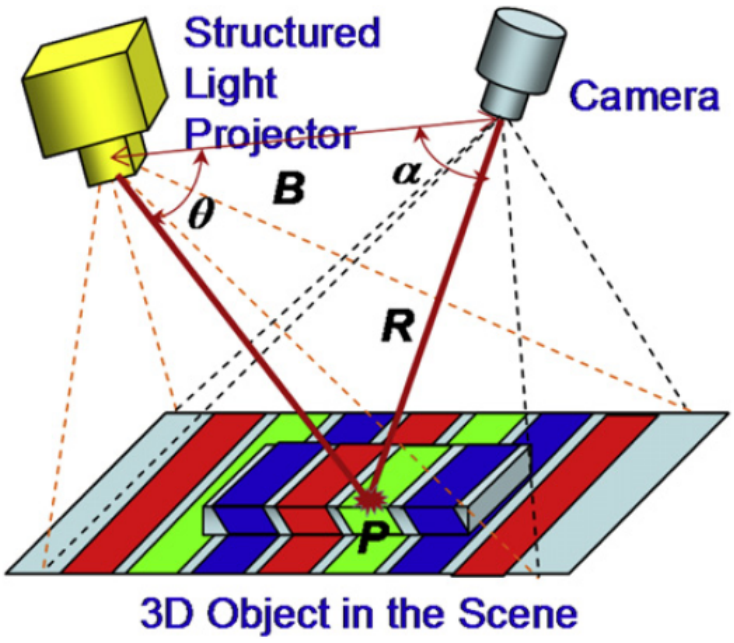

Figura 2.8: Sistema di ricostruzione 3D Shape from texture [32].

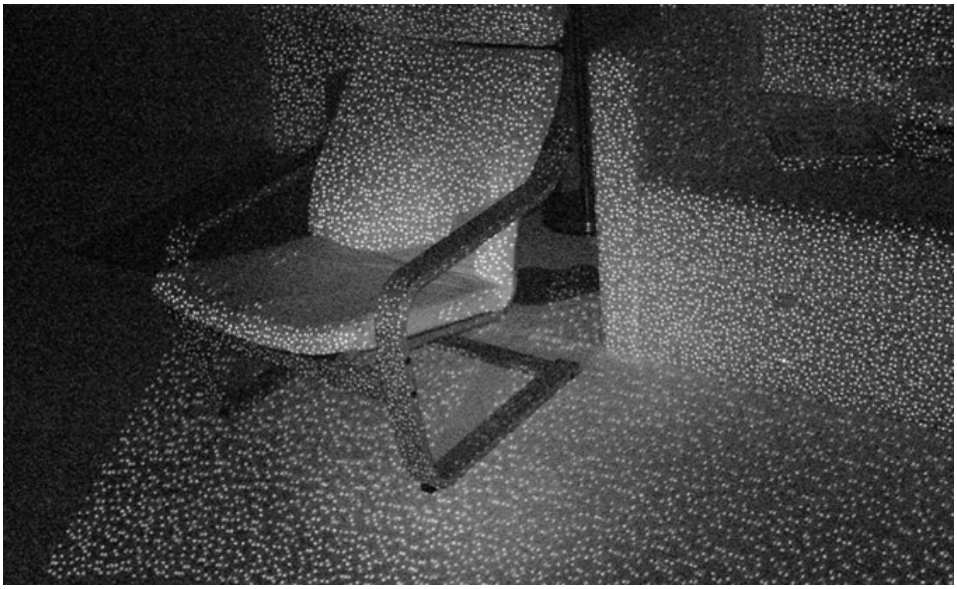

Figura 2.9: Pattern a luce strutturata proiettato su una scena [48].

Si immagini una superficie piana senza variazioni tridimensionali. In questo specifico caso, il pattern acquisito sarà molto simile a quello proiettato in quanto non sussisterebbero distorsioni strumentali all'acquisizione della scena. Diversamente, si immagini ora invece una superficie di natura irregolare. Questo sistema a luce strutturata catturerà ora le forme geometriche con una tecnica di acquisizione delle distorsioni che si vengono a creare sul pattern acquisito.

Nel caso in cui si desideri una trattazione esaustiva dell'argomento, si suggerisce la lettura di OSA, cur. *Structured-light 3D surface imaging: a tutorial*. 2011. URL: `https://www.osapublishing.org/aop/fulltext.cfm?uri=aop-3-2-128&id=211561`

Tecnologia di acquisizione

Microsoft Kinect – Figura 2.10 – nasce nel 2010 come accessorio per XBOX 360 in collaborazione con Microsoft. A differenza degli altri accessori per Nintendo e Sony, Kinect consente all'utente un'esperienza interattiva con la console senza dover interagire attraverso consueti controller o joystick. I costi contenuti e la portabilità multipiattaforma ha reso Microsoft Kinect molto popolare in numerose applicazioni di Computer Vision. Microsoft ha infatti rilasciato gratuitamente a Febbraio 2012 il Kinect Software Development kit (SDK) per Windows. Il sistema di acquisizione è stato rivisto negli anni e la prima release, dotata di tecnologia a luce strutturata, è stata

Figura 2.10: Tecnologia di acquisizione Microsoft Kinect [26].

sostituita nel 2013, da una seconda versione Microsoft Kinect One, che migliora notevolmente la risoluzione del sensore grazie alla nuova architettura Time of Flight. Inoltre l'interfaccia attraverso personal computer è stato reso più friendly rispetto alla versione precedente che rendeva necessari alcuni accessori addizionali.

Microsoft Kinect V1 incorpora architetture hardware di sensing molto avanzate. In particolare è composta da, un sistema a luce strutturata (proiettore infrarossi e telecamera infrarossi), una telecamera RGB e un set di 4 microfoni che, agendo in modo interconnesso, permettono di effettuare operazioni di 3D reconstruction, skeletal tracking, facial and voice recognition. Come anticipato precedentemente, il principio attraverso il quale Microsoft Kinect ricostruisce la profondità è lo shape from texture a luce strutturata.

I valori di profondità computati dai sensori Kinect tuttavia, spesso non sono accurati a causa di una calibrazione errata tra il proiettore IR e la telecamera IR. Questo problema potrebbe essere causato da calore o vibrazioni in fase di acquisizione oppure a causa di un proiettore a infrarossi poco saldo [34]. Per risolvere il problema il team Kinect ha sviluppato alcune tecniche di ricalibrazione che permettono una ridefinizione automatica dei parametri del sistema di acquisizione.

Caratterizzazione del rumore

Una delle problematiche più ricorrenti nell'ambito dei sistemi di acquisizione dell'immagine è la gestione del rumore. Una delle modalità per determinare le caratteristiche del rumore prevede di iniettare alcune tipologie di perturbazioni sul sensore. In base alla reazione e ad un'analisi del comportamento, vengono gestiti dei segnali correttivi come filtro basato su modelli statistici. Diversamente, è possibile stimare direttamente il segnale rumoroso a seguito di osservazioni analitiche ed empiriche [22, 28, 34].

Le tipologie di errore riportate dal Kinect in fase di acquisizione sono in genere:

Zero Depth (ZD) oppure Non-Measured Depth (nmd) Accade quando i punti di un oggetto sono troppo vicini o troppo lontani rispetto al piano immagine, oppure quando la profondità non può essere stimata, ad esempio, a causa di una superficie estremamente irregolare. In questo caso tutti i punti sono assunti a profondità nulla in fase di computazione;

Wrong Depth (WD) In questo caso Kinect riporta un valore non nullo ma errato a causa di una mancanza di accuratezza nella misurazione;

Unstable Depth (UD) In questo caso Kinect riporta un valore non nullo ma variabile nel tempo in assenza di cambiamenti nella scena. Questo tipo di errore, per sua natura temporale, è legato alla precisione della misurazione.

Queste descrizioni sono a titolo informativo ma per una trattazione più esaustiva, esiste ampia documentazione in rete. I principali riferimenti sono disponibili in documenti più strutturati [22]. In particolare è stato possibile riconoscere rumore di tipo spaziale (inerente al singolo frame), temporale (considerando una sequenza di frame) e di interferenza (legato all'uso di più Kinect contemporaneamente).

Al fine di rendere la trattazione più scorrevole, invece di una caratterizzazione estesa si è preferito portare come esempio un caso applicativo che propone una stima basata su un metodo euristico [28]. Il metodo si propone per una misura sistemica della distribuzione del rumore di tipo assiale e laterale in funzione della distanza e dell'angolo di osservazione della superficie dell'oggetto rispetto al piano immagine.

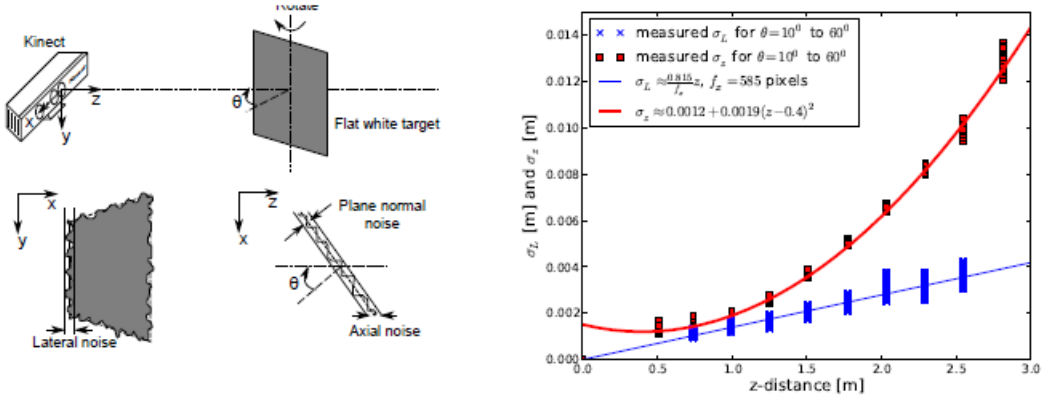

Figura 2.11: Caratterizzazione del rumore assiale e laterale in Microsoft Kinect [28].

È stato fatto ruotare liberamente un piano rispetto a un asse verticale fissato in corrispondenza del sensore, si è poi proceduto ad acquisire una sequenza di immagini tridimensionali – Figura 2.11. Dalla sequenza si è poi preceduto ad una estrazione delle distribuzioni statistiche del rumore.

Facendo un matching dei dati analizzati e interpolando i punti è possibile riportare le seguenti considerazioni:

- Il rumore laterale non varia significativamente con la distanza e assume al più andamento lineare.
- Il rumore assiale cresce con andamento quadratico con l'aumentare della distanza.

In seguito si riportano le deviazioni standard identificate attraverso interpolazione numerica.

$$\sigma_L(\vartheta)[px] = 0.8 + 0.035\frac{\vartheta}{\frac{\pi}{2} - \vartheta}, \qquad \sigma_L(\vartheta)[m] = \sigma_L(\vartheta)[px] z \frac{p_x}{f_x}, \tag{2.6}$$

$$\sigma_z(z, \vartheta)[px] = 0.0012 + 0.0019(z - 0.4)^2 + \frac{0.0001}{\sqrt{z}} \frac{\vartheta^2}{\left(\frac{\pi}{2} - \vartheta\right)^2}. \tag{2.7}$$

2.5 Time of Flight

Nell'ambito delle applicazioni real time si è recentemente affermata una nuova metodologia denominata Time of Flight. Tale metodologia ha soppiantato le metodiche precedenti per ragione di una maggior precisione a parità di applicazione.

I sistemi a tempo di volo si basano su un principio di funzionamento simile a quello dei radar. Viene inviata una radiazione infrarossa modulata opportunamente e viene stimata la dimensione spaziale dei dati. Tale misura avviene tenendo conto della differenza del tempo di trasmissione del segnale e del tempo di ricezione [13, 23].

Pulsed modulation

La misura della distanza della superficie dell'oggetto dal punto di osservazione è ottenuta misurando direttamente il tempo che la pulsazione impiega dall'emettitore a colpire la superficie dell'oggetto e a ritornare indietro dopo riflessione – Figura 2.12.

Se la radiazione è emessa verso la scena che si trova a una distanza ϱ dal punto di osservazione, questa viene riflessa dalla superficie e ritorna al mittente percorrendo la stessa distanza dopo un tempo τ. La distanza può quindi essere stimata attraverso la seguente relazione:

$$\varrho = \frac{c\tau}{2}, \tag{2.8}$$

con c velocità della luce.

Il problema principale nei sistemi a tempo di volo consiste nel determinare esattamente il tempo di arrivo del segnale riflesso. La precisione nella misura

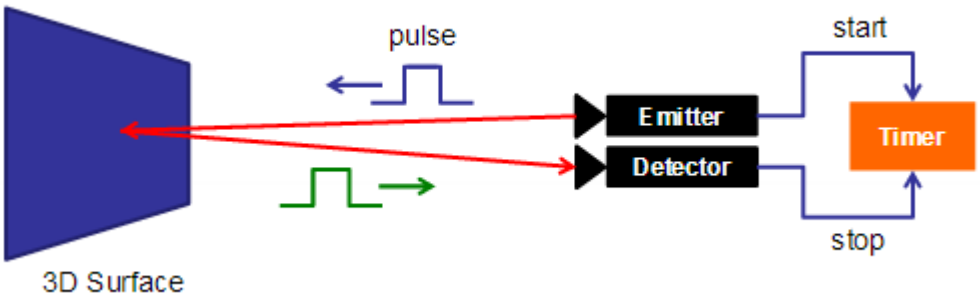

Figura 2.12: Sistema di ricostruzione 3D Time of flight a modulazione pulsata [51].

della distanza è quindi influenzata dalla qualità dello strumento impiegato per la misurazione.

Alcuni sistemi invece, per la misura della distanza, ricorrono a una discriminazione di fase. La distanza è calcolata confrontando la fase del segnale emesso con quella del segnale ricevuto dopo la riflessione sulla superficie dell'oggetto. In particolare, questa modalità viene utilizzata quando il tempo di ricezione del segnale supera una determinata soglia che potrebbe introdurre fenomeni di distorsione.

Continuos wave modulation

La misura della distanza in questo caso, invece, è ottenuta misurando la differenza di fase tra il segnale emesso e quello ricevuto dopo la riflessione – Figura 2.13. Consideriamo sinusoidale il segnale inviato dall'emettitore:

$$g(t) = \cos \omega t.$$

Successivamente alla riflessione, il segnale ritorna verso l'emettitore con una differenza di fase e ampiezza, in particolare:

$$s(t) = b + a \cos(\omega t + \varphi).$$

Facendo la convoluzione dei due segnali otteniamo:

$$c(\tau) = s * g = \int_{-\infty}^{+\infty} s(t)g(t+\tau)d\tau = \frac{a}{2}\cos(\omega t + \varphi) + b.$$

Campionando $c(\tau)$ sequenzialmente in quattro istanti differenti sfasati di $\frac{\pi}{2}$:

$$A_i = c\left(i\frac{pi}{2}\right), \quad i = 0, ..., 3,$$

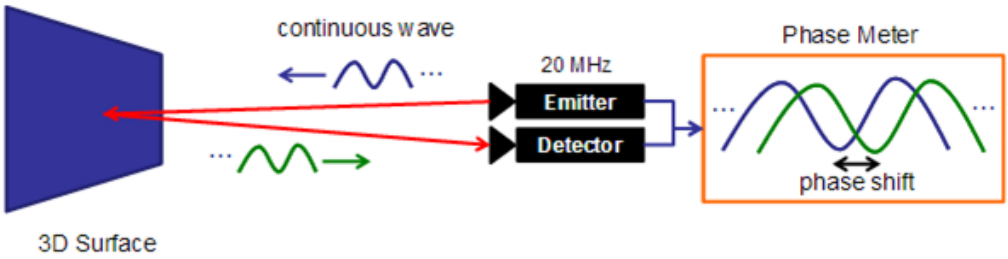

Figura 2.13: Sistema di ricostruzione 3D Time of flight a modulazione sinusoidale continua [51].

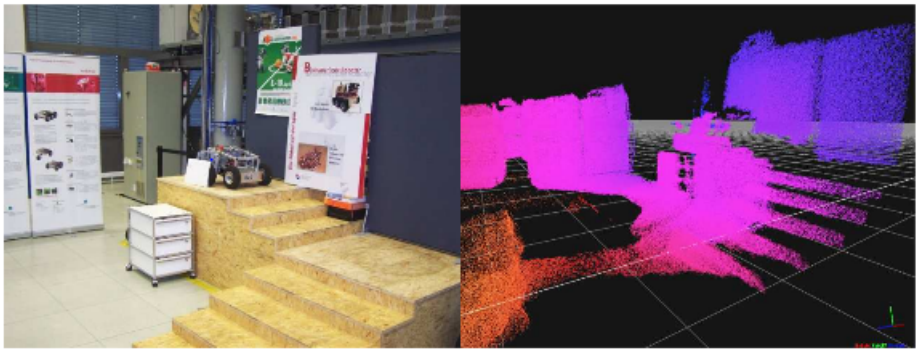

Figura 2.14: Immagine di intensità e dati tridimensionali risultanti ottenuti attraverso sistema Time of flight [23].

otteniamo:

$$\varphi = \arctan\left(\frac{A_3 - A_1}{A_0 - A_2}\right),$$

di conseguenza, la distanza calcolata attraverso il procedimento sopra descritto è pari a

$$\varrho = \frac{c}{2}\frac{\varphi}{2\pi\omega}. \tag{2.9}$$

Tecnologia di acquisizione

Microsoft Kinect One – Figura 2.15 – è la seconda release di XBOX Kinect lanciata in novembre 2013 dal team di sviluppo Microsoft. Il blocco di sensing è composto da una telecamera RGB con risoluzione 1080p, una telecamera di tipo time of flight e un set di microfoni.

Figura 2.15: Tecnologia di acquisizione Microsoft Kinect One [19].

La versione aggiornata offre un notevole aumento della risoluzione, l'autocalibrazione dei parametri della telecamera, l'incremento del campo di visione, la possibilità di gestire fino a 6 giocatori contemporaneamente tenendo anche traccia del battito cardiaco. Il tutto a costi molto abbordabili. Il principio di funzionamento, secondo cui Microsoft Kinect V2 ricostruisce la terza dimensione, è il Time of Flight, tale metodologia è stata descritta ampiamente nei passaggi precedenti.

Caratterizzazione del rumore

Mentre le problematiche relative al rumore con Microsoft Kinect V1 sono state ampiamente studiate e di conseguenza si può ritrovare estesa documentazione in rete, per quanto riguarda Microsoft Kinect V2 le informazioni di riferimento sono molto scarse in ragione della più recente tecnologia.

È possibile comunque determinare alcuni riferimenti introduttivi alla tematica [13, 34], che in più ampie trattazioni tecniche entrano in dettaglio specificatamente sulle telecamere che utilizzano sistema di acquisizione Time of Flight.

Da alcune prime informazioni, acquisite attraverso la rete, gli errori tipici della Kinect V2 comprendono una componente casuale e una sistemica [13]. In particolare, quella casuale, deriva dall'accuratezza del sensore e dal rumore sottostante ad ogni misurazione. La componente sistemica invece, può dipendere da fattori interni ed esterni. Specificatamente, i fattori interni sono legati all'implementazione fisica del sensore, mentre i fattori esterni sono direttamente correlati all'ambiente nel quale vengono svolte le misurazioni.

I principali elementi che possono inficiare le misurazioni, introducendo fenomeni di rumore sono i seguenti:

- riscaldamento del dispositivo;
- temperatura ambientale;
- distanza;
- materiale e colore della superficie dell'oggetto;
- superficie estremamente irregolare della scena;

dove per riscaldamento si intende il passaggio dallo stato stazionario allo stato di regime passando per uno stato transitorio. Uno dei parametri su cui invece si desidera focalizzare l'attenzione è il Motion blur. Quest'ultimo è causato dal movimento relativo tra punto di osservazione e oggetto ed assume una connotazione critica nella fase di ricostruzione tridimensionale della scena

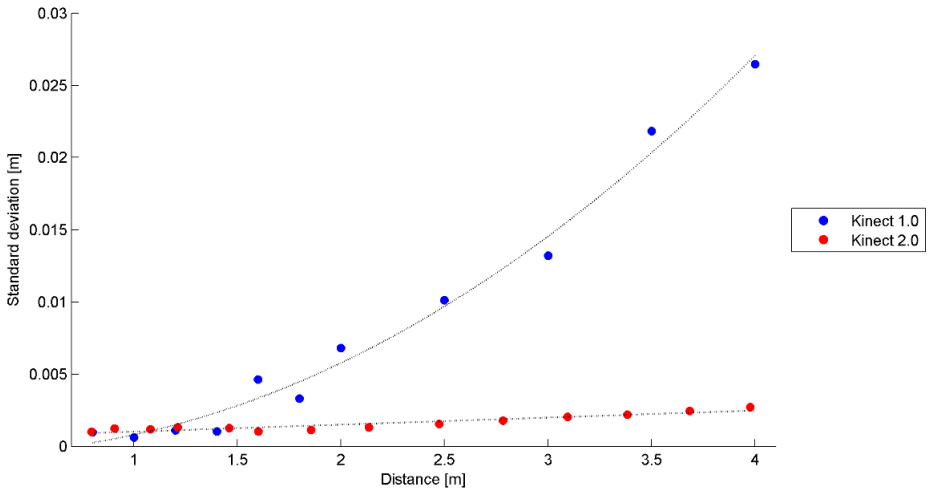

Figura 2.16: Caratterizzazione del rumore in Microsft Kinect e in Microsoft Kinect One [34].

a causa delle blurred region. Per ovviare alla deformazione causata dalla sfocatura in analisi si ricorre all'utilizzo di specifici algoritmi di identificazione delle blurred regions e a specifici deblurring methods.

È possibile quindi confrontare la forma d'onda rumorosa presente nelle due release di Microsoft Kinect [34]. I dati – Figura 2.16 – mostrano una maggiore accuratezza di misura da parte di Microsoft Kinect V2 rispetto alla sua versione precedente. In particolare, interpolando i dati analizzati è possibile riportare le seguenti considerazioni:

- Il rumore in Microsoft Kinect V1 cresce con andamento esponenziale con l'aumentare della distanza.
- Il rumore in Microsoft Kinect V2 assume invece andamento lineare.

In seguito, si riporta la varianza del rumore presente in Microsoft Kinect V2 ottenuta attraverso considerazioni di natura analitica:

$$\sigma_z^2 = \left(\frac{-Z}{f_{mod}}\right)^2 \sigma_{fmod}^2 + \left(\frac{Z}{2\pi\omega + \varphi}\right)^2 \sigma_\varphi^2 \tag{2.10}$$

con f_{mod} frequenza della modulante emessa dall'emettitore, φ sfasamento tra il segnale emesso ed il segnale acquisito dal sensore dopo riflessione, σ_φ deviazione standard della misurazione di fase.

Capitolo 3

Filtraggio e Interpolazione dei dati di profondità

Nei capitoli precedenti è stata effettuata un'analisi dei sistemi di acquisizione e delle modalità inerenti i sistemi di memorizzazione. Tali modalità sono state esaminate al fine di esplorare più in dettaglio le parti successive che sono in costante sviluppo e miglioramento sulla tecnica. In particolare, esamineremo i principali sistemi di filtraggio dell'informazione e di interpolazione. I sistemi di filtraggio vengono esercitati attraverso specifici algoritmi detti algoritmi di Smoothing ed effettuano operazioni di eliminazione di componenti spurie. Successivamente, attraverso l'interpolazione viene ulteriormente migliorata la qualità dell'immagine spaziale precedentemente filtrata.

3.1 Filtraggio

I sistemi di filtraggio rappresentano il primo vero passo verso la gestione della qualità dell'immagine. Progettare un ottimo sistema di acquisizione delle immagini senza prevedere una successiva fase che rappresenta la strategia per l'eliminazione del rumore è un grave errore.

Nel corso degli anni, è stato possibile studiare e sviluppare in maniera approfondita i filtri per le immagini RGB, di conseguenza, questa è una tecnica già molto matura di cui è possibile trovare molto materiale nella rete. Si desume che tale tecnologia è consolidata anche dal fatto che sistemi di filtraggio riportati in pubblicazioni non recenti sono tutt'ora validi ed utilizzati. Diversamente, i filtri per le depth images hanno avuto uno sviluppo ed un impulso evolutivo solo recentemente attraverso la realizzazione dei sensori di profondità. Le pubblicazioni in questo caso sono ancora in fase sperimentale e derivanti da progetti di ricerca [6, 9, 46].

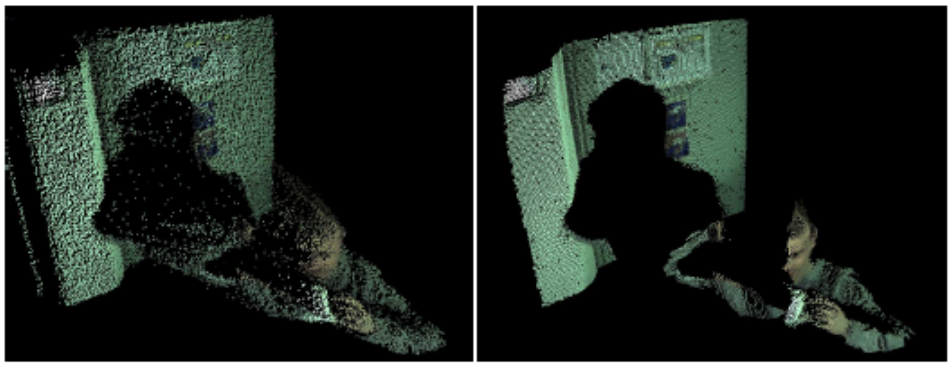

Figura 3.1: Dati tridimensionali raw e dati tridimensionali filtrati [46].

La maggior parte dei metodi di filtraggio per le depth images derivano dai corrispondenti filtri per le immagini RGB ma le differenze tra le due tipologie di filtraggio rimangono sostanziali e presentano problematiche più sofisticate non rilevabili dal precedente sistema. In questo caso, le immagini di profondità sono informazioni raw, ovvero forniscono esclusivamente la dimensione spaziale e di conseguenza, non sono soggette a problemi legati ai cambiamenti di luce. Tuttavia, sussistono tutta una serie di specifiche successive da calibrare in funzione delle fluttuazioni di profondità nel tempo, dello shadowing, dei bordi e della sovrapposizione di oggetti.

Scopo principale del filtraggio è di recuperare l'immagine originale in valore assoluto senza implicazioni dovute allo scenario ambientale [6]. Potremmo sintetizzare tale problematica attraverso la semplice formula:

$$v(i) = u(i) + n(i),$$

dove $v(i)$ rappresenta l'immagine acquisita, $u(i)$ è invece l'immagine originale alla quale si somma una componente variabile dovuta al rumore $n(i)$.

Il metodo più utilizzato è quello di dare una forma al rumore, esprimendone la varianza attraverso termini analitici o empirici. Attraverso un sistema di riconoscimento, l'algoritmo provvede alla successiva rimozione. I vari gruppi di ricerca e sviluppo hanno progettato sistemi basati su diverse strategie ma tutti hanno la necessità di basarsi su alcuni modelli ricavati da stime statistiche. Tali stime sono solitamente determinate attraverso analisi effettuate tramite calibrazioni parametriche.

La strategia di eliminazione del rumore deve necessariamente tener conto di problematiche di natura oscillatoria in quanto talvolta l'informazione è incapsulata assieme al rumore in una forma molto interconnessa. In casi come questi, l'algoritmo solitamente adotta il metodo di non filtrare l'informazione

per evitare di alterare l'immagine originale $u(i)$ evitando così operazioni troppo aggressive.

Tra le distinzioni che possiamo ritrovare in letteratura sulle tipologie di filtri, quella fondamentale riguarda la direzione del filtraggio. Tecnicamente si parla di filtri isotropi e filtri anisotropi (caratterizzati da una direzione specifica). Nel caso di filtri isotropi, il filtraggio avviene uniformemente senza considerare eventuali deroghe. L'immagine ottenuta in questo caso è solitamente rappresentata con alcune debolezze in corrispondenza dei bordi che si presentano sfocati e privi di consistenza. Diversamente, i filtri anisotropi, anche se presentano un numero maggiore di temi da tener conto nella parte di progettazione, hanno il grande vantaggio di determinare una risoluzione dell'immagine senza i problemi sui bordi introdotti dalla metodica precedente.

3.1.1 Filtro Gaussiano

In termini generali, volendo presentare un'introduzione inerente al filtraggio, è importante ricordare che le immagini vengono rappresentate logicamente secondo una specifica notazione compatta sotto forma matriciale. I filtri possono quindi essere visti come delle correzioni che applicano regole di natura logico-matematica direttamente su queste ultime.

Uno dei primi filtri progettato e applicato ai sistemi di acquisizione visuale è il filtro Gaussiano [6]. Tale sistema, essendo poco efficace per le informazioni di natura spaziale è stato successivamente rivisto, migliorato o sostituito attraverso metodiche più efficaci. In particolare, il filtro Gaussiano è un filtro isotropo, di conseguenza è impreciso sui bordi e la strategia con cui è stato realizzato si avvale di un kernel lineare simmetrico che scorre la matrice applicando una convoluzione.

Di seguito, si riporta la formula relativa al sistema di filtraggio in esame:

$$G_h(x, y) = \frac{1}{2\pi h^2} e^{-\frac{x^2+y^2}{2h^2}}, \tag{3.1}$$

con h deviazione standard.

3.1.2 Filtro Bilaterale

La tecnologia inerente al filtraggio bilaterale è stata sviluppata e sfruttata inizialmente per le immagini bidimensionali e successivamente sono state studiate delle varianti per gestire con modalità simili le immagini tridimensionali In questo particolare caso, la progettazione dell'algoritmo si basa sull'idea di migliorare il pixel centrale all'interno di un nucleo locale basandosi su una media pesata dei pixel adiacenti [9]. Tali pixel vengono continuamente misurati

in base alla loro distanza spaziale euclidea. A questo sistema di filtraggio si deve la nascita della maggior parte delle tecniche progettate successivamente. Di seguito, si riporta la formula relativa al sistema di filtraggio in esame:

$$B(x,y) = \frac{1}{W_{x,y}} \sum_{(x',y')\in\omega} D(x',y') G_{\sigma_s}(\|P_{x,y} - P_{x',y'}\|) G_{\sigma_d}(|D(x,y) - D(x',y')|), \tag{3.2}$$

con D immagine di profondità, $P_{x,y}$ pixel (x,y), $W_{x,y}$ costante di normalizzazione, ω intorno del pixel (x,y) e G_{σ_s}, G_{σ_d} funzioni gaussiane che determinano i coefficienti di pesatura spaziali.

Una delle varianti sul filtro bilaterale, è rappresentata dal Joint Bilateral Filter. Nella pratica, la tecnica di miglioramento anziché prendere in considerazione la profondità spaziale si avvale delle informazioni inerenti al colore. Tale metodica è utilizzata univocamente per le immagini RGB-D in quanto non potrebbe applicarsi alle immagini senza la presenza del colore. Di seguito, si riporta la formula relativa al sistema di filtraggio in esame:

$$BC(x,y) = \frac{1}{W_{x,y}} \sum_{(x',y')\in\omega} D(x',y') G_{\sigma_s}(\|P_{x,y} - P_{x',y'}\|) G_{\sigma_c}(|I(x,y) - I(x',y')|), \tag{3.3}$$

in cui è presente I immagine di intensità utilizzata in sostituzione dell'immagine di profondità D per computare i coefficienti di pesatura.

3.1.3 Filtro non locale

Come enunciato precedentemente, esistono filtri locali e non locali, il filtro che esaminiamo in questa sezione, è non locale e di conseguenza prende in esame un numero molto più vasto di pixel. Nella pratica, risulta simile al filtro bilaterale ma essendo applicato a una finestra molto più ampia mira a determinare una immagine con dei dettagli più sofisticati rispetto ai filtri locali in considerazione del fatto che è possibile distinguere i dettagli più fini dal rumore [9, 46]. Di seguito, si riporta la formula relativa al sistema di filtraggio in esame:

$$NLM(x,y) = \sum_{(x',y')\in W_{x,y}} \omega((x,y),(x',y')) D(x',y'), \tag{3.4}$$

con $W_{x,y}$ finestra di ricerca sufficientemente ampia attorno al pixel (x,y) e w coefficiente di pesatura dipendente dalla distanza spaziale euclidea d tra due patches centrare attorno a (x,y) e (x',y'),

$$\omega((x,y),(x',y')) = \frac{1}{Z_{x,y}} e^{-\frac{max(d^2 - 2\sigma^2, 0)}{h^2}}, \tag{3.5}$$

dove $Z_{x,y}$ è la costante di normalizzazione equivalente alla somma di tutti i coefficienti di pesatura, h è un parametro di filtraggio e σ è la deviazione standard del rumore.

Analogamente al filtro Joint Bilateral è stato introdotto tra le tecniche un filtro non locale che prende in esame le informazioni memorizzate inerenti il colore. Come nel caso del filtro precedente, le informazioni considerate hanno una gamma molto più ampia ma sono applicabili unicamente alle immagini RGB. Di seguito, si riporta la formula relativa al sistema di filtraggio in esame:

$$NLMC(x,y) = \sum_{(x',y') \in W_{x,y}} \omega((x,y),(x',y'))\omega_c((x,y),(x',y'))D(x',y'), \quad (3.6)$$

in cui ω_c è il coefficiente di pesatura computato sull'immagine di intensità.

3.1.4 Filtro temporale

Un altro metodo nato per l'elaborazione 2D e successivamente rielaborato per l'elaborazione tridimensionale è l'Adaptive threshold filter [9]. Tale modalità di filtraggio, adotta la strategia di escludere le modifiche di profondità spaziale dell'immagine per ogni pixel in base a considerazioni di intervallo di soglia. In particolare, l'algoritmo accetta le nuove misurazioni esclusivamente nel caso in cui la differenza dei valori della nuova immagine, rispetto all'immagine filtrata precedentemente eccede un determinato valore. Tale valore è differente per ogni pixel e, di conseguenza, la calibrazione avviene in accordo alla deviazione standard dei valori di profondità ad un determinata distanza. Di seguito, si riporta la formula relativa al sistema di filtraggio in esame:

$$z_f(t) = \begin{cases} z_m(t), & |z_m(t) - z_f(t-1)| > D_{max}(z_m(t)), \\ z_f(t-1), & \text{altrimenti}, \end{cases} \quad (3.7)$$

dove z_m è il valore di profondità riportato da Kinect e z_f quello filtrato.

Uno dei più recenti sistemi di filtraggio è il Temporal denoising filter [9]. Mentre buona parte delle metodiche precedenti nascono da esperienze sulla elaborazione 2D, questa metodologia nasce da un progetto appositamente strutturato per il filtraggio su sistemi 3D. La potenza del metodo si basa sull'idea che talvolta i sistemi di filtraggio durante il movimento introducono in realtà nuovi errori. Di conseguenza, invece, in caso di movimento lento questo sistema interrompe momentaneamente gli algoritmi di filtraggio. In questo modo, l'algoritmo è in realtà un sistema dinamico che prende delle decisioni strumentali alla ricostruzione della dinamica della scena.

L'acquisizione di nuovi valori, avviene a intervalli irregolari determinati da eventi, tuttavia esiste una stretta relazione tra il confronto di molteplici variabili le quali sono molto correlate tra di loro in base alle seguenti relazioni:

$$s_z(t) = z_m(t) - z_m(t-1), \tag{3.8}$$

$$\Delta_z(t) = z_m(t) - z_m(t-T), \tag{3.9}$$

$$d_z(t) = z_m(t) - z_f(t-1), \qquad \hat{d}_z(t) = \frac{|d_z(t)|}{D_{max}}, \tag{3.10}$$

$$v(t) = \frac{|\Delta_z(t)|}{T}, \qquad \hat{v}(t) = \frac{v(t) - v_{min}}{v_{max} - v_{min}}, \tag{3.11}$$

$$\hat{m}_z(t) = max\big(\hat{v}(t), \hat{d}_z(t)\big), \tag{3.12}$$

dove $s_z(t)$ è la differenza tra la misurazione corrente e la misurazione precedente, $\Delta_z(t)$ è la differenza tra la misurazione corrente e la misurazione precedente sfasata di un tempo T, $d_z(t)$ è un offset che esprime la differenza tra la misurazione corrente e la misurazione precedente filtrata, D_{max} è la soglia computata che descrive la relazione tra la deviazione standard dei dati di profondità e la distanza del soggetto dal sensore, $v(t)$ è la velocità media su un periodo di tempo T, $\hat{d}_z(t)$ esprime l'offset in forma normalizzata, $\hat{v}(t)$ è la normalizzazione della velocità computata tra le soglie v_{min} e v_{max}, infine $\hat{m}_z(t)$ è un parametro ibrido che viene settato come il valore massimo scelto tra $\hat{v}(t)$ e $\hat{d}_z(t)$.

In particolare, nel caso in cui la differenza del valore misurato di una scena rispetto alla precedente ecceda una soglia convenzionale, ne viene verificata la classificazione in base alla quale si discrimina la presenza o meno del soggetto in movimento. In questo particolare caso, il sistema di filtraggio effettua delle congetture in base alle quali viene applicata soltanto una parte dei filtri attivi, specificatamente viene inibito il filtraggio temporale. Nel caso in cui il soggetto invece sia stazionario, i sistemi di filtraggio sono tutti attivi e di conseguenza sussiste un guadagno $g_z(t)$ usato per filtrare i valori di profondità acquisiti. Il tutto è regolato in base alle seguenti formule:

$$g_z(t) = \begin{cases} g_{min} + \frac{1}{2}(1 - g_{min})\big(\hat{m}_z(t)\pi - \frac{\pi}{2}\big), & \text{se } \hat{m}_z(t) \leq 1, \\ \frac{d_z(t)}{s_z(t)}, & \text{altrimenti se } s_z(t) \neq 0, \\ 0, & \text{altrimenti,} \end{cases} \tag{3.13}$$

$$z_f(t) = \begin{cases} z_m(t), & \text{se } |s_z(t)| > D_{max}(z_m(t)), \\ z_f(t-1) + g_z(t)s_z(t), & \text{altrimenti.} \end{cases} \tag{3.14}$$

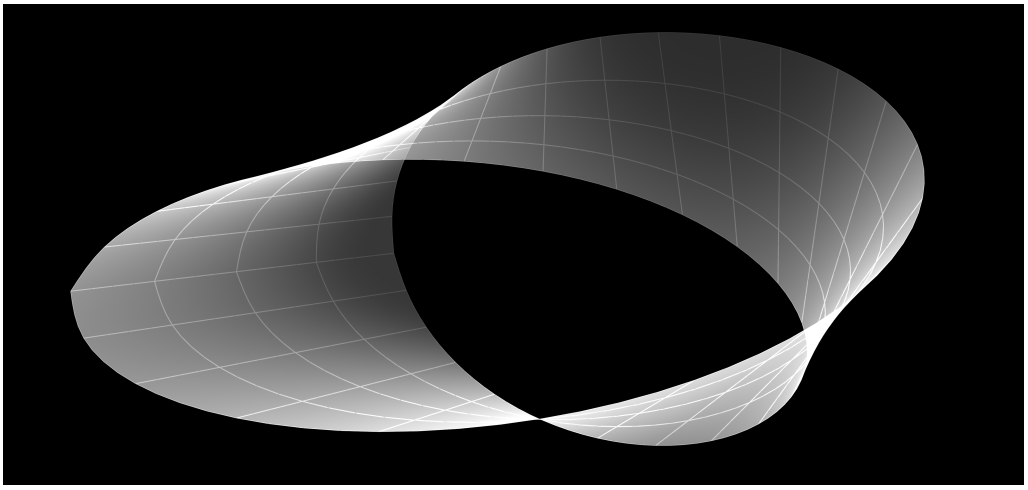

Figura 3.2: Nastro di Mobius [1].

3.2 Interpolazione

I sistemi di filtraggio non sono sufficienti da soli per dare una risposta concreta ai moderni sistemi di acquisizione dei dati in forma solida. Questo perché, nel caso di forme particolarmente complesse che presentano superfici irregolari, l'informazione risultante non è sufficiente a consolidare una forma definitiva del soggetto. La tecnica che fornisce una soluzione incrementale denominata interpolazione mira a trovare delle regole geometrico-matematiche inerenti il solido tali da poterle applicare per approssimare la superficie [10]. Nella pratica, la superficie viene analizzata geometricamente utilizzando la triangolazione come tecnica per supportare altre rappresentazioni stabilendo delle relazioni tra le misurazioni vicine. Questo rappresenta il primo passo per ricostruire le forme più complesse caratteristiche dei soggetti in esame.

Volendo trovare dei tratti distintivi che possano caratterizzare la tecnica, distinguiamo alcuni casi di seguito, in particolare:

- Nel caso in cui siano disponibili solo pochi punti l'algoritmo deve funzionare ugualmente per poi convergere alla superficie reale all'aumentare della densità dei dati;
- Nel caso in cui siano disponibili molti punti, l'algoritmo deve semplificare le misurazioni ridondanti;
- Nel caso in cui si rendano disponibili nuove informazioni, il sistema viene velocemente riaggiornato (dynamic flexibility);

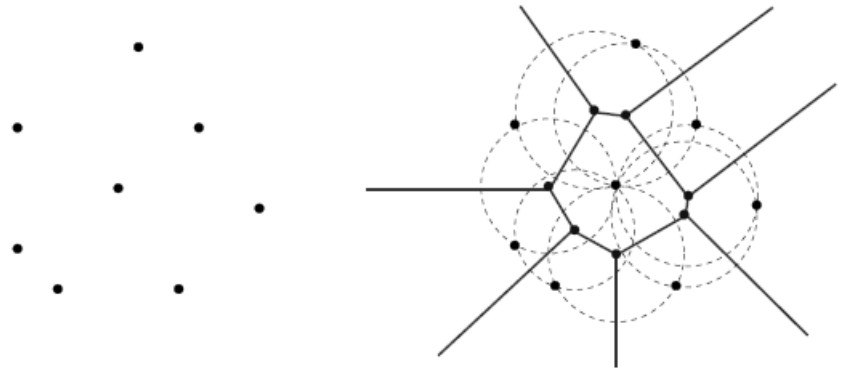

Figura 3.3: Diagramma di Voronoi [52].

- L'algoritmo ha delle regole di ottimizzazione volte a semplificare la complessità di computazione per ragioni di efficienza.

Nelle sezioni seguenti, verranno esaminati metodi, strumenti e tecniche che compongono il procedimento nella sua architettura. Verrà posta una particolare attenzione a fornire le spiegazioni di natura geometrica, le quali si sviluppano attraverso considerazioni di topologia matematica.

3.2.1 Diagramma di Voronoi

Supponiamo che all'interno del nostro dispositivo, successivamente alla misurazione dei punti 3D della superficie del soggetto vengano applicate le regole di filtraggio. Fino a questo punto, tuttavia, non abbiamo informazioni di interconnessione tra i vari punti, di conseguenza consideriamo di collegare l'intorno di punti adiacenti sulla superficie con segmenti logicamente connessi. In questo caso, operando senza imporre particolari logiche potremmo ottenere una rappresentazione somigliante (ma non necessariamente) a una struttura poliedrica. In questa accezione, è bene considerare che esistono innumerevoli tipologie di poliedri, tuttavia, nell'obiettivo della semplificazione è bene considerare esclusivamente quelli che minimizzano il numero totale di bordi sulla superficie del soggetto. Con questa semplice regola, l'algoritmo si assicura efficienza e similitudini dettate da congruenza nelle forme.

In particolare, insita nel procedimento, esiste una struttura geometrica che opera una semplice approssimazione poliedrica sulle superfici tridimensionali detta Triangolazione di Delaunay. Tale struttura è geometricamente derivata ad un'altra per tramite del Diagramma di Voronoi [10]. Quest'ultimo permette di estrarre una rappresentazione ottimizzata della forma convergendo verso una superficie reale all'aumentare del numero di campionature.

Si prenda come riferimento la Figura 3.3. Sia $d(P, Q)$ la distanza euclidea tra due punti P, Q e sia S un insieme finito di n punti $M_1, M_2, ..., M_n$. Il Diagramma di Voronoi di S è un insieme V composto da n poliedri $V_1, V_2, ..., V_n$ che chiudono il piano. Il singolo poliedro V_i in particolare rappresenta il luogo dei punti P_i più vicini al punto M_i rispetto ad ogni altro punto appartenente all'insieme iniziale di punti S.

$$V_i = \big\{P | d(P, M_i) \leq d(P, M_j), j = 1, ..., n\big\}. \tag{3.15}$$

Il Diagramma di Voronoi possiede molteplici proprietà e sarebbe possibile fornire una trattazione molto strutturata sull'argomento, tuttavia, nella presente si preferisce trascurare le specifiche di natura prettamente matematica in favore di una esposizione sintetica al fine di non deviare dall'obiettivo iniziale di una specializzazione legata alla ricostruzione tridimensionale. Si invita a prendere visione della proprietà fondamentale del diagramma la quale afferma che: il vertice del diagramma in oggetto appartenente a tre regioni V_i, V_j, V_k è equidistante dai tre punti P_i, P_j, P_k e coincide con il centro del cerchio passante per quei punti.

3.2.2 Triangolazione di Delaunay

Com'è intuibile per tramite di passaggi precedenti, esiste una naturale derivazione del diagramma di Voronoi. Tale collegamento va ad estendere un'architettura di procedimenti tali da risolvere qualsiasi problematica di profondità, rotazione e manipolazione del soggetto in un formato indipendente dalla misurazione.

In particolare, è possibile rappresentare la Triangolazione di Delaunay come sviluppo del Diagramma di Voronoi collegando i punti contigui [10], ovvero collegando i punti P_i e P_j appartenenti alle regioni V_i e V_j separate da un lato del diagramma – Figura 3.4. Considerando più nel dettaglio la struttura geometrica è intuitivo osservare che il cerchio circoscritto al triangolo di Delaunay avente come vertici i punti P_i, P_j, P_k non contiene nessun altro punto $P_l \neq \big\{P_i, P_j, P_k\big\}$ dell'insieme di punti dato. Questa particolare proprietà è detta proprietà del cerchio circoscritto e descrive la caratteristica fondamentale della Triangolazione di Delaunay.

Comprensibilmente, la struttura geometrica in oggetto potrebbe non essere definita in particolare casi. Nello specifico, la Triangolazione di Delaunay ammette la circostanza degenere in cui se esistono quattro o più punti concentrici, esisteranno di conseguenza due o più vertici di Voronoi coincidenti ed il modello geometrico non sarà univocamente determinato.

In base a quanto enunciato precedentemente, si rende ora necessario applicare procedimenti simili passando dalle due alle tre dimensioni. Di conse-

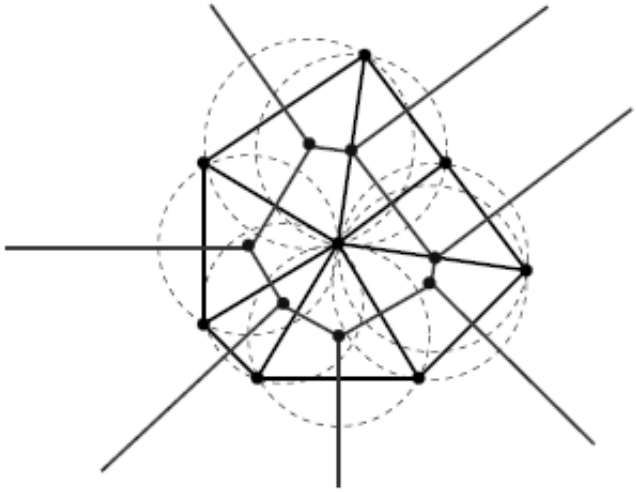

Figura 3.4: Triangolazione di Delaunay [52].

guenza, mentre nei casi precedenti si parlava genericamente di poliedri intesi come figure piane delimitate tra loro da lati, in questo caso si rende necessario interagire con poliedri delimitati tra loro da facce poligonali equidistanti dai punti contigui. Le facce poligonali si intersecano lungo gli spigoli che sono equidistanti da tre punti contigui a due a due. Gli spigoli si incontrano in vertici che sono equidistanti da quattro punti e quindi conincidono con il centro della sfera passante per quei quattro punti. La Triangolazione di Delaunay, come precedentemente affermato, si ottiene connettendo i punti contigui.

Scendendo via via nel procedimento, ci si potrebbe sorprendere della consequenzialità e dell'elegante intuitività nascosta nel modello. Infatti, esiste un interessante legame tra la Triangolazione di Delaunay, il numero dei punti acquisisti sulla superficie di un soggetto e la forma del soggetto stesso. In particolare, è verificabile che, al crescere della densità dei punti misurati tramite uno specifico sistema di acquisizione, i centri delle sfere di Delaunay convergono in direzione della struttura reale del soggetto. Tuttavia, come vedremo più avanti, le soluzioni che semplificano i procedimenti, sono sempre basate su concetti matematici molto complessi.

3.2.3 Algoritmo di Bowyer-Watson

La triangolazione di Delaunay si distingue in due diverse fasi, la prima è strumentale alle operatività di triangolazione in senso stretto; la seconda si rende necessaria in caso di modifica locale dei punti e permette un riequilibrio

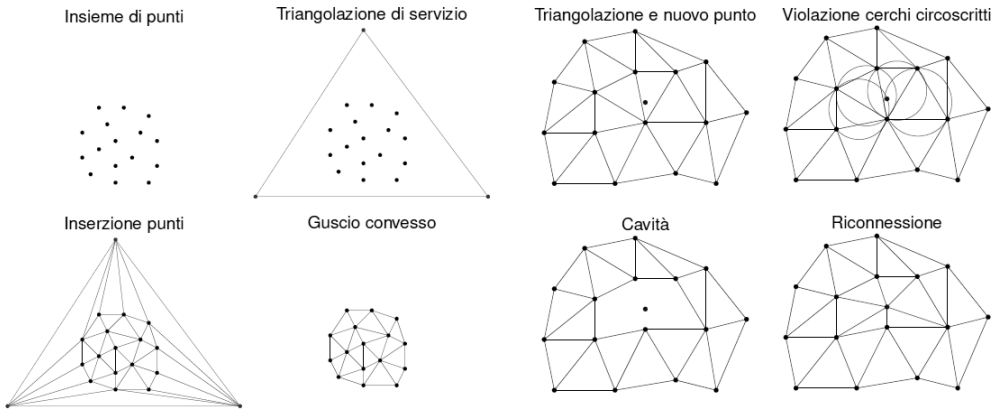

Figura 3.5: Algoritmo di Bowyer-Watson [52].

del sistema rispetto alla triangolazione precedente. Tali fasi vengono esercitate attraverso degli algoritmi specifici diversi tra loro e la loro strategia di utilizzo è alternativa una rispetto all'altra (Bowyer-Watson oppure Green-Sibson).

Per raggiungere la triangolazione completa, esistono alcune tecniche che permettono il raggiungimento definitivo tramite attività incrementali. In particolare, accade che dato un punto facente parte dell'insieme dei punti acquisiti e data una triangolazione di servizio contenente tutti i punti dell'insieme dato, vengono utilizzate delle tecniche di riconducibilità. Viene valutato l'utilizzo di ogni singola tecnica attraverso un apposito algoritmo che sfrutta le proprietà del cerchio circoscritto per determinare la triangolazione incrementale. La somma delle triangolazioni incrementali, determina la triangolazione completa del solido, infine viene rimossa la triangolazione di servizio.

L'inserimento di un nuovo punto P in una triangolazione di Delaunay T_n consiste in due passi – Figura 3.5:

Costruzione della cavità La cavità C è l'insieme dei triangoli di T_n il cui cerchio circoscritto contiene il punto P. I triangoli della cavità devono essere eliminati da T_n in quanto non sono aderenti alla proprietà del cerchio circoscritto;

Riconnessione La triangolazione di Delaunay T_{n+1} si ottiene connettendo il punto P con i punti P_k appartenenti al contorno della cavità.

Per emanazione, il metodo viene esteso al caso spaziale considerando di generare una triangolazione di servizio cubica e conseguentemente utilizzando per la costruzione della cavità la proprietà della sfera circoscritta. Per quanto riguarda il costo computazionale di k punti, è possibile utilizzare un algoritmo

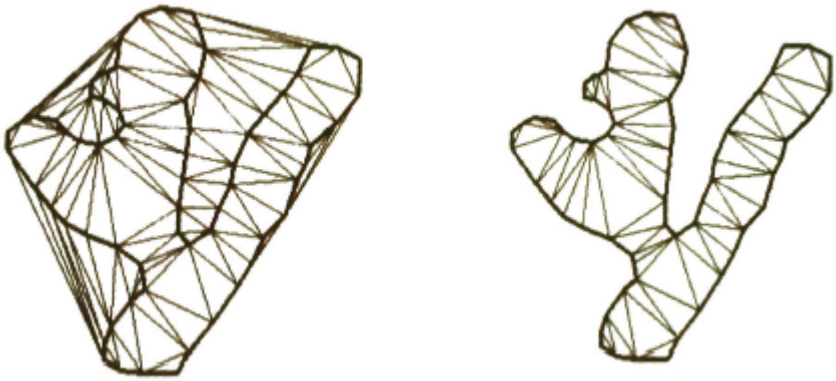

Figura 3.6: Costruzione dell'interpolazione poliedrica [10].

$O(k)$ nel caso piano e $O(k^2)$ nel caso tridimensionale con una worst case direttamente correlata alle degenerazioni di $O(k^2)$ nel caso piano e $O(k^3)$ nel caso spaziale.

3.2.4 Costruzione dell'interpolazione poliedrica

Scopo di questa sezione è inerente all'estrazione di un poliedro da una Triangolazione di Delaunay preesistente [10]. Si prenda come riferimento la Figura 3.6. Ipotizziamo O un punto esterno ad un soggetto X ed M un punto del contorno complessivo dX. Il punto M è visibile dal punto di osservazione O nel caso in cui il segmento OM non intersechi il contorno dX in ogni suo punto. Di conseguenza, OM è denominato raggio ottico. Ipotizziamo ora P un insieme di punti in dX misurati da un numero di posizioni O_i. Per ogni punto misurato è associata la posizione relativa alla misurazione. Di conseguenza, per ogni punto M visibile da O_i, il segmento MO_i è un raggio ottico. Per determinare lo spazio vuoto che caratterizza l'oggetto, vengono eliminati tutti i triangoli nella Triangolazione di Delaunay di P che intersecano il raggio ottico MO_i.

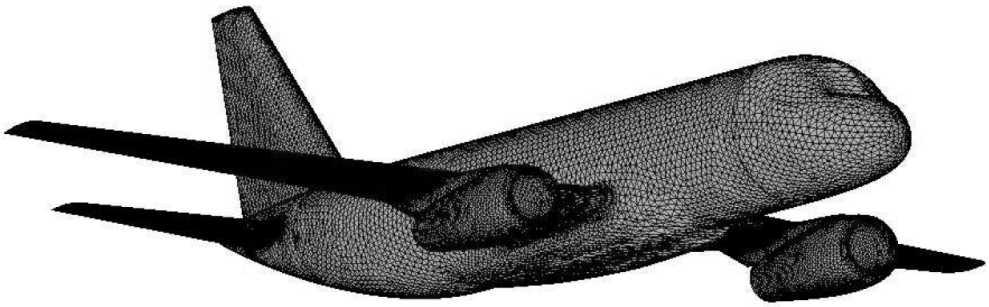

Figura 3.7: Ricostruzione finale del soggetto attraverso interpolazione poliedrica [52].

Una conseguenza di questa metodologia sviluppata attraverso un algoritmo, permette l'opportunità di rilevare attivamente la forma di un oggetto semplicemente muovendo un sensore attorno ad esso. Infatti, è dimostrabile che, il numero dei punti acquisiti è direttamente proporzionale alla approssimazione più convergente del soggetto, sempre considerando le iterazioni successive.

Capitolo 4

Associazione dinamica di immagini 3D

Riepilogando, nei capitoli precedenti è stato delineato un percorso attraverso il quale è stato possibile approfondire alcune tematiche come ad esempio la memorizzazione delle immagini 3D e l'acquisizione delle stesse attraverso varie tecniche. Abbiamo poi approfondito alcune specifiche inerenti il rumore e il filtraggio e successivamente abbiamo analizzato l'ottimizzazione derivante dall'utilizzo di particolari tecniche. Infine, abbiamo avuto modo di interagire con la struttura teorica su cui si basano i meccanismi per l'interpolazione dei dati tridimensionali soffermandoci sulle eleganti tematiche di Triangolazione di Delaunay.

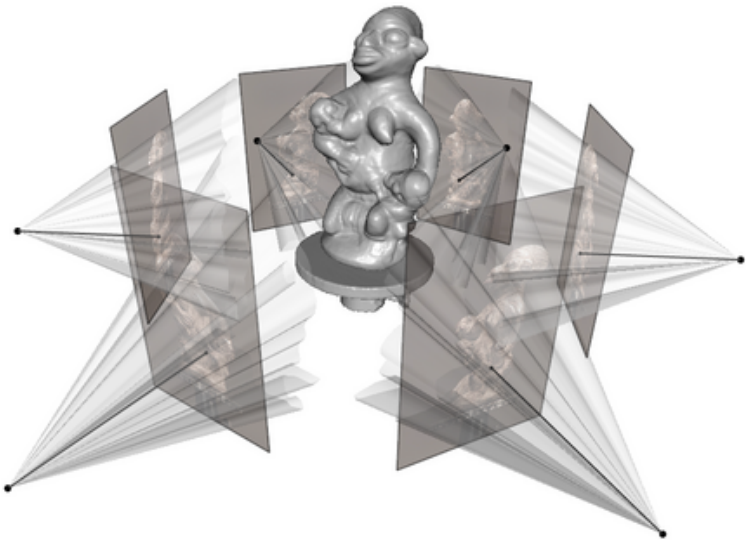

Figura 4.1: Associazione dinamica di immagini 3D [41].

Da qui in avanti, cercheremo di dare una forma strutturata, attraverso una sorta di cabina di regia, che metta in relazione le tecniche sopra enunciate come parti di un unico organismo. Esistono numerose tecniche e sarebbe estremamente dispersivo esaminarle tutte. Per scelta, abbiamo ritenuto di esaminare in maniera approfondita una tecnica che racchiuda buona parte delle componenti utilizzate da tutte le metodologie. In questo modo sarà possibile disporre di una idea intuitiva che consenta di approcciare ogni singola tecnica avendo presente l'iter di un processo complesso. Nello specifico, andremo ad analizzare alcuni prodotti quali Kinect Fusion ReconstructMe e Scanect tenendo conto che mentre per alcuni esiste un ampia letteratura che ne descrive il funzionamento, per altri essendo procedimenti basati su architetture proprietarie commerciali, non sarà possibile fornire spiegazioni approfondite.

4.1 Microsoft Kinect Fusion

Il progetto Kinect Fusion nasce all'interno del centro di ricerca Microsoft e successivamente si è esteso su varie ramificazioni attraverso community internazionali le quali hanno permesso uno sviluppo sofisticato dello strumento [18, 25, 27]. Volendo offrire una breve panoramica delle potenzialità possiamo dire che Microsoft Kinect Fusion permette delle acquisizioni di scena tridimensionali molto precise in tempo reale ma soprattutto con una capacità di interazione che consente il movimento dell'acquisitore sulla scena svincolato da problematiche di calibrazione continua. La potenza di questo strumento è in parte dovuta alla duttilità di progettazione in quanto appositamente strutturato per gestire applicazioni ibride inerenti tematiche di Augmented Reality – Figura 4.2.

Per avere un'immagine intuitiva del funzionamento di Kinect Fusion è utile pensare a Microsoft Kinect che si muove all'interno di un sistema di riferimento globale in tre dimensioni, il quale viene costantemente aggiornato e controaggiornato dinamicamente in maniera congiunta al movimento attraverso l'associazione delle Point Cloud generate dal sistema di acquisizione.

Come spiegato precedentemente, uno dei punti di forza di Kinect Fusion è l'interattività. Tale caratteristica la rende particolarmente efficace per applicazioni di tipo CAD, Computer Graphics, 3D Printing, Augmented Reality oppure applicazioni ibride e/o correlate alle precedenti. In particolare, è possibile acquisire i dati di scena e inserire delle varianti tridimensionali in grado di interagire in maniera simbiotica. È possibile gestire problematiche di avartizzazione le quali possono poi interfacciarsi con soggetti digitali che possono ricordare problematiche simili a quelle utilizzate nell'ambito dei

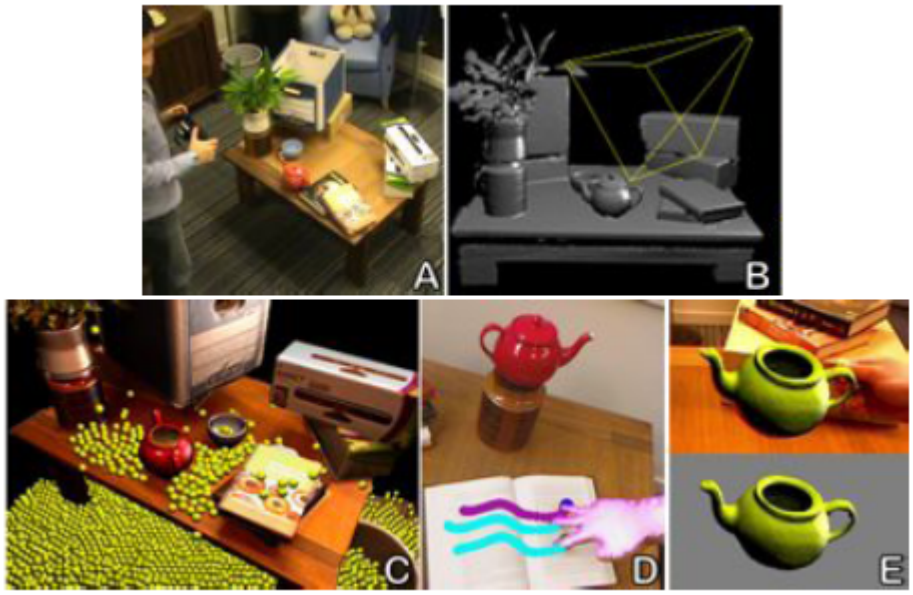

Figura 4.2: Principali campi applicativi di Microsoft Kinect Fusion [18].

sistemi intelligenti. Tuttavia, Kinect Fusion è molto di più in quanto tutto funziona in tempo reale, a basso costo e senza la necessità di un'utenza particolarmente esperta. Proprio per quest'ultimo motivo, tutte le community di sviluppo hanno collaborato alla definizione di un protocollo di intesa finalizzato a una flessibilità nello sviluppo di applicazioni complesse dotate però di comode interfacce di interazione da parte dell'utente.

Analogamente ad altre architetture di protocolli strutturati, è stata definita una gerarchia e una pipeline che stratifica i componenti applicativi – Figura 4.3. Tale sequenza è definita attraverso le seguenti fasi [18]:

Depth Map Conversion Prendendo come esempio la relatività del sistema di riferimento che viene convenzionalmente utilizzata come principio nell'ambito dell'osservazione astronomica, è possibile immaginare che esista un legame simile tra il soggetto osservato e il sistema di riferimento rispetto al punto di osservazione. Nello specifico, la seguente stratificazione si occupa di convertire la mappa di profondità acquisita da Microsoft Kinect in tempo reale dalle coordinate dell'immagine in un sistema tridimensionale di punti rispetto al riferimento spaziale della telecamera.

Camera Tracking Questa seconda fase mira a fornire un dettaglio sempre più preciso delle coordinate spaziali del soggetto relativamente ai frame

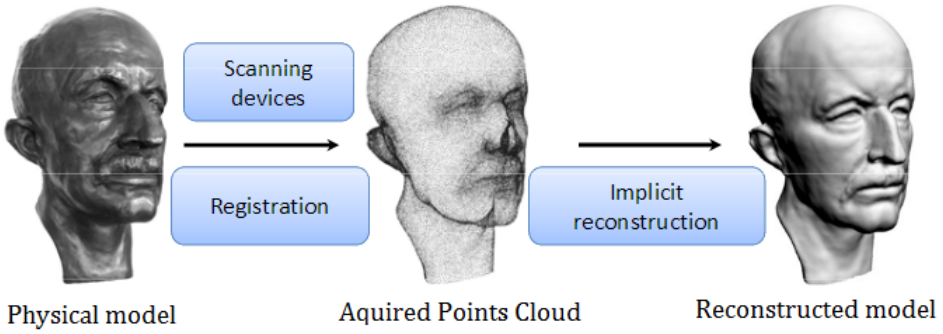

Figura 4.3: Principio di funzionamento di Microsoft Kinect Fusion [49].

fin qui acquisiti. Naturalmente, tale precisione aumenta al crescere del numero di frame e agli spostamenti della telecamera in quanto l'interazione determina una convergenza incrementale. Specificatamente, il seguente livelllo è applicato per computare una trasformazione rigida dei 6 gradi di libertà potenzialmente esprimibili al fine di allineare la corrente mappa di profondità relativamente ai frame precedenti. Tale operatività avviene in forma automatica attraverso l'interpretazione del famoso algoritmo ICP(Iterative Closest Point).

Volumetric Integration Le Point Cloud generate dinamicamente rappresentano in realtà dei buffer omogenei che vengono successivamente fusi in una immagine tridimensionale completa basata su un sistema di riferimento globale. In base ai movimenti della telecamera, il sistema di riferimento viene costantemente aggiornato per permettere il ricalcolo più accurato della posizione dei voxel (per voxel si intende una unità di misura in volume nell'ambito di un sistema di rappresentazione tridimensionale analogamente a quanto sono intesi i pixel nell'ambito di sistemi bidimensionali) nella griglia tridimensionale.

Raycasting Infine, la fase di irraggiamento sul soggetto permette di estrarre l'immagine del soggetto rispetto alla scena complessiva determinando un contrasto visibile all'utente.

Queste fasi vengono eseguite in forma ciclica e in modo parallelo in modo tale da gestire una sorta di interoperabilità tra scena e sistema di riferimento all'interno della GPU (Graphics Processing Unit) e interagendo attraverso l'architettura CUDA (Compute Unified Device Architecture).

È interessante ricordare che tutte queste attività, molto sofisticate e articolate tra di loro condividono un dispositivo che ha un prezzo di vendita molto basso rispetto al valore della tecnologia.

4.1.1 Conversione della mappa di profondità

Per gestire gli spostamenti di Microsoft Kinect nello spazio, è stato sviluppato un algoritmo che sfrutta la conversione della mappa di profondità in fase di acquisizione basato su corrispondenze matriciali [18, 27]. In particolare, Kinect Fusion associa dinamicamente ogni singolo pixel ad uno specifico thread intendendo con tale definizione un task parallelo eseguito concorrentemente ad altri sottoprocessi. In questo modo le mappe di pixel sono computabili attraverso una traslazione dinamica parallela.

Al tempo i-esimo, viene applicato un filtro bilaterale (*Sezione*3.1.2) alla mappa di profondità acquisita $R_i(\vec{u})$ al fine di operare uno smoothing dei dati raw,

$$D_i(\vec{u}) = \frac{1}{W_{\vec{p}}} \sum_{\vec{q} \in \Omega} R_i(\vec{q}) N_{\sigma_s}(\|\vec{u} - \vec{q}\|) N_{\sigma_r}(|R_i(\vec{u}) - R_i(\vec{q})|). \tag{4.1}$$

Successivamente, ogni thread generato attraverso architettura CUDA opera in parallelo ad ogni pixel $\vec{u} = (x, y)$ ed in base alla matrice di profondità ne vengono derivate altre due allo scopo di contenere rispettivamente i vertici e le normali agli stessi (per vertici si intende semplicemente dei punti nello spazio tridimensionale). Di conseguenza, data la matrice di calibrazione intrinseca K della telecamera a infrarossi di Kinect, ogni thread, operando su un singolo pixel, ricalcola la distanza come un vertice tridimensionale nel sistema di riferimento della telecamera:

$$\vec{v}_i(\vec{u}) = D_i(\vec{u}) K^{-1}[\vec{u}, 1]. \tag{4.2}$$

A partire dalla matrice dei vertici è possibile derivare la matrice delle normali con una relazione 1:1 per ogni vertice considerando le corrispondenze tra i vertici vicini. Infatti, dato che ogni frame rappresenta la misura di una superficie in una griglia regolare, la computazione delle normali può avvenire utilizzando il prodotto vettoriale tra i vertici vicini della mappa:

$$\vec{n}_i(\vec{u}) = (\vec{v}_i(x+1, y) - \vec{v}_i(x, y)) \times (\vec{v}_i(x, y+1) - \vec{v}_i(x, y)). \tag{4.3}$$

In particolare, quindi le matrici di vettori sopra riportate rappresentano un buffer di lavoro che viene applicato tramite regole di trasformazione al sistema di riferimento globale. La gestione della posizione della telecamera nei sei gradi di libertà al tempo i-esimo rispetto al sistema di riferimento globale sarà descritta da una trasformazione matriciale rigida $T_i = [R_i|\vec{t_i}]$ contenente la matrice di rotazione 3x3 R_i ed il vettore di traslazione $\vec{t_i}$. A partire da questa trasformazione, i vertici e le normali potranno conseguentemente essere

convertiti nel sistema di riferimento globale di coordinate:

$$\vec{v}_i^g(\vec{u}) = T_i \vec{v}_i(\vec{u}), \tag{4.4}$$

$$\vec{n}_i^g(\vec{u}) = R_i \vec{n}_i(\vec{u}). \tag{4.5}$$

4.1.2 Camera Tracking

Precedentemente abbiamo esaminato come la mappa dei pixel sia relazionata attraverso il sistema di riferimento della telecamera (*Sezione*4.1.1), di qui in avanti, scenderemo in maggior profondità determinando i modelli che permettono la trasformazione $T_i = [R_i, \vec{t_i}]$ verso il modello globale.

Come molto probabilmente si potrebbe intuire, la ricostruzione tridimensionale della scena può avvenire solamente in funzione di un'associazione dinamica delle point cloud ottenute per tramite del sistema di acquisizione. Queste ultime devono infatti essere messe in relazione tra di loro in un sistema di riferimento globale statico attraverso un'opportuna sincronizzazione [18, 27].

Uno dei problemi più frequenti nell'ambito dei sistemi matriciali riguarda la sincronizzazione delle nuvole di punti. Supponiamo siano date le coordinate di due insiemi di punti in differenti sistemi di riferimento. Il problema di determinare la trasformazione rigida ottimale tra i due sistemi è un problema di Absolute orientation e tale sincronizzazione è stata risolta con un algoritmo denominato ICP (Iterative Closest Point). La filosofia che sta alla base di questo algoritmo è una analisi con una iterazione ricorsiva verificando la migliore vicinanza di coppie di punti chiamati corrispondenze. Conseguentemente viene a crearsi la situazione di migliore equilibrio nell'architettura geometrica di riferimento.

Prima di procedere nella descrizione particolareggiata di tale algoritmo, in funzione di una visione più strutturata ed organica delle argomentazioni, si rivela necessaria l'introduzione di alcuni concetti analitici inerenti i quaternioni e le rotazioni nello spazio.

4.1.2.1 Quaternioni

La seguente sezione, ha lo scopo di introdurre alcuni concetti matematici inerenti i quaternioni e le rotazioni nello spazio. Per una trattazione più estesa è consigliabile l'approfondimento personale attraverso consultazione di un testo avanzato di Algebra lineare e Geometria.

In analisi matematica, i quaternioni sono un insieme di numeri che estende l'insieme dei numeri complessi. Questi sono spesso utilizzati in applicazioni di

Computer Vision e Computer Graphics, in particolare nell'ambito del calcolo numerico delle rotazioni spaziali.

Si definisce quaternione un elemento dello spazio lineare $\mathbb{H}(\mathbb{R})$ a quattro dimensioni:

$$\vec{q} = q_0 + q_1 i + q_2 j + q_3 k = (q_0, q_1, q_2, q_3), \qquad q_i \in \mathbb{R}, i = 0, ..., 3. \tag{4.6}$$

Considerando la sovrapposizione della parte reale $\vec{q}_r = q_0$ e vettoriale $\vec{q}_v = q_1 i + q_2 j + q_3 k$ è possibile rappresentare il quaternione attraverso la relazione semplificata:

$$\vec{q} = q_r + \vec{q}_v = (q_r, \vec{q}_v). \tag{4.7}$$

I numeri i, j, k, sono detti numeri ipercomplessi e soddisfano la relazione anticommutativa:

$$i^2 = j^2 = k^2 = ijk = -1. \tag{4.8}$$

A partire dalla relazione sopra descritta è possibile definire tutti i prodotti degli elementi base:

$$ij = k, \qquad ji = -k, \tag{4.9}$$
$$jk = i, \qquad kj = -i, \tag{4.10}$$
$$ki = j, \qquad ik = -j. \tag{4.11}$$

Algebra dei quaternioni Di seguito si riportano le principali proprietà algebriche dei quaternioni in forma sintetica.

Dato il quaternione

$$\vec{q} = q_0 + q_1 i + q_2 j + q_3 k = (q_0, q_1, q_2, q_3) = q_r + \vec{q}_v = (q_r, \vec{q}_v),$$

la norma del quaternione è definita come:

$$\|\vec{q}\| = \sqrt{\vec{q}\vec{q}^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \tag{4.12}$$

Sono conseguentemente caratterizzati i seguenti quaternioni notevoli:

- Quaternione nullo

$$\vec{0} = 0 + 0i + 0j + 0k = (0, 0, 0, 0) = \vec{0} + \vec{0} = (\vec{0}, \vec{0}); \tag{4.13}$$

- Quaternione unitario

$$\|\vec{q}\| = 1; \tag{4.14}$$

- Quaternione coniugato

$$\vec{q}^{*} = q_0 - q_1 i - q_2 j - q_3 k = (q_0, -q_1, -q_2, -q_3) = q_r - \vec{q}_v = (q_r, \vec{q}_v); \quad (4.15)$$

$$(\vec{q}^{*})^{*} = \vec{q} \qquad \|\vec{q}\| = \|\vec{q}^{*}\|; \quad (4.16)$$

- Quaternione puro

$$\vec{q}_v = 0 + q_1 i + q_2 j + q_3 k = (0, q_1, q_2, q_3) = 0 + \vec{q}_v = (0, \vec{q}_v); \quad (4.17)$$

$$\vec{q}_v^{*} = -\vec{q}_v. \quad (4.18)$$

- Quaternione inverso $\vec{q}^{-1}$:

$$\vec{q}\vec{q}^{-1} = (1, 0, 0, 0). \quad (4.19)$$

  È importante notare che, nel caso di quaternione unitario, il quaternione inverso coincide con il quaternione coniugato:

$$\vec{q}^{*}\vec{q}\vec{q}^{-1} = \|\vec{q}\|^2 \vec{q}^{-1} = \vec{q}^{*}.$$

  Infatti, sviluppando la relazione precedente si ottiene intuitivamente che:

$$\vec{q}^{-1} = \vec{q}^{*}. \quad (4.20)$$

Dati due quaternioni

$$\vec{h} = h_0 + h_1 i + h_2 j + h_3 k = (h_0, h_1, h_2, h_3) = h_r + \vec{h}_v = (h_r, \vec{h}_v),$$

$$\vec{g} = g_0 + g_1 i + g_2 j + g_3 k = (g_0, g_1, g_2, g_3) = g_r + \vec{g}_v = (g_r, \vec{g}_v),$$

sono definite le operazioni di:

- Somma

$$\vec{h} + \vec{g} = (h_0 + g_0, h_1 + g_1, h_2 + g_2, h_3 + g_3) = (h_r + g_r, \vec{h}_v + \vec{g}_v); \quad (4.21)$$

- Differenza

$$\vec{h} - \vec{g} = (h_0 - g_0, h_1 - g_1, h_2 - g_2, h_3 - g_3) = (h_r - g_r, \vec{h}_v - \vec{g}_v); \quad (4.22)$$

- Prodotto

$$\begin{aligned}\vec{h}\vec{g} &= h_0g_0 + h_0g_1i + h_0g_2j + h_0g_3k + \\ &+ h_1g_0i + h_1g_1i^2 + h_1g_2ij + h_1g_3ik + \\ &+ h_2g_0j + h_2g_1ij + h_2g_2j^2 + h_2g_3jk + \\ &+ h_3g_0k + h_3g_1ik + h_3g_2jk + h_3g_3k^2 = \\ &= (h_0g_0 - h_1g_1 - h_2g_2 - h_3g_3) + \\ &+ (h_1g_0 + h_0g_1 - h_3g_2 + h_2g_3)i + \\ &+ (h_2g_0 + h_3g_1 + h_0g_2 - h_1g_3)j + \\ &+ (h_3g_0 - h_2g_1 + h_1g_2 + h_0g_3)k = \\ &= (h_rg_r - \vec{h}_v \cdot \vec{g}_v, h_r\vec{g}_v + g_r\vec{h}_v + \vec{h}_v \times \vec{g}_v),\end{aligned} \tag{4.23}$$

con:

$$\vec{h}_v \cdot \vec{g}_v = \sum_i \vec{h}_{v_i}\vec{g}_{v_i} = \vec{h}_v^t\vec{g}_v = \vec{g}_v^t\vec{h}_v,$$

prodotto scalare e

$$\vec{h}_v \times \vec{g}_v = det \begin{bmatrix} i & j & k \\ h_1 & h_2 & h_3 \\ g_1 & g_2 & g_3 \end{bmatrix},$$

prodotto vettoriale.

Il prodotto tra due quaternioni soddisfa le seguenti proprietà:

$$(\vec{h}\vec{g})^* = \vec{h}^*\vec{g}^*, \qquad \|\vec{h}\vec{g}\| = \|\vec{h}\|\|\vec{g}\|. \tag{4.24}$$

Si può inoltre osservare che similmente al prodotto definito sui numeri complessi, il prodotto definito sui quaternioni è non commutativo (essendo $\vec{h}_v \times \vec{g}_v = -\vec{g}_v \times \vec{h}_v$).

Si può inoltre espandere il prodotto di due quaternioni nel seguente modo:

$$\vec{h}\vec{g} = \begin{bmatrix} h_0 & -h_1 & -h_2 & -h_3 \\ h_1 & h_0 & -h_3 & h_2 \\ h_2 & h_3 & h_0 & -h_1 \\ h_3 & -h_2 & h_1 & h_0 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = Q_L\vec{g} \tag{4.25}$$

oppure,

$$\vec{h}\vec{g} = \begin{bmatrix} g_0 & -g_1 & -g_2 & -g_3 \\ g_1 & g_0 & -g_3 & g_2 \\ g_2 & g_3 & g_0 & -g_1 \\ g_3 & -g_2 & g_1 & g_0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} = Q_R\vec{h} \tag{4.26}$$

Quaternioni e rotazioni Nello spazio tridimensionale, in base al teorema di Eulero, le rotazioni di un corpo rispetto al sistema di riferimento sono equivalenti alla rotazione di un angolo ϑ attorno a un asse $\vec{u}_n$ rispetto all'origine. I quaternioni riescono a rappresentare questa trasformazione con una quadrupla.

Dato un quaternione unitario:

$$\vec{u} = (u_0, u_1, u_2, u_3) = (u_0, \vec{u}_n) = e^{\vec{u}_n \vartheta} = \cos\vartheta + \vec{u}_n \sin\vartheta, \tag{4.27}$$

questo rappresenta la rotazione di un angolo 2ϑ attorno all'asse rappresentato dal versore $\vec{u}_n = (u_1, u_2, u_3)^t$. Viceversa, data la rotazione ϑ attorno all'asse rappresentato dal versore $\vec{u}_n = (u_1, u_2, u_3)^t$, il quaternione unitario che rappresenta la rotazione è dato da:

$$\vec{u} = \left(\cos\frac{\vartheta}{2}, \sin\frac{\vartheta}{2}u_1, \sin\frac{\vartheta}{2}u_2, \sin\frac{\vartheta}{2}u_3\right) = \left(\cos\frac{\vartheta}{2}, \vec{u}_n \sin\frac{\vartheta}{2}\right) = \cos\frac{\vartheta}{2} + \vec{u}_n \sin\frac{\vartheta}{2}. \tag{4.28}$$

Una rotazione rigida nello spazio tridimensionale può essere rappresentata attraverso una matrice di rotazione 3×3 R. Considerando che un quaternione unitario rappresenta anch'esso una rotazione nello spazio, è possibile associare ad ogni matrice di rotazione un quaternione unitario e viceversa.

Una delle proprietà più tipiche dei vettori prevede che la lunghezza di un vettore non venga influenzata dalla rotazione come diversamente avviene all'angolo tra due vettori. Una delle conseguenze di questa affermazione è il fatto che la rotazione non influenza il prodotto scalare anche se inverte il verso del prodotto vettoriale. Si potrebbe ipotizzare di rappresentare le rotazioni attraverso un semplice prodotto tra quaternioni considerando che il prodotto tra quaternioni preserva il prodotto scalare. Tuttavia, il prodotto tra due quaternioni non è adatto a rappresentare correttamente la rotazione in base al fatto che il prodotto tra un quaternione e un vettore non risulta essere necessariamente un vettore. In base alle precedenti considerazioni, è possibile rappresentare le rotazioni con un prodotto composto. Considerato un vettore $\vec{p} = (0, \vec{p}_v)$, il vettore $\vec{p}_k$ ruotato rispetto al quaternione $\vec{q} = (q_r, \vec{q}_v) = \left(\cos\frac{\vartheta}{2}, \vec{q}_n \sin\frac{\vartheta}{2}\right)$, attorno all'asse $\vec{q}_n$ di un angolo ϑ si ottiene con il prodotto:

$$\vec{p}_k = \vec{q}\vec{p}\vec{q}^{-1}. \tag{4.29}$$

Per un quaternione unitario (4.20):

$$\vec{p}_k = \vec{q}\vec{p}\vec{q}^{*}. \tag{4.30}$$

Il teorema precedente è dimostrabile sviluppando tale prodotto di seguito.

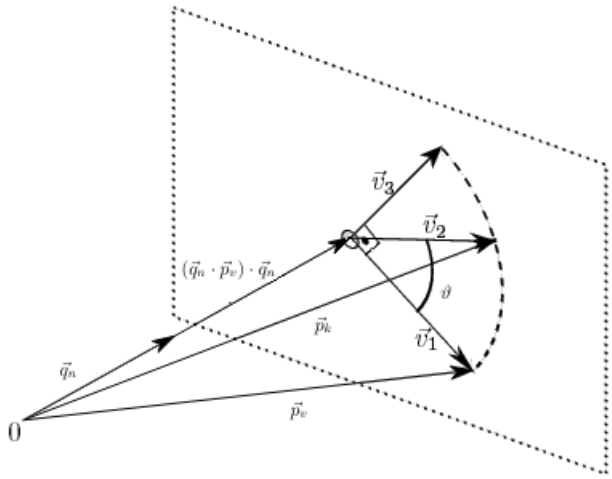

Figura 4.4: Quaternioni e rotazioni spaziali [11].

Utilizzando due volte la proprietà del prodotto tra due quaternioni (4.23) e sfruttando le proprietà del prodotto vettoriale $\vec{a} \times (\vec{b} \times \vec{c}) = \vec{b}(\vec{c} \cdot \vec{a}) - \vec{c}(\vec{a} \cdot \vec{b})$ si ottiene:

$$\begin{aligned}
\vec{p}_k &= (q_r, \vec{q}_v)(0, \vec{p}_v)(q_r, -\vec{q}_v) = \\
&= (-\vec{q}_v \cdot \vec{p}_v, q_r\vec{p}_v + \vec{q}_v \times \vec{p}_v)(q_r, -\vec{q}_v) = \\
&= \Big((-\vec{q}_v \cdot \vec{p}_v)q_r + (q_r\vec{p}_v + \vec{q}_v \times \vec{p}_v) \cdot \vec{q}_v, q_r^2\vec{p}_v + 2q_r(\vec{q}_v \times \vec{p}_v) + (\vec{q}_v \cdot \vec{p}_v)\vec{q}_v - (\vec{q}_v \times \vec{p}_v) \times \vec{q}_v \Big) = \\
&= \Big(0, 2(\vec{q}_v \cdot \vec{p}_v)\vec{q}_v + (q_r^2 - \vec{q}_v \cdot \vec{q}_v)\vec{p}_v + 2q_r(\vec{q}_v \times \vec{p}_v) \Big).
\end{aligned}$$

Esprimendo il quaternione di trasformazione $\vec{q}$ secondo la notazione di Eulero (4.27) risulta:

$$\vec{p}_k = 2\sin^2\frac{\vartheta}{2}(\vec{q}_n \cdot \vec{p}_v)\vec{q}_n + \left(\cos^2\frac{\vartheta}{2} - \sin^2\frac{\vartheta}{2}\right)\vec{p}_v + 2\cos\frac{\vartheta}{2}\sin\frac{\vartheta}{2}\vec{q}_n \times \vec{p}_v.$$

Utilizzando le formule di bisezione possiamo ulteriormente semplificare l'equazione:

$$\vec{p}_k = (1 - \cos\vartheta)(\vec{q}_n \cdot \vec{p}_v)\vec{q}_n + \cos\vartheta\vec{p}_v + \sin\vartheta\vec{q}_n \times \vec{p}_v.$$

Quest'ultima è la formula di Rodrigues e rappresenta la rotazione nello spazio di un angolo ϑ attorno a un asse $\vec{q}_n$. È possibile dimostrare tale

affermazione considerando la decomposizione in Figura 4.4:

$$
\begin{aligned}
\vec{v}_2 &= \cos\vartheta\vec{v}_1 + \sin\vartheta\vec{v}_3,\\
\vec{v}_1 &= \vec{p}_v - (\vec{q}_n \cdot \vec{p}_v)\vec{q}_n\\
\vec{v}_3 &= \vec{v}_1 \times \vec{q}_n =\\
&= (\vec{p}_v - (\vec{q}_n \cdot \vec{p}_v)\vec{q}_n) \times \vec{q}_n =\\
&= \vec{p}_v \times \vec{q}_n - (\vec{q}_n \cdot \vec{p}_v)\vec{q}_n \times \vec{q}_n =\\
&= \vec{p}_v \times \vec{q}_n\\
\vec{v}_2 &= \cos\vartheta(\vec{p}_v - (\vec{q}_n \cdot \vec{p}_v)\vec{q}_n) + \sin\vartheta(\vec{p}_v \times \vec{q}_n),\\
\vec{p}_k &= (\vec{q}_n \cdot \vec{p}_v)\vec{q}_n + \vec{v}_2 =\\
&= (\vec{q}_n \cdot \vec{p}_v)\vec{q}_n + \cos\vartheta(\vec{p}_v - (\vec{q}_n \cdot \vec{p}_v)\vec{q}_n) + \sin\vartheta(\vec{p}_v \times \vec{q}_n) =\\
&= (1 - \cos\vartheta)(\vec{q}_n \cdot \vec{p}_v)\vec{q}_n + \cos\vartheta\vec{p}_v + \sin\vartheta\vec{q}_n \times \vec{p}_v.
\end{aligned}
$$

Procediamo ora a determinare la relazione che lega i quaternioni alle matrici di rotazione prendendo in consideriamo il prodotto di partenza (4.30). È possibile utilizzare la proprietà del prodotto dei quaternioni (4.23) in modo tale che:

$$\vec{p}_k = \vec{q}\vec{p}\vec{q}^* = Q_L\vec{p}\vec{q}^* = Q_L Q_R^t\vec{p}.$$

Sviluppando il prodotto delle due matrici si ottiene:

$$
Q_L Q_R^t = \begin{bmatrix}
q_0^2 + q_1^2 + q_2^2 + q_3^2 & 0 & 0 & 0\\
0 & q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_3q_0) & 2(q_1q_3 + q_2q_0)\\
0 & 2(q_1q_2 + q_3q_0) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_1q_0)\\
0 & 2(q_1q_3 - q_2q_0) & 2(q_2q_3 + q_1q_0) & q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{bmatrix}.
$$

Considerando che il primo termine coincide con la norma unitaria del quaternione, questo non presenta alcun effetto sulla moltiplicazione matrice vettore. Conseguentemente, associando la relazione appena derivata con la trasformazione inerente la rotazione di un punto nello spazio tridimensionale $\vec{p}_k = R\vec{p}_v$, è possibile determinare la relazione che lega i quaternioni unitari con le rotazioni nello spazio:

$$
R = \begin{bmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_3q_0) & 2(q_1q_3 + q_2q_0)\\
2(q_1q_2 + q_3q_0) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_1q_0)\\
2(q_1q_3 - q_2q_0) & 2(q_2q_3 + q_1q_0) & q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{bmatrix}. \qquad (4.31)
$$

4.1.2.2 Iterative Closest Point

Con gli strumenti matematici presentati nella Sezione 4.1.2.1 è ora possibile descrivere con maggior accuratezza l'algoritmo ICP [4, 7, 15, 18, 25, 27, 42].

La sincronizzazione di due nuvole di punti nello spazio può essere immaginata come il risultato di una trasformazione tra i due sistemi di rifermento in modo tale che le origini coincidano e gli assi omologhi siano tra loro paralleli. La trasformazione è per sua natura rigida, ovvero, non modifica le posizioni relative dei soggetti ma solamente le coordinate di tali posizioni e può essere determinata come la somma di una rotazione e di una traslazione – Figura 4.5. Questa rototraslazione lega tra loro coppie di punti dette corrispondenze aventi differenti coordinate di posizione ma rappresentanti fisicamente lo stesso punto. Intuitivamente si evince dunque che l'algoritmo è implementabile solamente nel caso in cui vi sia una sovrapposizione minima tra le varie acquisizioni di punti.

Considerando che non è possibile determinare analiticamente la trasformazione ottima tra i due insiemi di punti a causa dell'incertezza di misurazione, si cercherà di sviluppare un algoritmo iterativo che, a partire dalle corrispondenze, permetta l'approssimazione della rotazione e della traslazione ottime minimizzando una specifica funzione errore e riducendo in questo modo la distanza tra i punti omologhi.

I dati nello spazio possono essere rappresentati come [4]:

- Insiemi di punti;
- Insiemi di segmenti;
- Insiemi di triangoli;
- Insiemi di entità parametriche;
- Insiemi di entità implicite.

Può essere determinata la distanza d di un singolo punto $\vec{p}$ da un insieme di riferimento X:

$$d(\vec{p}, X) = min\big\{\|\vec{x} - \vec{p}\|, \vec{x} \in X\big\}. \tag{4.32}$$

Il punto più vicino $\vec{y}$ appartenente all'insieme X che minimizza la distanza dal punto $\vec{p}$, soddisfa la relazione:

$$d(\vec{p}, \vec{y}) = d(\vec{p}, X). \tag{4.33}$$

Il calcolo della distanza per insiemi semplici come insiemi di punti o di segmenti è di facile intuizione mentre per insiemi costituiti da entità parametriche o

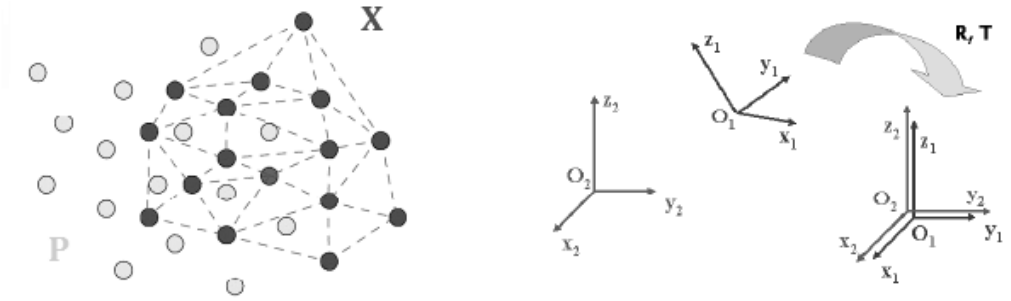

Figura 4.5: Caratterizzazione geometrica dell'algoritmo ICP [50].

implicite, può essere più problematico ed è necessario l'utilizzo di metodi iterativi come il metodo di Newton, oppure di metodi numerici come il metodo dei moltiplicatori di Lagrange. Al fine di una documentazione tecnica più dettagliata relativamente alla minimizzazione della distanza euclidea è possibile consultare l'Appendice A.

Ipotizziamo $P = \{\vec{p}_i\}$ un insieme di punti ottenuto per mezzo della tecnologia di acquisizione che deve essere allineato con un riferimento $X = \{\vec{x}_i\}$, $N_x = N_p$, dove ad ogni punto $\vec{p}_i$ corrisponde un punto $\vec{x}_i$ del medesimo indice. Nell'ipotesi che per piccoli spostamenti del sistema di acquisizione le nuvole di punti siano in prima approssimazione ben allineate, l'algoritmo ICP determina la corrispondenza dei punti tra i due insiemi assumendo che i punti omologhi siano quelli più vicini tra i due insiemi. Questo avviene formalmente cercando di determinare il punto più vicino (Closest Point) $\vec{y}_i \in X$ ad ogni singolo punto $\vec{p}_i$ della point cloud appena acquisita P.

Per basarci su una notazione di riferimento compatta, delineiamo ora come C l'operatore Closest point e come $Y = \{\vec{y}_i\}$ l'insieme delle corrispondenze dei punti vicini risultante dall'applicazione di tale operatore, in modo che sia verificata la relazione $Y = C(P, X)$.

Computazione della trasformazione ottima Al fine di determinare la trasformazione ottima tra i sistemi di riferimento, necessaria alla sincronizzazione delle nuvole di punti, verrà definita una funzione obiettivo la cui minimizzazione implicherà l'allineamento dei due insiemi [15]. Considerando l'obiettivo di una trasformazione tra i due sistemi di riferimenti del tipo:

$$\vec{y}_i = R\vec{p}_i + \vec{t},$$

con R matrice di rotazione e $\vec{t}$ offset di traslazione, la funzione errore è definita come la sommatoria degli scarti ai minimi quadrati delle coppie di punti,

avendo come riferimento la rototraslazione dei due insiemi da allineare:

$$f(\vec{q}) = \frac{1}{N_p}\sum_{i=1}^{N_p}\|\vec{y}_i - R\vec{p}_i - \vec{t}\|^2. \tag{4.34}$$

È utile a questo punto riferire tutte le misurazioni rispetto a $\vec{\mu_p}$ e $\vec{\mu_y}$, centri di massa rispetto agli insiemi di punti P e Y:

$$\vec{\mu_p} = \frac{1}{N_p}\sum_{i=1}^{N_p}\vec{p}_i, \qquad \vec{\mu_y} = \frac{1}{N_y}\sum_{i=1}^{N_y}\vec{y}_i.$$

Le nuove coordinate sono espresse secondo la notazione:

$$\vec{p}_i^{\,c} = \vec{p}_i - \vec{\mu_p}, \qquad \vec{y}_i^{\,c} = \vec{y}_i - \vec{\mu_y},$$

e la funzione errore viene riscritta come:

$$f(\vec{q}) = \frac{1}{N_p}\sum_{i=1}^{N_p}\|\vec{y}_i^{\,c} - R\vec{p}_i^{\,c} - \vec{t}^{\,c}\|^2,$$

dove l'offset di traslazione è definito come:

$$\vec{t}^{\,c} = \vec{t} - \vec{\mu_y} + R\vec{\mu_p}.$$

Se espandiamo la sommatoria con la proprietà del quadrato di un binomio otteniamo:

$$f(\vec{q}) = \frac{1}{N_p}\sum_{i=1}^{N_p}\|\vec{y}_i^{\,c} - R\vec{p}_i^{\,c}\|^2 - 2\frac{\vec{t}^{\,c}}{N_p}\sum_{i=1}^{N_p}[\vec{y}_i^{\,c} - R\vec{p}_i^{\,c}] + \|\vec{t}^{\,c}\|^2.$$

Il termine centrale è identicamente nullo considerando che tutte le misure in questo caso sono riferite ai baricentri mentre il primo ed il terzo termine sono sempre positivi. Essendo che il primo termine non dipende da $\vec{t}^{\,c}$, l'errore totale risulta minimo rispetto alla traslazione se $\vec{t}^{\,c} = 0$ ovvero se:

$$\vec{t} = \vec{\mu_y} - R\vec{\mu_p} \tag{4.35}$$

In base a questa considerazione è possibile riscrivere la funzione obiettivo come:

$$f(\vec{q}) = \frac{1}{N_p}\sum_{i=1}^{N_p}\|\vec{y}_i^{\,c} - R\vec{p}_i^{\,c}\|^2,$$

ed utilizzando nuovamente la proprietà del quadrato del binomio si ottiene:

$$f(\vec{q}) = \frac{1}{N_p}\sum_{i=1}^{N_p}\|\vec{y}_i^c\|^2 - 2\frac{1}{N_p}\sum_{i=1}^{N_p}[\vec{y}_i^c \cdot R\vec{p}_i^c] + \sum_{i=1}^{N_p}\|\vec{p}_i^c\|^2.$$

Considerando che il primo ed il terzo termine sono sempre positivi, la funzione è minimizzata se il termine centrale $\sum_{i=1}^{N_p} \vec{y}_i^c \cdot R\vec{p}_i^c$ diventa più grande possibile.

Avendo chiare le proprietà dei quaternioni è possibile riscrivere quest'ultimo con il prodotto composto (4.23):

$$\sum_{i=1}^{N_p} \vec{y}_i^c \cdot R\vec{p}_i^c = \sum_{i=1}^{N_p} \vec{y}_i^c \cdot \vec{q}\vec{p}_i^c\vec{q}^*.$$

ed utilizzando le proprietà del prodotto:

$$\sum_{i=1}^{N_p} \vec{y}_i^c \cdot \vec{q}\vec{p}_i^c\vec{q}^* = \sum_{i=1}^{N_p}(\vec{q}\vec{p}_i^c) \cdot (\vec{y}_i^c\vec{q}).$$

Se i due vettori possono essere scritti secondo la notazione $\vec{p}_i^c = (0, \vec{p}_{i_x}^c, \vec{p}_{i_y}^c, \vec{p}_{i_z}^c)^t$ e $\vec{y}_i^c = (0, \vec{y}_{i_x}^c, \vec{y}_{i_y}^c, \vec{y}_{i_z}^c)^t$, allora utilizzando le proprietà del prodotto dei quaternioni (4.23) si ottiene:

$$\vec{q}\vec{p}_i^c = \begin{bmatrix} 0 & -\vec{p}_{i_x}^c & -\vec{p}_{i_y}^c & -\vec{p}_{i_z}^c \\ \vec{p}_{i_x}^c & 0 & \vec{p}_{i_z}^c & -\vec{p}_{i_y}^c \\ \vec{p}_{i_y}^c & -\vec{p}_{i_z}^c & 0 & \vec{p}_{i_x}^c \\ \vec{p}_{i_z}^c & \vec{p}_{i_y}^c & -\vec{p}_{i_x}^c & 0 \end{bmatrix} \vec{q} = Q_{p_i}\vec{q}$$

e

$$\vec{y}_i^c\vec{q} = \begin{bmatrix} 0 & -\vec{y}_{i_x}^c & -\vec{y}_{i_y}^c & -\vec{y}_{i_z}^c \\ \vec{y}_{i_x}^c & 0 & \vec{y}_{i_z}^c & -\vec{y}_{i_y}^c \\ \vec{y}_{i_y}^c & -\vec{y}_{i_z}^c & 0 & \vec{y}_{i_x}^c \\ \vec{y}_{i_z}^c & \vec{y}_{i_y}^c & -\vec{y}_{i_x}^c & 0 \end{bmatrix} \vec{q} = Q_{y_i}\vec{q}$$

e di conseguenza:

$$\begin{aligned} \sum_{i=1}^{N_p}(\vec{q}\vec{p}_i^c) \cdot (\vec{y}_i^c\vec{q}) &= \sum_{i=1}^{N_p}(Q_{p_i}\vec{q}) \cdot (Q_{y_i}\vec{q}) = \\ &= \sum_{i=1}^{N_p}(\vec{q}^t Q_{p_i}^t) \cdot (Q_{y_i}\vec{q}) = \\ &= \vec{q}^t\left(\sum_{i=1}^{N_p} Q_{p_i}^t \cdot Q_{y_i}\right)\vec{q} \\ &= \vec{q}^t Q(\Sigma_{py})\vec{q} \end{aligned} \tag{4.36}$$

La matrice $Q(\Sigma_{py})$ rappresenta la trasformazione dalla quale verranno estratti i parametri necessari alla sincronizzazione delle nuvole di punti. Dato che i concetti fin'ora illustrati potrebbero non risultare di facile comprensione, è utile parametrizzare la forma della matrice $Q(\Sigma_{py})$ attraverso una notazione più espressiva che sintetizza i passaggi fin'ora esposti [4].

Avendo chiaro il concetto di varianza come la misura della dispersione dei punti di una variabile aleatoria rispetto al valore atteso, è possibile estendere tale accezione considerando la correlazione tra due insiemi di variabili per tramite della matrice di covarianza:

$$\Sigma_{py} = \frac{1}{N_p}\sum_{i=1}^{N_p}[(\vec{p_i} - \vec{\mu_p})(\vec{y_i} - \vec{\mu_y})^t] = \frac{1}{N_p}\sum_{i=1}^{N_p}[\vec{p_i}\vec{y_i}^t] - \vec{\mu_p}\vec{\mu_y}^t. \tag{4.37}$$

Questo concetto è molto importante dal punto di vista concettuale perché permette una gestione più snella a tempo di esecuzione dell'algoritmo ICP.

A partire dalla matrice di covarianza, infatti, si può riscrivere la matrice $Q(\Sigma_{py})$ attraverso una notazione semplificata come:

$$Q(\Sigma_{py}) = \begin{bmatrix} tr(\Sigma_{py}) & \Delta^t \\ \Delta & \Sigma_{py} + \Sigma_{py}^t - tr(\Sigma_{py})I_3 \end{bmatrix}, \tag{4.38}$$

con $tr(\Sigma_{py}) = \sum_{i=1}^{n} \Sigma_{py_{ii}}$ traccia della matrice di covarianza, $\Delta = [A_{23}, A_{31}, A_{12}]^t$ vettore ottenuto dai coefficienti A_{ij} della matrice antisimmetrica $A = \Sigma_{py} - \Sigma_{py}^t$ e I_3 matrice di identità 3x3.

Successivamente alla determinazione della matrice (4.38), per determinare la rotazione che minimizza la funzione obiettivo è necessario definire il quaternione unitario $\vec{q}$ che massimizza il termine $\vec{q}^t Q(\Sigma_{py})\vec{q}$, inoltre, per rendere la rotazione significativa, assumiamo $\vec{q}$ correlato a $Q(\Sigma_{py})$ [15]. Per spiegare quest'ultima affermazione, si ipotizzi il movimento rotatorio circolare di una trottola su un piano orizzontale. La rotazione, in questo caso, rappresenta una trasformazione lineare attorno a un asse determinato da un autovettore. L'autovettore, infatti, nell'esempio considerato rappresenta la direzione invariante di trasformazione. Applicando questo concetto alle trasformazioni nelle tre dimensioni, ne viene determinata una quarta che rappresenta anch'essa la direzione attraverso la quale viene applicata la trasformazione. Quest'ultima è rappresentata dal quaternione $\vec{q}$.

La matrice 4x4 simmetrica $Q(\Sigma_{py})$ possiede 4 autovalori $\lambda_i, i = 1, ..., 4$. A partire da questi possono essere determinati 4 autovettori $\vec{e_i}, i = 1, ..., 4$ in modo tale che:

$$Q(\Sigma_{py})\vec{e_i} = \lambda_i \vec{e_i}, \qquad i = 1, ..., 4.$$

Gli autovettori si estendono in uno spazio a 4 dimensioni e di conseguenza possono essere scritti come combinazione lineari di un quaternione $\vec{q}$:

$$\vec{q} = \alpha_1\vec{e}_1 + \alpha_2\vec{e}_2 + \alpha_3\vec{e}_3 + \alpha_4\vec{e}_4.$$

Dato che gli autovettori sono per definizione ortogonali e che il quaternione è unitario si verifica facilmente che:

$$\vec{q}\vec{q} = \|\vec{q}\|^2 = \alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 = 1. \tag{4.39}$$

Si osserva ora che:

$$Q(\Sigma_{py})\vec{q} = \alpha_1\lambda_1\vec{e}_1 + \alpha_2\lambda_2\vec{e}_2 + \alpha_3\lambda_3\vec{e}_3 + \alpha_4\lambda_4\vec{e}_4.$$

dato che $\vec{e}_i, i = 1, ..., 4$ sono autovettori di $Q(\Sigma_{py})$.

Inoltre considerando che gli autovettori sono per definizione ortogonali tra loro:

$$\vec{q}^t Q(\Sigma_{py})\vec{q} = \alpha_1^2\lambda_1 + \alpha_2^2\lambda_2 + \alpha_3^2\lambda_3 + \alpha_4^2\lambda_4.$$

Supponiamo ora che gli autovalori siano ordinati secondo:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$$

è immediato osservare che la forma quadratica $\vec{q}^t Q(\Sigma_{py})\vec{q}$ non può essere maggiore del più grande degli autovalori (4.39):

$$\vec{q}^t Q(\Sigma_{py})\vec{q} \leq \alpha_1^2\lambda_1 + \alpha_2^2\lambda_2 + \alpha_3^2\lambda_3 + \alpha_4^2\lambda_4 \leq \lambda_1 \tag{4.40}$$

Abbiamo seguito tutte le singole fasi del procedimento dal punto di vista concettuale. Dal punto di vista implementativo, senza entrare nello specifico, vengono esposti alcuni suggerimenti utili per velocizzare le fasi di elaborazione.

Come molto probabilmente si potrebbe ricordare, il procedimento algebrico che consente di determinare gli autovalori è relativo alla risoluzione del polinomio caratteristico associato alla matrice (4.38):

$$det\big(Q(\Sigma_{py}\big) - \lambda I) = 0. \tag{4.41}$$

Una volta determinato l'autovalore λ_m che massimizza il termine $\vec{q}^t Q(\Sigma_{py})\vec{q}$, per trovare il corrispondente autovettore, è necessario risolvere l'equazione omogenea:

$$\big(Q(\Sigma_{py}) - \lambda_m I\big)\vec{e}_m = 0. \tag{4.42}$$

Evidentemente, sia il procedimento che le modalità di calcolo sono complesse e articolate, tuttavia, non entreremo nello specifico in quanto esistono delle

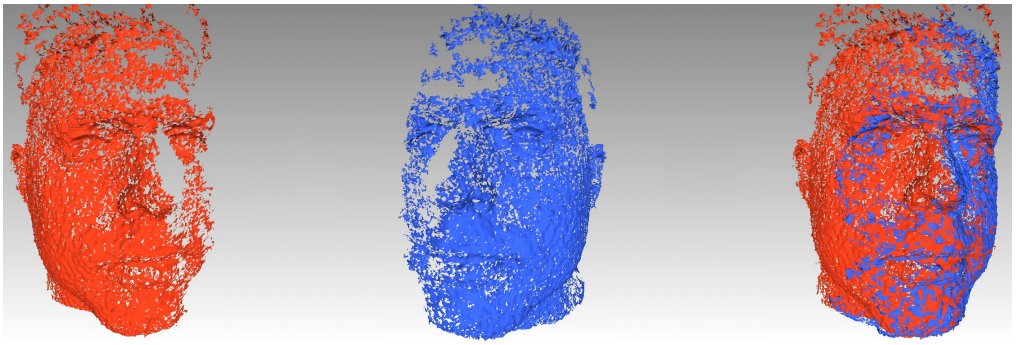

Figura 4.6: Sincronizzazione di point cloud mediante algoritmo ICP [2].

pratiche correnti standard volte alla risoluzione matematica dell'intera fase attraverso metodi numerici.

Il quaternione estratto dall'equazione omogenea (4.42), come precedentemente anticipato rappresenta la direzione invariante di rotazione. A quest'ultimo, sfruttando le proprietà dei quaternioni (4.31), è associabile la matrice di rotazione ottima R. Calcolata la rotazione ottima, il vettore che determina l'offset di traslazione (4.35) secondo la minimizzazione della funzione errore, è derivato semplicemente come:

$$\vec{t} = \vec{\mu_y} - R\vec{\mu_p}. \tag{4.43}$$

Di conseguenza, la trasformazione rigida tra i due sistemi di riferimento è definita da:

$$T = [R|\vec{t}]. \tag{4.44}$$

Ancora una volta, Per basarci su una notazione di riferimento compatta, dato $Y = C(P, X)$, insieme delle corrispondenze, definiamo la computazione della trasformazione rigida dei due insiemi nel rispetto della minimizzazione della funzione obiettivo con $(T, f(\vec{q})) = Q(P, Y)$. L'aggiornamento della posizione dell'insieme iniziale inoltre sarà invece determinata secondo la scrittura $P = T(P)$.

Algoritmo ICP esteso Il sistema finora sviluppato purtroppo non è molto preciso e si rivela dunque essenziale rendere iterativo l'algoritmo per minimizzare la distanza tra i punti dei due insiemi e di conseguenza raggiungere l'allineamento delle point cloud.

Considerate le varie premesse ed avendo chiare le notazioni esposte precedentemente relative alle varie fasi dell'algoritmo, si può quindi costruire l'algoritmo iterativo ICP secondo i seguenti passaggi [4]:

1. $Y_k = C(P_k, X)$, costo computazionale: $O(N_p N_x)$ worst case, $O(N_p \log N_x)$ average;
2. $(T_k, f(\vec{q})_k) = Q(P_0, Y_k)$, costo computazionale: $O(N_p)$;
3. $P_{k+1} = T_k(P_0)$, costo computazionale: $O(N_p)$;
4. if$(f(\vec{q})_k - f(\vec{q})_{k+1} < \tau)$ break;

Considerato che tutto il procedimento è molto complesso e sofisticato, sono stati effettuatu alcuni studi per determinare il miglior algoritmo per gestire la sincronizzazione tra le diverse point cloud [42]. Tale studio ha fatto un confronto in base a dei criteri omogenei di performance con una scala di valori. L'algoritmo migliore risultante da questi confronti ha risolto la problematica con tempi dell'ordine di 30 ms.

4.1.3 Integrazione volumetrica

In base alle tecniche precedentemente enunciate, avendo nota la posizione di Microsoft Kinect rispetto al sistema di riferimento globale, ogni misurazione di distanza, può essere convertita dal sistema relativo al sistema globale per mezzo della trasformazione rigida $T_i = [R, \vec{t_i}]$. Tale trasformazione è computabile secondo le regole di convergenza insite nell'algoritmo ICP (Sezione 4.1.2.2).

Per ricostruire le scene tridimensionali, Microsoft Kinect Fusion, invece di utilizzare una fusione delle point cloud o una mesh, sfrutta una rappresentazione dello spazio quantizzata [8, 18, 27]. Tale rappresentazione è determinata attraverso una capacità finita dipendente dalla risoluzione. Per capacità finita si intende un volume finito di spazio caratterizzato da dimensioni note suddiviso uniformemente attraverso una voxel grid tridimensionale. È possibile intuire fisicamente la struttura dei voxel immaginando delle memorie di massa di personal computer. Tali memorie sono dei contenitori indirizzabili attraverso degli address.

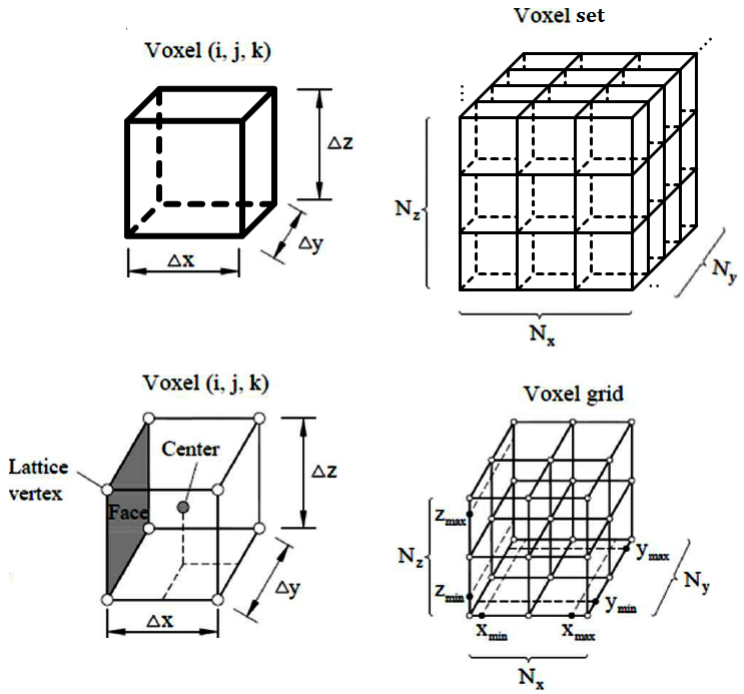

Figura 4.7: Rappresentazione dei voxel [55].

Per voxel, infatti, si intende uno spazio contenitore tridimensionale all'interno del quale è possibile salvare dei valori scalari corrispondenti a specifiche funzioni. Tale concetto rappresenta un'estensione alla terza dimensione del concetto di pixel per una specifica bitmap.

Tramite un singolo voxel è possibile rappresentare le seguenti forme – Figura 4.7:

- Un blocco solido regolare
- Una rappresentazione poligonale spaziale

Come conseguenza a questa rappresentazione il volume è caratterizzato da una determinata risoluzione esprimibile attraverso un fattore di scala. A seguito di tale valore, il volume dei singoli voxel esprimerà una specifica misura in metri cubi.

Il passo successivo alla predisposizione del volume è l'integrazione dei vertici traslati dal sistema di riferimento della telecamera al sistema di riferimento globale attraverso una specifica funzione distanza – Figura 4.8. Tale funzione SDF (Signed Distance Function) definisce una superficie implicita $D(x, y, z) = 0$ nella quale, viene memorizzato un valore convenzionale positivo

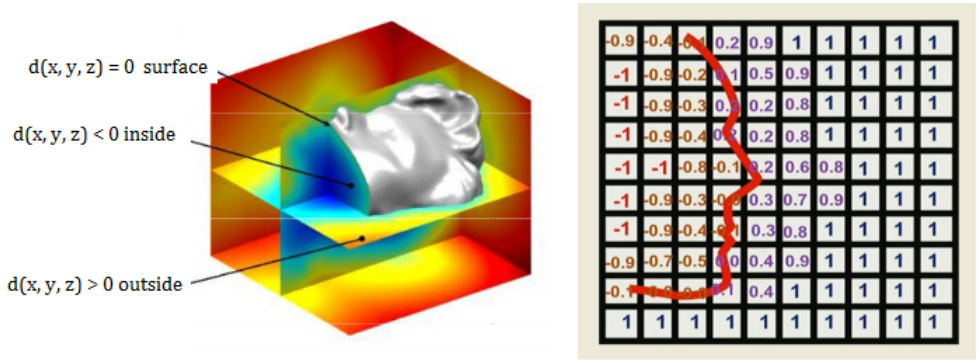

Figura 4.8: Troncated Signed Distance Function - TSDF [50].

di fronte alla superficie, negativo dietro e nullo in corrispondenza dell'interfaccia dove avviene il cambio di segno [8, 33]. In realtà, la vicinanza o meno dal profilo è caratterizzata da un troncamento della funzione distanza (TSDF - Troncated SDF), ovvero, superata una certa soglia il valore di riferimento è troncato a un intero. Tale funzionalità rappresenta in realtà una ottimizzazione perché in questo modo la rappresentazione spaziale del solido acquisito ha una scala di valori più ampia e viene permessa la caratterizzazione della funzione distanza con un conseguente aumento della precisione.

L'algoritmo così definito in realtà presenta una ulteriore caratteristica dovuta ai continui aggiornamenti dei valori codificati nei voxel. In particolare, a tempo di esecuzione, durante l'aggiornamento viene fatta una media ovvero se ad esempio un singolo voxel all'istante t_0 codifica una determinata funzione distanza $D(x, y, z)_0 = 0,5$ e successivamente all'acquisizione computata al tempo t_1 il valore misurato è codificato pari $D'(x, y, z) = 0,3$ la funzione distanza risultante sarà pari alla media aritmetica delle acquisizioni precedenti $D_1(x, y, z) = 0,4$.

In realtà Microsoft Kinect Fusion associa ad ogni voxel oltre alla specifica funzione distanza $D(x, y, z)$ uno specifico peso $W(x, y, z)$ dipendente dalla tecnologia di acquisizione il quale permette di rappresentare in modo più preciso le variazioni di incertezza lungo la superficie definita implicitamente. In questo modo l'aggiornamento di ogni singolo voxel può avvenire secondo la scrittura:

$$D_{i+1}(x, y, z) = \frac{W_i(x, y, z)D_i(x, y, z) + w_{i+1}(x, y, z)d_{i+1}(x, y, z)}{W_i(x, y, z) + w_{i+1}(x, y, z)}, \tag{4.45}$$

$$W_{i+1}(x, y, z) = W_i(x, y, z) + w_{i+1}(x, y, z). \tag{4.46}$$

Per caratterizzare in modo più strutturato questa sezione si rende necessaria un'ultima precisazione. Precedentemente avevamo espresso il concetto

secondo il quale ad ogni singolo pixel era associabile uno specifico thread. In questo caso, dato che nella rappresentazione tridimensionale i voxel sono in numero ben superiore rispetto ai pixel, i thread utilizzati per l'integrazione volumetrica sono specifici a una singola finestra del volume totale la quale scorre costantemente lungo un asse.

4.1.4 Raycasting

Riepilogando lo stato dell'arte, è stato possibile esaminare in primo luogo le regole di scansione della camera, per poi passare all'algoritmo ICP che è il principale incaricato dell'allineamento delle nuvole di punti. Successivamente, è stato possibile entrare nello specifico della risoluzione tridimensionale ampliando il concetto sui voxel ed il loro utilizzo in questo contesto. Di qui in avanti, ci addentreremo nella funzionalità di rendering che rappresenta l'anello finale della catena del sistema KinectFusion [18, 27, 35].

Nel campo della scansione, per poter effettuare l'attività di rendering dell'isosuperficie, generalmente vengono utilizzate delle metodologie di costruzione poliedrica e successivamente vengono renderizzate attraverso hardware dedicati per la gestione grafica professionale. In particolare, il risultato che osserviamo sullo schermo viene prodotto attraverso un hardware specifico, noto come SGI Infinite Reality che prende in input la superficie poligonale implicita costruita attraverso l'algoritmo Marching Cubes. Questa tecnica è molto dispendiosa dal punto di vista delle performance in quanto si basa su concetti geometrico-matematici complessi. In effetti, in presenza di superfici di piccole dimensioni il sistema è perfettamente equilibrato, tuttavia, nel caso in cui le dimensioni superino determinate soglie i tempi di elaborazione aumentano in maniera esponenziale. In altri termini, potremmo dire che l'interattività dell'applicazione è inversamente proporzionale all'ampiezza della superficie scansionata.

Per ovviare a questo inconveniente, KinectFusion ha iniettato nelle operatività di ottimizzazione un algoritmo denominato Ray Tracing che si occupa di snellire la procedura di rendering [35]. In questo modo, nel caso di superfici più grandi viene mantenuta l'interattività caratteristica dello strumento in quanto non si rende necessaria la memorizzazione tramite mesh poligonale effettuata attraverso l'algoritmo di Marching cubes. In particolare, la superficie è renderizzata direttamente tramite l'idea di un raggio virtuale proiettato sulla superficie definita implicitamente – Figura 4.9.

Come precedentemente anticipato, nel caso di rappresentazioni bidimensionali, ad ogni pixel viene associato uno specifico thread. Nello specifico, per restituire la figura finale a video, è possibile immaginare una sorta di irraggiamento la cui fonte luminosa ha inizio dietro lo schermo e va a col-

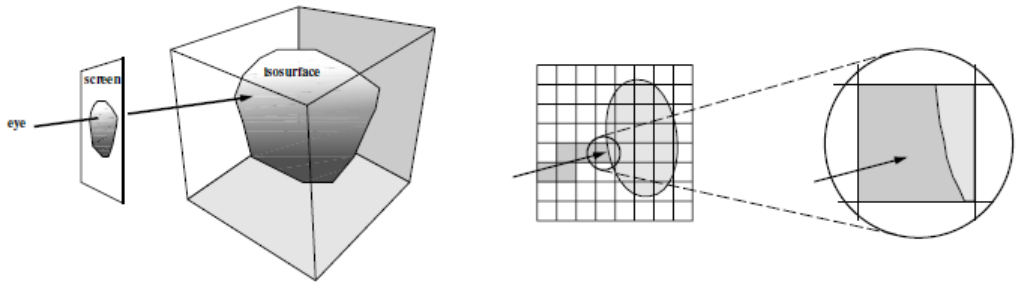

Figura 4.9: Estrazione dell'isosuperficie [35].

pire l'isosuperficie. Come conseguenza di questo modello, ad ogni thread è associato un singolo raggio virtuale fittizio caratterizzato da una posizione e da una direzione. Il passo successivo a questa rappresentazione, nel quale il raggio attraversa il volume definito dai voxel, è costituito dall'analisi della superficie implicita osservando il cambio di segno. In particolare, il punto di intersezione della superficie verrà interpolato in base alle relazioni esistenti tra i punti vicini al cambio di segno.

Fin qui, abbiamo esaminato il processo che porta al rinvenimento della posizione della superficie rispetto a quanto proiettato fittiziamente attraverso lo schermo. Dai passaggi precedenti è possibile ricordare che oltre alle posizioni dei vertici vengono memorizzate anche le direzioni delle normali agli stessi. Di conseguenza, quando il raggio colpisce l'isosuperficie, l'effetto conseguente è una riflessione misurabile con caratteristiche e proprietà. Tali misurazioni consentono un'analisi della deviazione del raggio attraverso un confronto con le normali ai vertici unitamente alle posizioni degli stessi e alla direzione iniziale del raggio. Infine l'immagine viene renderizzata, considerando anche le zone d'ombra, secondo le proprietà di riflettanza computando l'intensità luminosa del raggio riflesso. È importante considerare che uno dei vantaggi di questa metodica è proprio la determinazione delle zone d'ombra come conseguenza dell'irraggiamento sull'isosuperficie.

4.2 Altri sistemi di ricostruzione di scene

Oltre a Microsoft Kinect Fusion, esistono altre architetture tecnologiche che hanno degli scopi analoghi. In particolare si tratta di ReconstructMe distribuito da Profactor e Skanect distribuito da Manctl. Queste ulteriori piattaforme essendo basate su un brevetto commerciale sono di tipo proprietario e non sono state rilasciate documentazioni tecniche inerenti alla costruzione degli algoritmi. Di conseguenza non si conoscono i punti di forza e ci si limita all'utilizzo attraverso interfaccia visuale. Nel tempo sono state comunque costruite alcune librerie OpenSource con funzioni di governo per invocazione dei servizi incapsulati nel prodotto software. Esistono inoltre alcune librerie OpenSource anche per Kinect Fusion, ad esempio Kinfu, le quali hanno una suite ridotta di funzionalità rispetto a quella estesa ma hanno il pregio di essere gratuite.

Capitolo 5

Note tecniche - Software e Framework

Nei passaggi precedenti abbiamo avuto modo di esaminare la parte teorica relativamente alla 3D reconstruction approfondendo alcune peculiarità implementative. Questa esplorazione ha volutamente preso in esame la sfera tecnica e matematica con aspetti di natura topologica al fine di sviscerare il più possibile i contenuti necessari su cui basare la tecnica implementativa.

Di qui in avanti esamineremo i tool di sviluppo e i software opensource strumentali alla realizzazione simbiotica all'architettura di riferimento. Lasceremo uno spazio per un benchmark tra le diverse librerie utilizzate per realizzazioni afferenti alla Computer Vision, Computer Graphic e 3D reconstruction.

Gli indicatori di confronto tra le diverse tecnologie che presenteremo sono i seguenti:

- Punti di forza dell'architettura;
- Diffusione e frequenza di utilizzo nei campi di riferimento;
- Linguaggio e piattaforma di utilizzo;
- Note bibliografiche.

5.1 Architetture di base

Scopo della sezione è una disamina di tipo tecnico sulle librerie più frequentemente utilizzate nell'ambito di architetture dedicate alla robotica, Computer Vision e Computer Graphics. Non verranno approfondite tematiche dedicate allo sviluppo in senso pratico, tuttavia, presenteremo alcuni confronti

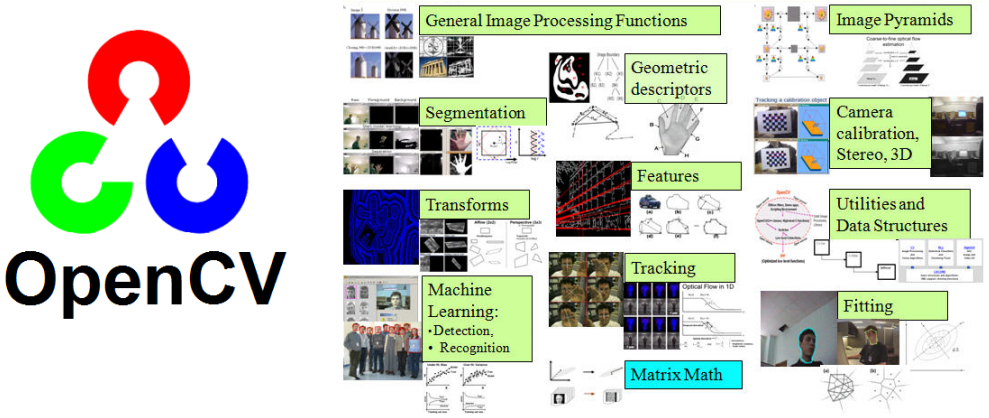

Figura 5.1: OpenCV - Open Source Computer Vision Library [5, 12].

e suggerimenti dedicati all'utilizzo del singolo framework in relazione alle diverse tipologie di realizzazione.

Le librerie sulle quali si basano la maggior parte degli sviluppi applicativi realizzate con filosofia opensource sono: AForge.NET, OpenCV, OpenGL, PCL, ROS e VLFeat. Ognuna di queste tecnologie presenta alcuni punti di forza, per ciascuna verranno forniti alcuni indicatori di riferimento e in alcuni casi presenteremo specifici approfondimenti circostanziati da riferimenti bibliografici per permettere ulteriori indicazioni di utilizzo.

5.1.1 OpenCV - Open Source Computer Vision Library

Inizialmente sviluppata e lanciata da Intel nel 1999 è ora uno dei principali strumenti multipiattaforma e opensource con licenza BSD, di conseguenza il software realizzato è completamente libero da ogni onere di natura commerciale – Figura 5.1. Questa architettura tecnica è stata scritta in C++ ma supporta anche applicazioni specifiche scritte in Java, C o Python.

Tale tecnologia rappresenta uno dei punti cardine nell'ambito dello sviluppo applicativo inerente alla Computer Vision, Augmented Reality e in particolare alla 3D Reconstruction. Offre alcune specifiche componenti applicative dedicate alla Structure from Motion, alla Stereo Vision nell'ambito dei sistemi SLAM e altre applicazioni frequentemente invocate tra le problematiche della realtà aumentata. In aggiunta permette alcune feature particolari come Object Identification, Motion Tracking e Machine Learning.

Per informazioni tecniche specifiche si rimanda a: G. Bradski. «The OpenCV Library». In: *Dr. Dobb's Journal of Software Tools* (2000). URL: `https://github.com/itseez/opencv`

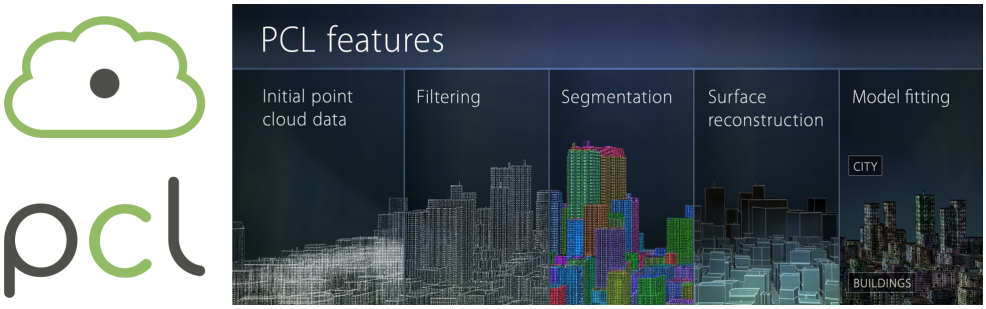

Figura 5.2: PCL - Point Cloud Library [43].

5.1.2 PCL - Point Cloud Library

Recente tecnologia diffusa nel mercato nel 2010 attraverso Willow Garage, laboratorio di ricerca dedicato alla robotica. Questa tecnologia è molto apprezzata per risolvere problematiche dedicate alla Robotic Perception in quanto offre specifiche e sofisticate componenti applicative necessarie alla gestione degli algoritmi per la computazione delle Point Cloud – Figura 5.2. Nello specifico, è una libreria che offre una suite completa di funzionalità molto utili nell'ambito della 3D Reconstruction e problematiche affini come la fusione e registrazione delle Point Cloud, il filtraggio del rumore, la segmentazione e la rimozione degli outliers.

Questo framework applicativo scritto in C++ con modalità opensource viene offerto con licenza BSD. È molto utilizzato all'interno di aziende multinazionali che offrono prodotti software dedicati alla Computer Vision essendo multipiattaforma e supportando compilazioni per ambienti Linux, MacOS, Windows, Android/iOS.

Per informazioni tecniche specifiche si rimanda a: Radu Bogdan Rusu e Steve Cousins. «3D is here: Point Cloud Library (PCL)». in: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, mag. 2011. URL: `https://github.com/PointCloudLibrary/pcl`

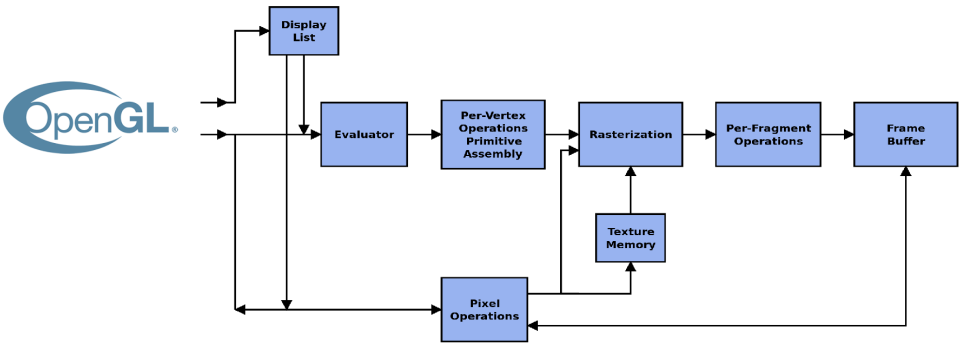

Figura 5.3: OpenGL - Open Graphics Library [29].

5.1.3 OpenGL - Open Graphics Library

Nata nei laboratori Silicon Graphics, questa tecnologia ha avuto i suoi esordi nel 1992 e nello specifico è una API molto diffusa per applicazioni di Computer Graphics 2D/3D e Augmented Reality – Figura 5.3. In particolare è una delle più utilizzate nei campi del gaming e della simulazione tridimensionale. Tale architettura opensource è multipiattaforma e particolarmente ottimizzata per ambienti Microsoft, Linux, Unix, MacOS e Playstation. Il suo utilizzo specifico è dedicato alla grafica 3D e a una gestione degli acceleratori nonché alla rasterizzazione ovvero alle problematiche di rendering più in generale.

Essendo una tecnologia molto consolidata nel tempo sono stati scritti diversi connettori per permettere l'utilizzo da più fonti con piattaforme e linguaggi differenti, di conseguenza la sua invocazione è possibile attraverso C++ ma anche con C, Java, Python, C#, Perl e Visual Basic.

Per informazioni tecniche specifiche si rimanda a: OpenGL.org. *OpenGL*. 2013. URL: `http://www.opengl.org/wiki_132/index.php?title=OpenGL_Reference&oldid=9108`

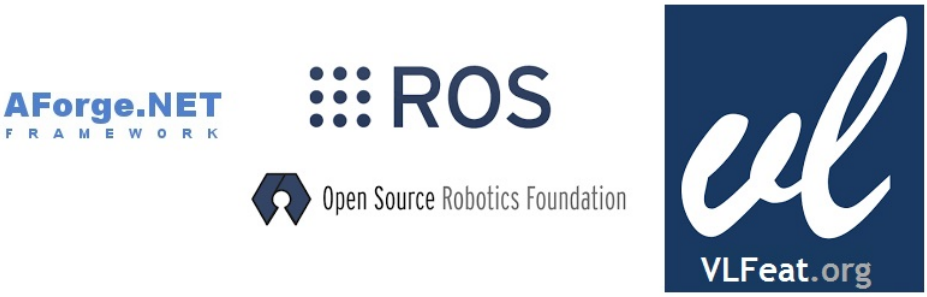

Figura 5.4: Altre librerie general purpose - AForge.Net, ROS, VLFeat [3, 39, 54].

5.1.4 Altre librerie general purpose

Oltre alle più diffuse tecnologie sopra riportate ne esistono numerose altre comunque apprezzate e diffuse sempre dedicate alla Computer Vision, 3D Reconstruction e a specifiche applicazioni inerenti al mondo della robotica – Figura 5.4. Ognuna di queste tecnologie presenta dei punti di forza utilizzati su specifiche applicazioni ma essendo meno diffuse si riporta un semplice elenco con un suggerimento bibliografico – Tabella 5.1.

Tabella 5.1: Altre librerie general purpose.

Tecnologia	Linguaggio	Licenza	Riferimento
AForge.NET	C#	LGPLv3, GPLv3	`http://aforgenet.com/`
ROS	C++, Python	BSD	`https://github.com/ros`
VLFeat	C, Matlab	BSD	`http://www.vlfeat.org/`

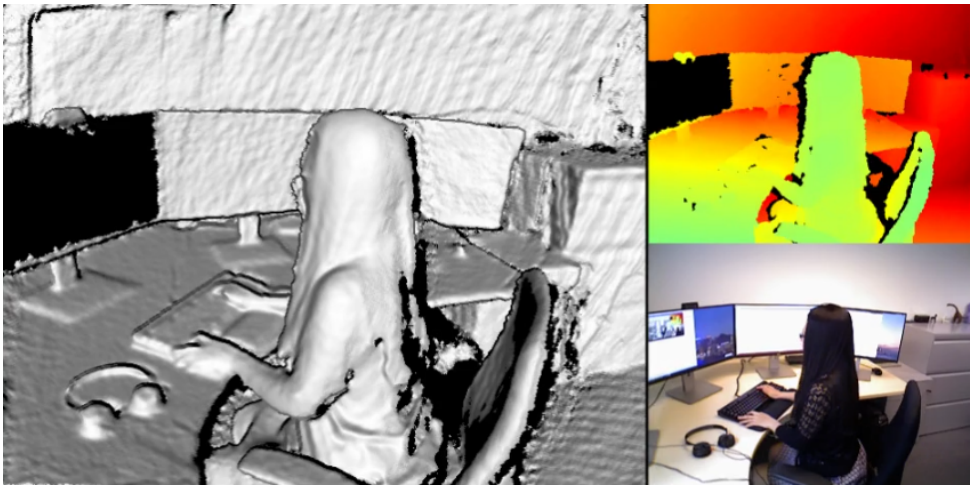

Figura 5.5: Infinitam by Oxford Isis Innovation Accademy [38].

5.2 Framework dedicati alla 3D Reconstruction

Scopo della sezione è una disamina di librerie specializzate sulle problematiche della 3D Reconstruction. Si tratta di particolari componenti applicativi nati dalla collaborazione di team di sviluppo facenti parte di progetti internazionali opensource. In particolare, tali progetti hanno permesso di consolidare esperienze applicative sulla problematica della Structure from Motion, Multiple View Stereovision e Simultaneous Localization and Mapping.

Tali progetti sono immediatamente utilizzabili e continuamente aggiornati attraverso i relativi team che governano l'evoluzione del software. Considerando che esiste una vasta letteratura in rete, riteniamo più utile concentrarsi su una sola delle architetture tecniche più innovative dedicando un elenco esaustivo sulle altre tecnologie.

5.2.1 InfiniTAM

Uno dei framework più innovativi, rilasciato recentemente nel 2015 è InfiniTAM licenziato attraverso la Oxford Isis Innovation Accademy – Figura 5.5. Si tratta di un framework multipiattaforma dedicato a problematiche real time con specifici riferimenti a tematiche di fusione con gestione di volumi densi e volumi sparsi delle Point Cloud. Essendo di recente costituzione, è stato appositamente progettato per poter essere invocabile sia attraverso architetture opensource ma anche da software vendor proprietari. In effetti,

benché scritto in C++ è utilizzabile sia da architetture Linux ma anche da architetture Microsoft, MacOS e Android/iOS.

Per informazioni tecniche specifiche si rimanda a: V. A. Prisacariu et al. «A Framework for the Volumetric Integration of Depth Images». In: *ArXiv e-prints* (2014). URL: `https://github.com/victorprad/InfiniTAM`

5.2.2 Altri framework dedicati

Sempre nell'ambito di progetti opensource, sono nate delle altre tecnologie talvolta come fork di tecnologie precedenti ognuna delle quali con specifiche caratteristiche e peculiarità.

Di seguito, alcune tabelle di riferimento categorizzate in funzione dell'area tematica risolta. È possibile trovare ampia letteratura in rete per quanto riguarda la tematica inerente la Structure from Motion, di seguito i principali indicatori – Tabella 5.2:

Tabella 5.2: Altri framework dedicati - Structure from Motion.

Structure from Motion - SfM			
Tecnologia	Linguaggio	Licenza	Riferimento
Bundler	C++	GNU	`https://github.com/snavely/bundler_sfm`
Colmap	C++	GNU	`https://github.com/colmap/colmap`
MAP-Tk	C++	BSD	`https://github.com/Kitware/maptk`
MICMAC	C++	CeCILL-B	`http://logiciels.ign.fr/?Micmac`
MVE	C++	BSD, GPL	`https://github.com/simonfuhrmann/mve`
OpenMVG	C++	MPL2	`https://github.com/openMVG/openMVG`
OpenSfM	Python	BSD	`https://github.com/mapillary/OpenSfM/`
TheiaSfM	C++	New BSD	`https://github.com/sweeneychris/TheiaSfM`

È possibile trovare ampia letteratura in rete per quanto riguarda la tematica inerente la Multiple View Stereovision, di seguito i principali indicatori – Tabella 5.3:

Tabella 5.3: Altri framework dedicati - Multiple View Stereovision.

Multiple View Stereovision - MVS			
Tecnologia	Linguaggio	Licenza	Riferimento
Colmap	C++	GNU	`https://github.com/colmap/colmap`
GPUIma	C++	GNU	`https://github.com/kysucix`
HPMVS	C++	GNU	`https://github.com/alexlocher/hpmvs`
MICMAC	C++	CeCILL-B	`http://logiciels.ign.fr/?Micmac`
MVE	C++	BSD, GPL	`https://github.com/simonfuhrmann/mve`
OpenMVS	C++	AGPL3	`https://github.com/cdcseacave/openMVS/`
PMVS	C++	GNU	`https://github.com/pmoulon/CMVS-PMVS`

È possibile trovare ampia letteratura in rete per quanto riguarda la tematica inerente i sistemi SLAM, di seguito i principali indicatori – Tabella 5.4:

Tabella 5.4: Altri framework dedicati - Simultaneous Localization And Mapping.

Simultaneous Localization And Mapping - SLAM			
Tecnologia	Linguaggio	Licenza	Riferimento
DTSLAM	C++	BSD	`https://github.com/plumonito/dtslam`
LSD-SLAM	C++	GNU	`https://github.com/tum-vision/lsd_slam/`
ORB-SLAM	C++	GPLv3	`https://github.com/raulmur/ORB_SLAM2`
REBVO	C++	GNU	`https://github.com/JuanTarrio/rebvo`
SVO	C++	GNU	`https://github.com/uzh-rpg/rpg_svo`

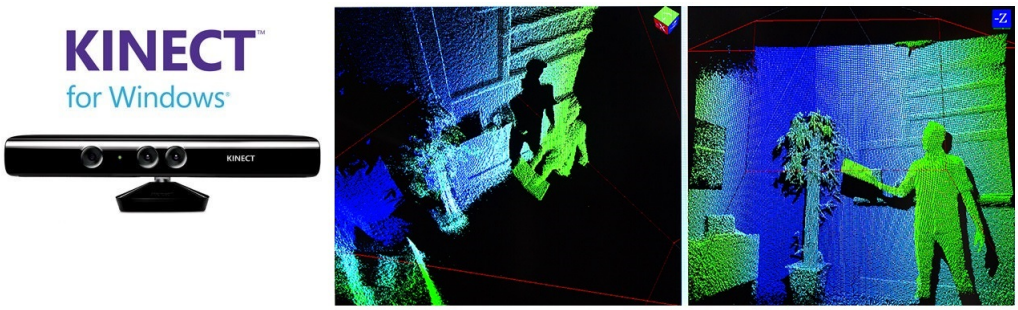

Figura 5.6: Kinect for Windows SDK [24].

5.3 Framework specifici Kinect compliance

Microsoft Kinect si è imposto con un suo software su licenza all'interno di un mercato di riferimento tipicamente opensource. Di conseguenza, sono pochi i framework applicativi disponibili per questo dispositivo, tuttavia, consentono una grande duttilità e semplicità di utilizzo. In questo scenario, si sono affermate principalmente tre architetture tecniche di cui due gratuite ovvero OpenKinect, OpenNI e una acquistabile su licenza del fornitore di tecnologia ovvero Kinect for Windows SDK. Esistono inoltre alcuni tool gratuiti specifici realizzati attraverso progetti opensource che esamineremo specificatamente in sezioni dedicate.

5.3.1 Kinect for Windows SDK

Questo SDK è stato sviluppato da uno specifico gruppo di progetto interno a Microsoft ed è stato lanciato nel 2011 e costantemente aggiornato – Figura 5.6. Scritto in C# non permette un utilizzo multipiattaforma e consente invece un richiamo esclusivamente da architettura Microsoft.Net con C# e/o C++.

Questo toolkit di sviluppo include sia i driver per il sensore che permettono di interagire con il dispositivo Microsoft Kinect v1/v2 sia le API verso i servizi. Benché sia possibile il download la tipologia di licenza commerciale ne limita fortemente la diffusione, tuttavia è uno strumento molto intuitivo e gradito agli sviluppatori, in particolare offre servizi di depth image processing, skeleton tracking, human gesture recognition, speech recognition, face tracking e molti altri.

Per informazioni tecniche specifiche si rimanda a : Microsoft. *Kinect for Windows SDK*. 2011. URL: `https://developer.microsoft.com/it-it/windows/kinect/develop`

Figura 5.7: Libfreenect by OpenKinect [30].

5.3.2 Libfreenect

Parallelamente al software licenziato ad uso commerciale è nata la community OpenKinect specializzata per lo sviluppo di un analogo software con utilizzo opensource denominato Libfreenect – Figura 5.7. Tale community ha effettuato un primo rilascio verso la fine del 2010 con licenza GPL2 e/o Apache 2.0. Si tratta di un software multipiattaforma sia per ambienti Microsoft che per ambienti Linux e Mac. Nel tempo sono stati sviluppati alcuni wrappers per rivestire di uno strato di software utile all'invocazione da Python, C, C++, C#, Java, Actionscript e Ruby.

Questo framework permette di gestire immagini sia in formato RGB sia in formato 3D con l'utilizzo di accelerometri e audio. Naturalmente offre una suite di servizi ridotta partendo comunque da funzioni di 3D Reconstruction e Point Cloud Fusion.

Per informazioni tecniche specifiche si rimanda a : OpenKinect. *Libfreenect*. 2010. URL: `https://openkinect.org/wiki/Main_Page`

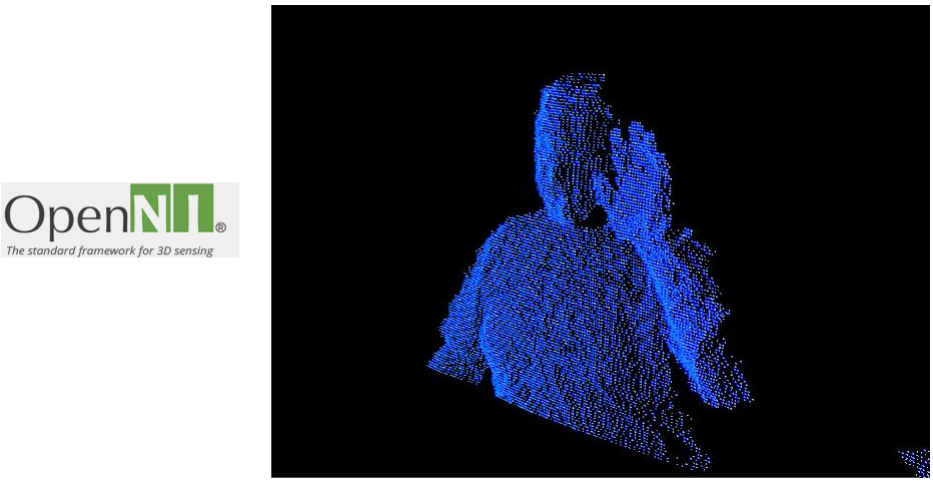

Figura 5.8: OpenNI - Open Natural Interaction [31].

5.3.3 OpenNI - Open Natural Interaction

OpenNI nasce dal progetto opensource omonimo nel 2010 ed è stato acquisito da Apple nel 2013 – Figura 5.8. Attualmente, è l'SDK più utilizzato per le operatività di 3D sensing e possiede il suo punto di eccellenza nella gestione della Natural user interface. Realizzato con modalità opensource e multipiattaforma consente la definizione di API specifiche per rendere indipendente lo sviluppo dal linguaggio originale. Attualmente esistono diversi wrappers e in particolare quelli più frequentemente utilizzati sono quelli per Python, C++, Java, C# in architettura .NET. Tale framework è frequentemente utilizzato per applicazioni dove necessitano funzionalità per la voice recognition, il motion tracking e la 3D reconstruction.

Per informazioni tecniche specifiche si rimanda a : OpenNI. *OpenNI*. 2010. URL: `http://openni.ru/index.html`

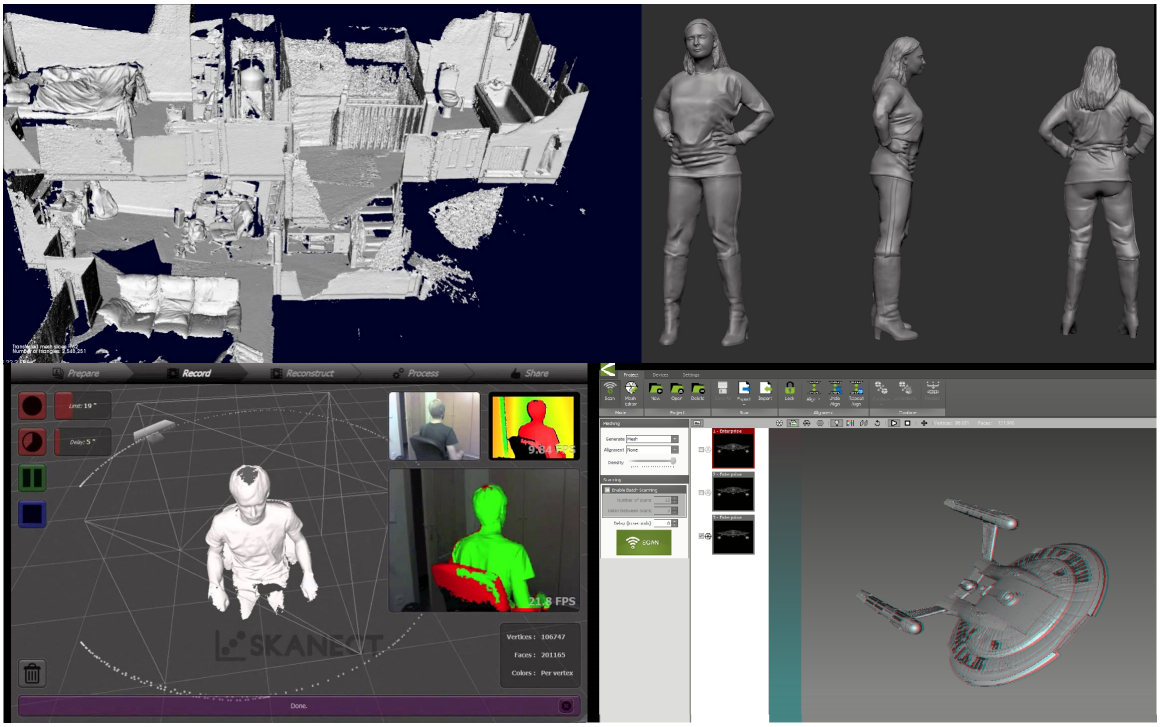

Figura 5.9: Kintinuous, KScan 3D, ReconstructME, Scanect [20, 21, 40, 45].

5.3.4 Altri tool e software dedicati

Nelle sezioni precedenti sono state rapidamente esaminati i framework più ricorrenti e frequentemente utilizzati. Di seguito un elenco riepilogativo di ulteriori tool di sviluppo inerenti architetture Microsoft Kinect – Figura 5.9. Per maggiori approfondimenti nella Tabella 5.5 sono riportati i relativi punti di contatto.

Tabella 5.5: Altri tool e software dedicati.

Tecnologia	Riferimento
Kinect Fusion	`https://msdn.microsoft.com/en-us/library/dn188670.aspx`
Kintinuous	`https://github.com/mp3guy/Kintinuous`
KScan 3D	`http://www.kscan3d.com/`
LiveScan 3D	`https://github.com/MarekKowalski/LiveScan3D`
ReconstructME	`http://reconstructme.net/`
Scanect	`http://skanect.occipital.com/`

Appendice A

Minimizzazione della distanza

La seguente appendice, ha lo scopo di introdurre alcuni concetti analitici inerenti la minimizzazione della distanza [15]. Per una trattazione più estesa è consigliabile l'approfondimento personale attraverso consultazione di generici testi di Algebra lineare e Geometria.

Dati due punti $\vec{r_1} = (x_1, y_1, z_1)$ e $\vec{r_2} = (x_2, y_2, z_2)$ la distanza euclidea $d(\vec{r_1}, \vec{r_2})$ è definita da:

$$d(\vec{r_1}, \vec{r_2}) = \|\vec{r_1} - \vec{r_2}\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \tag{A.1}$$

Ipotizziamo A un insieme di punti $\vec{a_i}$:

$$A = \{\vec{a_i}\}, \qquad i = 1, ..., N_a. \tag{A.2}$$

La distanza tra il punto $\vec{p}$ e l'insieme di punti A è definita come:

$$d(\vec{p}, A) = min\{d(\vec{p}, \vec{a_i}), i \in [1, ..., N_a]\}. \tag{A.3}$$

Di conseguenza, il punto $\vec{a_j}$ appartenente ad A più prossimo a $\vec{p}$ soddisfa la relazione:

$$d(\vec{p}, \vec{a_j}) = d(\vec{p}, A). \tag{A.4}$$

Dati due punti $\vec{r_1}$ e $\vec{r_2}$, sia l il segmento che li congiunge. La distanza tra il generico punto $\vec{p}$ ed il segmento l è definita da:

$$d(\vec{p}, l) = min\{\|u\vec{r_1} + v\vec{r_2} - \vec{p}\|, u + v = 1\}, \qquad u \in [0, 1], v \in [0, 1]. \tag{A.5}$$

Ipotizziamo L un insieme di segmenti l_i:

$$L = \{l_i\}, \qquad i = 1, ..., N_l. \tag{A.6}$$

La distanza tra il punto $\vec{p}$ e l'insieme di segmenti L è definita come:

$$d(\vec{p}, L) = min\{d(\vec{p}, l_i), i \in [1, ..., N_l]\}. \tag{A.7}$$

Di conseguenza, il punto $\vec{y_j}$ appartenente ad L più prossimo a $\vec{p}$ soddisfa la relazione:

$$d(\vec{p}, \vec{y_j}) = d(\vec{p}, L). \tag{A.8}$$

Dati tre punti $\vec{r_1}$, $\vec{r_2}$ e $\vec{r_3}$ sia t il triangolo da essi definito. La distanza tra il generico punto $\vec{p}$ ed il triangolo t è definita da:

$$d(\vec{p}, t) = min\big\{\|u\vec{r_1} + v\vec{r_2} + w\vec{r_3} - \vec{p}\|, u + v + w = 1\big\}, \\ u \in [0,1], v \in [0,1], w \in [0,1]. \tag{A.9}$$

Ipotizziamo T un insieme di triangoli t_i:

$$T = \big\{t_i\big\}, \qquad i = 1, ..., N_t. \tag{A.10}$$

La distanza tra il punto $\vec{p}$ e l'insieme di triangoli T è definita come:

$$d(\vec{p}, T) = min\big\{d(\vec{p}, t_i), i \in [1, ..., N_t]\big\}. \tag{A.11}$$

Di conseguenza, il punto $\vec{y_j}$ appartenente a T più prossimo a $\vec{p}$ soddisfa la relazione:

$$d(\vec{p}, \vec{y_j}) = d(\vec{p}, T). \tag{A.12}$$

A.1 Distanza tra punti ed entità parametriche

Definiamo $E = \vec{r}(\vec{u})$ entità parametrica, dove intendiamo per $\vec{u} = u \in \mathbb{R}^1$ una curva parametrica e per $\vec{u} = (u, v) \in \mathbb{R}^2$ una superficie parametrica. La distanza tra il generico punto $\vec{p}$ e la generica entità parametrica E è definita da:

$$d(\vec{p}, E) = min\big\{d\big(\vec{p}, \vec{r}(\vec{u})\big), \vec{r}(\vec{u}) \in E\big\}. \tag{A.13}$$

Ipotizziamo F un insieme di entità parametriche E_i:

$$F = \big\{E_i\big\}, \qquad i = 1, ..., N_e. \tag{A.14}$$

La distanza tra il punto $\vec{p}$ e l'insieme di entità parametriche F è definita come:

$$d(\vec{p}, F) = min\big\{d(\vec{p}, E_i), i \in [1, ..., N_e]\big\}. \tag{A.15}$$

Di conseguenza, il punto $\vec{y_j}$ appartenente ad F più prossimo a $\vec{p}$ soddisfa la relazione:

$$d(\vec{p}, \vec{y_j}) = d(\vec{p}, F). \tag{A.16}$$

In questo caso, i calcoli per computare la distanza sono considerevoli. Il metodo utilizzato per ovviare al problema è basato su un'approssimazione.

Per una curva parametrica $C = \{\vec{r}(u)\}$ è possibile computare una spezzata chiusa $L(C, \delta)$ in modo tale che l'approssimazione non devi mai dalla curva reale per più di una distanza δ. Associando ad ogni punto della spezzata il corrispondente argomento u della curva parametrica, è possibile ottenere una stima dell'argomento u_a del punto più prossimo alla spezzata.

Allo stesso modo, per una superficie parametrica $S = \{\vec{r}(\vec{u})\}$ è possibile computare una un insieme di triangoli $T(S, \delta)$ in modo tale che l'approssimazione non devi mai dalla superficie reale per più di una distanza δ. Associando ai vertici di ogni triangolo il corrispondente argomento $\vec{u}$ della superficie parametrica, è possibile ottenere una stima dell'argomento $\vec{u_a}$ del punto più prossimo all'insieme di triangoli.

Come risultato delle assunzioni precedenti è ora possibile ipotizzare che un valore iniziale $\vec{u_a}$ sia noto e che $\vec{r}(\vec{u_a})$ sia molto prossimo al punto più vicino dell'entità parametrica. Conseguentemente, il problema della distanza di un punto da un'entità parametrica si può risolvere utilizzando la minimizzazione di Newton. La funzione obiettivo da minimizzare è:

$$f(\vec{u}) = \|\vec{r}(\vec{u}) - \vec{p}\|^2. \tag{A.17}$$

Sia $\nabla = [\frac{d}{d\vec{u}}]^t$ l'operatore differenziale gradiente. Per una curva parametrica, il minimo della funzione si ottiene per $\nabla_f = [f_u]^t = 0$ e la matrice Hessiana degenera nel caso:

$$\nabla\nabla^t(f) = [f_{uu}]. \tag{A.18}$$

Per una superficie parametrica invece, il gradiente è definito da $\nabla_f = [f_u, f_v]^t$ e la matrice Hessiana è data da:

$$\nabla\nabla^t(f) = \begin{bmatrix} f_{uu} & f_{uv} \\ f_{uv} & f_{vv} \end{bmatrix}, \tag{A.19}$$

dove le derivate parziali della funzione obiettivo sono date da:

$$f_u(\vec{u}) = 2\vec{r}_u^t(\vec{u})\big(\vec{r}(\vec{u}) - \vec{p}\big), \tag{A.20}$$

$$f_v(\vec{u}) = 2\vec{r}_v^t(\vec{u})\big(\vec{r}(\vec{u}) - \vec{p}\big), \tag{A.21}$$

$$f_{uu}(\vec{u}) = 2\vec{r}_{uu}^t(\vec{u})\big(\vec{r}(\vec{u}) - \vec{p}\big) + 2\vec{r}_u^t(\vec{u})\vec{r}_u(\vec{u}), \tag{A.22}$$

$$f_{vv}(\vec{u}) = 2\vec{r}_{vv}^t(\vec{u})\big(\vec{r}(\vec{u}) - \vec{p}\big) + 2\vec{r}_v^t(\vec{u})\vec{r}_v(\vec{u}), \tag{A.23}$$

$$f_{uv}(\vec{u}) = 2\vec{r}_{uv}^t(\vec{u})\big(\vec{r}(\vec{u}) - \vec{p}\big) + 2\vec{r}_u^t(\vec{u})\vec{r}_v(\vec{u}). \tag{A.24}$$

La formula di aggiornamento di Newton per una generica entità parametrica è definita come:

$$\vec{u}_{k+1} = \vec{u}_k - [\nabla\nabla^t(f)(\vec{u}_k)]^{-1}\nabla f(\vec{u}_k), \tag{A.25}$$

dove $\vec{u}_0 = \vec{u}_a$. Utilizzando il metodo di Newton sopra descritto è dimostrabile che è possibile computare il punto più prossimo ottenendo la convergenza con al più cinque iterazioni.

A.2 Distanza tra punti ed entità implicite

Definiamo $I = \vec{g}(\vec{r}) = 0$ entità implicita. La distanza tra il generico punto $\vec{p}$ e la generica entità implicita I è definita da:

$$d(\vec{p}, I) = min\big\{d(\vec{p}, \vec{r}), \vec{g}(\vec{r}) = 0\big\} = min\big\{\|\vec{r} - \vec{p}\|, \vec{g}(\vec{r}) = 0\big\}. \tag{A.26}$$

Ipotizziamo J un insieme di entità implicite I_k:

$$J = \big\{I_k\big\}, \qquad k = 1, ..., N_i. \tag{A.27}$$

La distanza tra il punto $\vec{p}$ e l'insieme di entità implicite J è definita come:

$$d(\vec{p}, J) = min\big\{d(\vec{p}, I_k), k \in [1, ..., N_i]\big\}. \tag{A.28}$$

Di conseguenza, il punto $\vec{y_j}$ appartenente ad J più prossimo a $\vec{p}$ soddisfa la relazione:

$$d(\vec{p}, \vec{y_j}) = d(\vec{p}, J). \tag{A.29}$$

Anche in questo caso, i calcoli per computare la distanza sono considerevoli. Il metodo utilizzato per ovviare al problema è ancora una volta basato su un'approssimazione e computando la distanza dei punti dai segmenti o dai triangoli è di conseguenza possibile ottenere una stima del punto più prossimo $\vec{r}_a$ grazie al quale si riesce ad ottenere la distanza esatta.

Il problema della distanza di un punto da un'entità implicita si può risolvere minimizzando la seguente funzione obiettivo:

$$f(\vec{r}) = \|\vec{r} - \vec{p}\|^2, \qquad \vec{g}(\vec{r}) = 0. \tag{A.30}$$

Uno degli approcci utilizzabili è quello di risolvere il sistema di equazioni non lineari dei moltiplicatori di Lagrange numericamente:

$$\begin{cases} \nabla f(\vec{r}) + \vec{\lambda}^t \nabla \vec{g}(\vec{r}) = 0, \\ \vec{g}(\vec{r}) = 0. \end{cases} \tag{A.31}$$

Anche se elegante, il metodo sopra descritto ha pochi riscontri pratici essendo che in ambiente CAD sono pochissime le applicazioni che salvano curve e superfici come entità implicite. Un metodo utilizzabile quando $\vec{g}(\vec{r}_0)$ è prossimo a zero è il seguente:

$$\vec{r}_{k+1} = \vec{r}_k - \frac{\nabla \vec{g}(\vec{r}_k)\vec{g}(\vec{r}_k)}{\|\nabla \vec{g}(\vec{r}_k)\|^2}. \tag{A.32}$$

Essendo che questa approssimazione non è vera in generale, il risultato delle iterazioni non è utilizzabile nel caso in cui siano richieste distanze molto precise.

Bibliografia

[1] *3D Modelling*. 2015. URL: https://christevcreative.com/tag/3d-modelling/.

[2] *4D Facial Dynamics*. 2015. URL: http://dynface4d.isr.uc.pt/database.php.

[3] AForgeNET.com. *AForgeNET*. 2010. URL: http://aforgenet.com/.

[4] Paul J. Besl e Neil D. McKay. «A method for Registration of 3-D Shapes». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 14, Issue: 2, Feb 1992)*. A cura di IEEE. 2001, pp. 239–256.

[5] G. Bradski. «The OpenCV Library». In: *Dr. Dobb's Journal of Software Tools* (2000). URL: https://github.com/itseez/opencv.

[6] Antoni Buandes, Bartomeu Coll e Jean-Michel Morel. «A non-local algorithm for image denoising». In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* A cura di IEEE. 2005, pp. 1–6.

[7] Yang Chen e Gerard Medioni. «Object modelling by registration of multiple range images». In: *Image and Vision Computing* (1992), pp. 2724–2729.

[8] Brian Curless e Marc Levoy. «A volumetric method for building complex models from range images». In: *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. A cura di ACM. 1996, pp. 303–311.

[9] Kyis Essmaeel, Luigi Gallo, Ernesto Damiani et al. «Comparative evaluation of methods for filtering Kinect depth data». In: *Multimed Tools Appl* (2015), pp. 7335–7354.

[10] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA: The MIT Press, 1999, pp. 403–471.

[11] Basile Graf. «Quaternions And Dynamics». In: *Cornell University Library* (2007), pp. 1–22.

[12] Eric Gregori. *Introduction to Computer Vision Using OpenCV*. URL: http://www.embedded-vision.com/platinum-members/bdti/embedded-vision-training/documents/pages/introduction-computer-vision-using-op.

[13] Miles Hansard, Seungkyu Lee, Ouk Choi et al. «Time-of-Flight Cameras: Principles, Methods and Applications». In: *SpringerBriefs in Computer Science*. Springer, 2012, pp. 1–41.

[14] James Hays. *CS 143 Introduction to Computer Vision*. 2013. URL: http://cs.brown.edu/courses/cs143/.

[15] Berthold K. P. Horn. «Closed-form solution of absolute orientation using unit quaternions». In: *Journal of the Optical Society of America A* (1986), pp. 629–642.

[16] Benjamin Huhle, Timo Schairer, Philipp Jenke et al. «Fusion of range and color images for denoising and resolution enhancement with a non-local filter». In: *Computer Vision and Image Understanding*. 2015, pp. 1336–1345.

[17] *Image Histograms*. 2011. URL: http://betterphotographytutorials.com/2011/11/28/image-histograms-%E2%80%93-part-1/.

[18] Shahram Izadi, David Kim, Otmar Hilliges et al. «KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera». In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. A cura di ACM. 2011, pp. 1–10.

[19] *Kinect One and Kinect V2*. 2015. URL: https://kinectsen.wikispaces.com/Kinect+One+and+Kinect+V2.

[20] *Kintinuous, extended Kinect Fusion*. URL: https://github.com/mp3guy/Kintinuous.

[21] *KScan3D*. URL: http://www.kscan3d.com/.

[22] Tanwi Mallick, Partha Pratim Das e Arun Kumar Majundar. «Characterizations of Noise in Kinect Depth Images: A Review». In: *IEEE Sensors Journal* (2014), pp. 1731–1740.

[23] Stefan May, Stefan Fuchs et al. «Robust 3D-mapping with time-of-flight cameras». In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. A cura di IEEE. 2009, pp. 1–7.

[24] Microsoft. *Kinect for Windows SDK*. 2011. URL: https://developer.microsoft.com/it-it/windows/kinect/develop.

[25] Microsoft, cur. *Kinect Fusion*. 2015. URL: https://msdn.microsoft.com/en-us/library/dn188670.aspx.

[26] *Microsoft to discontinue Kinect for Windows V1*. URL: http://3rd-strike.com/microsoft-to-discontinue-kinect-for-windows-v1/.

[27] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges et al. «KinectFusion: Real-Time Dense Surface Mapping and Tracking». In: *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. A cura di IEEE. 2011, pp. 1–10.

[28] Chuong V. Nguyen, Shahram Izadi e David Lovell. «Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking». In: *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*. A cura di IEEE. 2012, pp. 524–530.

[29] OpenGL.org. *OpenGL*. 2013. URL: http://www.opengl.org/wiki_132/index.php?title=OpenGL_Reference&oldid=9108.

[30] OpenKinect. *Libfreenect*. 2010. URL: https://openkinect.org/wiki/Main_Page.

[31] OpenNI. *OpenNI*. 2010. URL: http://openni.ru/index.html.

[32] OSA, cur. *Structured-light 3D surface imaging: a tutorial*. 2011. URL: https://www.osapublishing.org/aop/fulltext.cfm?uri=aop-3-2-128&id=211561.

[33] Stanley Osher e Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*. Cambridge, MA: Springer, 2002.

[34] Diana Pagliari e Livio Pinto. «Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors». In: *MDPI Sensors* (2015), pp. 27579–27589.

[35] Steven Parker, Peter Shirley, Yarden Livnat et al. «Interactive ray tracing for isosurface rendering». In: *Proceedings of the conference on Visualization '98*. A cura di ACM. 1998, pp. 1–6.

[36] Emmanuel Prados e Olivier Faugeras. «Shape from shading: a well-posed problem?» In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. A cura di IEEE. 2005, pp. 1–9.

[37] *Princeton Vision and Robotics Group*. URL: https://vision.princeton.edu/.

[38] V. A. Prisacariu et al. «A Framework for the Volumetric Integration of Depth Images». In: *ArXiv e-prints* (2014). URL: https://github.com/victorprad/InfiniTAM.

[39] Morgan Quigley, Brian Gerkey, Ken Conley et al. «ROS: an open-source Robot Operating System». In: *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. Kobe, Japan, 2009. URL: https://github.com/ros.

[40] *ReconstructME*. URL: http://reconstructme.net/.

[41] *Research on 3D Object Modelling*. URL: http://carlos-hernandez.org/research.html.

[42] Szymon Rusinkiewicz e Marc Levoy. «Efficient Variants of the ICP Algorithm». In: *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. A cura di IEEE. 2001, pp. 145–152.

[43] Radu Bogdan Rusu e Steve Cousins. «3D is here: Point Cloud Library (PCL)». In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, mag. 2011. URL: https://github.com/PointCloudLibrary/pcl.

[44] Hamed Sarbolandi, Damien Lefloch e Andreas Kolb. «Kinect Range Sensing: Structured-Light versus Time-of-Flight Kinect». In: *Journal of Computer Vision and Image Understanding* (2015), pp. 1–58.

[45] *Scanect*. URL: http://skanect.occipital.com/.

[46] Timo Schairer, Benjamin Huhle, Philipp Jenke et al. «Parallel Non-Local Denoising of Depth Maps». In: *In International Workshop on Local and Non-Local Approximation in Image Processing (EUSIPCO Satellite Event)*. A cura di CiteSeer. 2008, pp. 1–7.

[47] Stackable, cur. *Producing 3D point clouds with a stereo camera in OpenCV*. 2014. URL: https://erget.wordpress.com/2014/04/27/producing-3d-point-clouds-with-a-stereo-camera-in-opencv/.

[48] *Structured Light vs Microsoft Kinect*. 2012. URL: http://www.hackengineer.com/structured-light-vs-microsoft-kinect/.

[49] *Surface Reconstruction Part 1*. URL: http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/03_Surface_Reconstruction.pdf.

[50] *Surface Reconstruction Part 2*. URL: http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/04_Surface_Reconstruction.pdf.

[51] *Time of Flight and Kinect Imaging*. 2011. URL: http://campar.in.tum.de/twiki/pub/Chair/TeachingSs11Kinect/2011-DSensors_LabCourse_Kinect.pdf.

[52] *Triangolazione di Delaunay*. URL: http://www2.mate.polimi.it/corsi/papers/7034d9c55ef1fa85ee8db41feb71.pdf.

[53] Emanuele Trucco e Alessandro Verri. *Introductory techniques for 3-D Computer Vision*. Upper Saddle River, NJ: Prentice Hall, 1998, pp. 1–13, 15–50, 51–66, 139–150, 177–191, 219–245.

[54] A. Vedaldi e B. Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. 2008. URL: http://www.vlfeat.org/.

[55] Richie Zirbes. *Scientific Visualization: Volume Surface Rendering*. URL: http://johnrichie.com/V2/richie/isosurface/volume.html.