POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in
Ingegneria dell'Automazione

POLITECNICO
MILANO 1863

ANALISI E SVILUPPO DI TECNICHE DI REGISTRAZIONE DI NUVOLE DI PUNTI PER LA RICOSTRUZIONE DI AMPIE SCENE

Relatore: Ing. Remo **Sala**
Correlatore: Ing. Silvio **Giancola**
Correlatore: Ing. Alessandro **Basso**

Tesi di Laurea di:
Daniele **Piron** Matr. 804798

Anno Accademico 2015-2016

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

Lo scopo del presente lavoro è lo sviluppo di una metodologia volta all'acquisizione e alla ricostruzione tridimensionale di ampie scene basandosi su tecniche Simultaneous Localization and Mapping. Nel caso di studio considerato, sia la localizzazione che la ricostruzione della mappa avviene elaborando informazioni ricavate da nuvole di punti acquisite tramite la telecamera a tempo di volo Kinect V2. Tale dispositivo viene movimentando liberamente (6 gradi di libertà) durante l'esplorazione della scena. Il software implementato permette l'acquisizione e l'allineamento di nuvole di punti consequenziali utilizzando un approccio basato sull'individuazione e descrizione di alcuni punti caratteristici della scena.

L'approccio standard prevede la rilevazione di questi punti dalle immagini acquisite e la descrizione tramite caratteristiche bidimensionali. Nel presente lavoro, invece, si è deciso di sperimentare una rilevazione di punti notevoli presi dalle nuvole e quindi descritti in termini tridimensionali. Questo approccio permette di sfruttare non solo informazioni colorimetriche ma anche informazioni geometriche e spaziali degli oggetti presenti nella scena.

Vengono presentati due casi applicativi: una ricostruzione di un sito archeologico e una ricostruzione di una scena del crimine. Per entrambi i casi, la ricostruzione ottenuta utilizzando il metodo proposto, viene confrontata con quella ricavata tramite l'utilizzo di diversi dispositivi e impiegando i software di ricostruzione proprietari. Col metodo sviluppato in questo lavoro è stato possibile ricostruire la mappa della scena con errori massimi nell'ordine di poche decine di centimetri. Tali errori sono compatibili con quelli ottenuti utilizzando un approccio standard a descrittori 2D, vale a dire il software RTAB-Map. L'approccio proposto tuttavia può essere utilizzato anche in situazioni di scarsa luminosità, introducendo così un importante vantaggio rispetto alla tecnica tradizionale.

Abstract

This thesis aims to develop a method for data acquisition and three-dimensional reconstruction of wide scene, based on Simultaneous Localization and Mapping techniques. For the present case study, both localization and map's reconstruction take place with the elaboration of information obtained from point clouds acquired with time of flight camera Kinect V2. This device is moved freely (6 degrees of freedom) during exploration of the scene. The implemented software allows the acquisition and alignment of consequential point clouds using an approach based on detection and description of some distinctive points of the scene.

A wide exploited approach performs the detection of these points of interest analysing the acquired images and it describes them in terms of bi-dimensional features. On the contrary, in the present work, a new approach is tested. It is based onto the detection and the description of the keypoints in terms of 3D features. This type of approach allows to use not only colorimetric information, but also the geometric and spatial information about the objects in the scene.

In the thesis, two applications of this approach are presented: a reconstruction of an archaeological site and a reconstruction of a crime scene. In both cases, the reconstruction obtained with the proposed method is compared to the ones obtained with different devices and using their own reconstruction software. the developed method, for both cases, led to a complete reconstruction of the wide scene with errors in the order of tens of centimeters. These errors are comparable with the errors obtained with standard approach of bi-dimensional descriptors, like the software RTAB-Map. Moreover, the proposed approach can be used also under poor illumination conditions, introducing, therefore, a great advantage compared to the traditional technique.

Introduzione

Il presente lavoro di tesi sperimentale è stato svolto all'interno del Vision Bricks Laboratory (VBLab) / 3D Vision [1], laboratorio di ricerca che opera nel Dipartimento di Meccanica del Politecnico di Milano e che si occupa principalmente di visione artificiale bi e tridimensionale e di misure senza contatto.

La tesi si pone come obiettivo lo sviluppo di metodi di ricostruzione tridimensionale di una scena attraverso tecniche di *Simultaneous Localization And Mapping* (SLAM) [2]. Tali tecniche, attraverso l'elaborazione delle misure ricavate durante l'esplorazione da diversi tipi di sensori (quali per esempio GPS od odometri), permettono di stimare probabilisticamente la traiettoria percorsa (*localization*), e allo stesso tempo di ricostruire una mappa dell'ambiente (*mapping*) grazie a informazioni ricavate da ulteriori sensori (quali per esempio telecamere o scanner a tempo di volo).

Nella maggior parte dei casi, si utilizza un sistema mobile dotato di ruote sensorizzate, dalle quali si ricavano informazioni per la stima dello spostamento tra una posizione e la successiva, e di telecamere o scanner 3D, dalle quali si estraggono informazioni per la ricostruzione dell'ambiente. Tale movimento presenta due soli gradi di libertà in quanto ci si sposta nel piano. Liberando il vincolo sulla rotazione attorno all'asse perpendicolare a tale piano è possibile aggiungere un grado di libertà.

Nel presente caso di studio, invece, si prendono in considerazione sei gradi di libertà (tre di traslazione e tre di rotazione). Infatti, la mappa viene realizzata a valle di un movimentazione libera di una telecamera a tempo di volo all'interno della scena. Inoltre, sia il processo di *localization* che di *mapping* vengono realizzati utilizzando le stesse informazioni: lo spostamento tra un'acquisizione e la successiva viene stimato tramite la comparazione di nuvole di punti consecutive e, mediante il corretto allineamento delle stesse, si costruisce la mappa. Particolare attenzione in questa tesi è stata indirizzata ai metodi di registrazione di coppie di nuvole di punti, ovvero alla stima della matrice di rototraslazione tra due pose del sensore a partire unicamente dalle nuvole stesse.

Le diverse tecniche di registrazione, descritte in letteratura, sono classificabili

principalmente in due categorie: quelle basate su Iterative Closest Point (ICP) [3], e quelle basate su *keypoints*. L'approccio ICP consiste in un algoritmo iterativo che permette di calcolare le corrispondenze tra tutti i punti di due nuvole diverse e minimizzare una funzione obiettivo, tipicamente descritta dalla somma dei quadrati delle distanze tra tali punti.

L'approccio a *keypoints*, invece, si basa sulla ricerca di punti caratteristici, tipicamente spigoli o bordi di oggetti, che descrivono localmente la scena. Si parla di keypoints 2D quando tali punti vengono estratti da un'immagine, basandosi quindi su variazioni di colore o di intensità luminosa tra pixel adiacenti; si parla di keypoints 3D quando tali punti si estraggono da nuvole, basandosi per esempio sulla variazione della direzione delle normali tra voxel adiacenti.

L'approccio a keypoints 2D è chiaramente obbligato quando si hanno a disposizione solo fotogrammi che rappresentano la scena, ma può anche essere sfruttato a supporto di tutti quei casi in cui si hanno sia fotogrammi che nuvole di punti, per esempio quando si acquisisce con una telecamera a tempo di volo come la Kinect. In questi casi, il procedimento di SLAM più utilizzato prevede l'identificazione dei punti caratteristici (2D keypoint *detection*) nelle immagini acquisite in posizioni diverse, la descrizione di questi punti (keypoint *description*) e la ricerca delle loro corrispondenze nelle diverse immagini. Tali corrispondenze vengono riportate sui relativi punti della nuvola 3D grazie alla conoscenza del modello matematico o dei dati di calibrazione del dispositivo. A partire da questi accoppiamenti è possibile stimare la rototraslazione tra le due nuvole.

Inoltre, l'approccio a keypoints 2D è preferibile, rispetto all'approccio keypoints 3D e a quello ICP, per il minor carico computazionale che permette anche applicazioni real time. In questo modo però non si tengono in considerazione le informazioni sulla forma tridimensionale e sulla posizione spaziale degli oggetti presenti nell'ambiente; informazioni che potrebbero tornare utili per una registrazione più affidabile o in tutti i casi in cui non si hanno a disposizione foto o informazioni di colore o luminosità, basti pensare a esplorazioni notturne o di gole o altri ambienti ostili.

Lo scopo di questa tesi è dunque lo sviluppo di un software che utilizzi metodi di registrazione basati sulla rilevazione di keypoints in 2D e 3D e la loro descrizione in 3D a supporto delle tecniche SLAM. Questo metodo innovativo, basato dunque su informazioni tridimensionali della scena, rende possibile la ricostruzione di scene anche in condizioni di scarsa luminosità.

Prima dello sviluppo del codice di elaborazione delle nuvole di punti, si sono studiati ed analizzati i principali metodi di registrazione descritti in letteratura a partire dalla registrazione manuale, ancora molto utilizzata per l'allineamento di poche nuvole di punti, passando agli algoritmi di ICP, e concludendo con la registrazione *features based*, ovvero basata sulla rilevazione e descrizione di

keypoints. Sono state effettuate diverse prove al fine di valutare pregi e difetti di ognuno.

Oltre alle prove effettuate su quattro diverse tipologie di ICP e su cinque diversi descrittori, tutti utilizzabili nel software implementato, si sottolinea il fatto che in questa tesi si è riuscito a confrontare rilevatori di keypoints 2D e rilevatori di keypoints 3D sfruttando il concetto di ripetibilità del keypoint e la proiettabilità della nuvola di punti fornita dal Kinect V2. Ad oggi non sono presenti in letteratura pubblicazioni e studi che presentino questo metodo.

Si è quindi passati allo sviluppo vero e proprio del software proposto. Per l'implementazione del codice sono stati utilizzati il sistema operativo Ubuntu, un ambiente di sviluppo QT, il linguaggio di programmazione C++ e la Point Cloud Library, una libreria all'interno di un progetto open source che permette la gestione delle nuvole [4]. Il software prevede: funzioni base per la gestione di nuvole di punti, come per esempio il caricamento, il salvataggio e l'eliminazione; funzioni per la loro elaborazione, come filtri di outliers, sottocampionamenti e calcolo delle normali; e funzioni per l'acquisizione con Kinect V2 di nuvole colorate, ottenute combinando le informazioni di distanza e l'immagine in scala di colori (RGB) catturata. La parte fondamentale del programma, poi, consiste negli algoritmi che da un insieme di acquisizioni sequenziali ricostruiscono tridimensionalmente la scena.

Le ricostruzioni 3D ottenute sono state comparate con quelle ricavate utilizzando altri tipi di sensore o altri algoritmi di registrazione. In particolare, si confrontano scene acquisite tramite laser scanner Stonex X300 [5], capace di ricostruire tutta la scena in un'unica acquisizione, hand scanner Sense [6], laser a triangolazione Minolta [7] e Kinect V2. Quest'ultimo dispositivo viene utilizzato sia all'interno del software sviluppato che all'interno di un software open source RTAB-Map [8], vincitore del prestigioso premio IROS 2014 Microsoft Kinect Challenge, con la differenza che quest'ultimo utilizza unicamente keypoint 2D.

Due casi studio vengono presentati. Il primo riguarda la ricostruzione di un sito archeologico situato a Noceto, un paese ad ovest di Parma, in cui è stata ritrovata una vasca costruita, per scopi religiosi, intorno al 1500 a.C. Far rivivere un reperto archeologico grazie alla fruizione del suo modello 3D virtuale è diventato negli ultimi anni un sistema molto utilizzato sia nei grandi musei che nelle università di Archeologia. Questa ricostruzioni infatti è stata richiesta dal docente del corso di Geoarcheologia dell'Università degli Studi di Milano e direttore dei lavori del sito, Mauro Cremaschi.

Il secondo caso applicativo descritto in questa tesi, nasce dall'esigenza di valutare e confrontare diverse tecniche per la ricostruzione di una scena del crimine. Per l'impossibilità di effettuare prove in vere scene del crimine, ne è stata ricreata una verosimile nel laboratorio Visual Brick LAB (VBLAB) del Politecnico di Milano.

Lo scopo di queste ricostruzioni è quello di congelare la scena e la posizione degli oggetti presenti in essa in modo da poterla analizzare in assenza di inevitabili contaminazioni e soprattutto per poter tenere traccia dei dati che saranno poi necessari nello sviluppo delle indagini. Entrambe le attività sono state svolte in collaborazione con il LABANOF il Laboratorio di Antropologia e Odontoiatria Forense situato presso la Sezione di Medicina Legale del Dipartimento di Morfologia Umana e Scienze Biomediche dell'Università degli Studi di Milano [9].

Lo sviluppo della tesi ha richiesto molto tempo per elaborare un'architettura per il software che lo renda robusto e ne migliori la flessibilità. Il software è pensato infatti per essere un hub di moduli per la gestione, l'elaborazione e la registrazione di nuvole di punti in modo da poter essere espanso e utilizzato in vari ambiti dai componenti del VBLab nei prossimi anni. Per dare un'idea del software implementato e dell'importanza data in questa tesi, si informa che in esso sono presenti più di 12000 righe di codice.

La struttura della tesi è descritta come segue.

Il **primo capitolo** illustra lo stato dell'arte relativo alle tecniche di misura 3D e alla descrizione degli strumenti di misura impiegati in esse. Particolare attenzione viene indirizzata alle telecamere a tempo di volo. Viene poi presentato il principale dispositivo utilizzato in questa tesi: il Kinect V2.

Il **secondo capitolo** introduce la tecnica SLAM e due strutture principali d'approccio utilizzate: la rete bayesiana dinamica e il Graph-Based.

Il **terzo capitolo** espande la descrizione del Graph-Based soffermandosi sulla fase di registrazione di nuvole. In particolare, vengono sviluppati i tre metodi principali: la registrazione manuale, con ICP o basata sulla descrizione di feature.

Il **quarto capitolo** descrive l'attività sperimentale nella quale si sono valutati i principali algoritmi su cui si basano le tecniche di registrazione: ICP, rilevatori e descrittori di keypoint.

Il **quinto capitolo** illustra il software implementato e i due casi studio sopra citati, riportando confronti anche con altri tipi di sensori e di algoritmi di registrazione. Seguono le conclusioni sui risultati ottenuti in questo studio.

Capitolo 1

Tecniche di scansione 3D

Le scansioni tridimensionali possono essere effettuate mediante un'ampia gamma di strumenti, che si avvalgono di diverse tecnologie ottimizzate in relazione ai differenti campi d'impiego. Nel seguito si descrive brevemente le caratteristiche principali di ciascuna tecnologia, presentando un quadro dello stato dell'arte attuale dal quale partire al fine di evidenziare i pregi, i difetti e le potenzialità di ognuna. Viene poi analizzata la struttura di una telecamera a tempo di volo presentando il modello matematico di una generale telecamera ed il modello di un dispositivo a tempo di volo ad onda continua. Infine sono descritti il principio di funzionamento e le caratteristiche tecniche di un dispositivo commerciale (Kinect V2), che sarà utilizzato in questa tesi per la ricostruzione di scene.

1.1 Introduzione

Le tecniche di misura di forme 3D si suddividono in sistemi a contatto e senza contatto. I metodi di acquisizione a contatto prevedono un'interazione fisica tra lo strumento e l'oggetto di misura. Un esempio di questo metodo è la macchina di misura, cartesiana o a giunti articolati, riportata in figura 1.1. Le misure vengono effettuate mediante l'utilizzo di un tastatore, il quale viene posto manualmente, o in modo automatico, a contatto con la superficie da analizzare. Per ogni punto viene registrata la posizione dell'end effector, rispetto ad un sistema di riferimento solidale alla macchina.

La tesi si concentrerà sullo studio di sistemi di acquisizione tridimensionale senza contatto, in quanto non risulta possibile la ricostruzione di una stanza tastando ogni singolo punto della scena, non solo per il tempo impiegato durante l'acquisizione e l'ingombro dello strumento, ma anche per non mutare l'ambiente da acquisire, a maggior ragione se la scena acquisita è una scena del crimine.

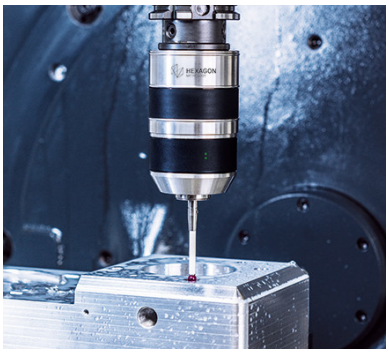


Figura 1.1: Particolare di un tastatore per misura con contatto.

I sistemi senza contatto possono essere suddivisi in due categorie: trasmissivi e riflessivi. I sistemi trasmissivi, come ad esempio la tomografia assiale computerizzata (TAC), sfruttano la proprietà dei raggi X di attraversare un corpo ed essere totalmente o parzialmente assorbiti in funzione delle caratteristiche del materiale che incontrano lungo il percorso tra emettitore e ricevitore. I sistemi trasmissivi risultano essere oltre che molto costosi, pericolosi per l'uomo e non adatti per la ricostruzione di una scena.

I sistemi riflessivi, di cui un modello è rappresentato in figura 1.2, si basano sulla capacità degli oggetti di riflettere le onde elettromagnetiche e/o le onde sonore. Essi si suddividono in sistemi non ottici ed ottici. I principali sistemi riflessivi non ottici sono il RADAR (RAdio Detection And Ranging) e il SONAR (SOund Navigation And Ranging). Entrambi misurano la distanza tra emettitore e bersaglio registrando il ritardo temporale tra l'invio e la ricezione dell'impulso elettromagnetico, nel primo caso, e sonoro, nel secondo. L'accuratezza dei sistemi non ottici non risulta sufficiente a rilevare il dettaglio che può essere presente in una scena indoor. Le tecniche di acquisizione 3D approfondite si limiteranno dunque ai sistemi di misura senza contatto riflessivi ottici.

1.2 Sistemi di visione riflessivi ottici

Come accennato, i sistemi di visione riflessivi si basano sulla capacità degli oggetti di riflettere le onde luminose nel visibile. Basati sull'impiego di telecamere, consentono un'analisi veloce ed esente dal contatto tra strumento di misura e misurando, applicabile in diversi ambiti. Essi presentano il vantaggio di essere in generale più veloci ed economici dei sistemi non ottici anche se il campo di misura, le proprietà dei materiali e le condizioni ambientali possono introdurre alcune limitazioni.

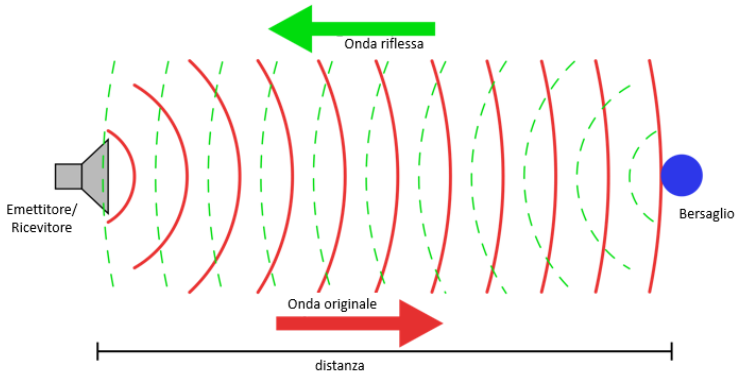


Figura 1.2: Modello di un sistema riflessivo.

Un distinzione basilare che separa le due diverse categorie di strumenti basati sulla radiazione luminosa, riguarda la sorgente di luce impiegata per effettuare la misura. Se la luce è quella ambiente, i metodi di misura si dicono passivi. Con un metodo passivo ci si limita a osservare l'ambiente, senza l'utilizzo di particolari sorgenti di luce. Se la luce che investe l'oggetto usata per ricavare l'informazione non è quella ambientale, come nei sistemi passivi, ma assume una struttura definita a priori, nota al sensore e prodotta da una sorgente tipica del sistema utilizzato, si parla di sistemi attivi.

Per questo in genere nei sistemi che usano questa tipologia di funzionamento la sorgente di luce è integrata nel dispositivo e l'illuminazione viene opportunamente sincronizzata con l'acquisizione. Questi sistemi in generale conducono a risultati aventi migliori accuratezze ma spesso comportano maggiori costi e presentano limiti di applicabilità.

1.2.1 Sistemi ottici passivi

I sistemi ottici passivi sono caratterizzati dalla mancanza di una fonte di illuminazione integrata. Nel seguito sono presentate le principale tecniche che appartengono a questa categoria.

Sfocamento

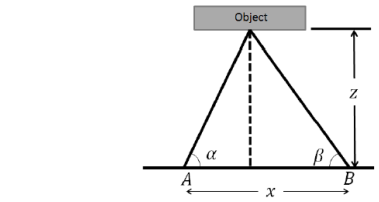
La tecnica consiste nell'acquisizione di diverse immagini di piani paralleli, spostando il fuoco dell'obiettivo lungo un asse che generalmente coincide con l'asse di propagazione della luce. Le informazioni così raccolte, costituite da una successione di immagini di piani focali adiacenti, permettono attraverso opportuni algoritmi di proiezione tridimensionale basati sulla relazione che lega profondità, sfocamento e caratteristiche dell'ottica, di ricostruire il volume osservato.

La necessità di dover acquisire numerose immagini, di calibrare la relazione sfocamento-profondità e di misurare in modo accurato lo sfocamento, in aggiunta all'utilizzo di ottiche professionali caratterizzate da elevate prestazioni (al fine di ottenere risultati accurati), rende questo metodo sia lento che costoso.

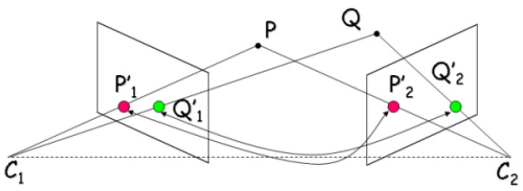
Stereoscopia o visione stereoscopica

Questo metodo simula il sistema di visione umano, ricava infatti le informazioni di profondità dall'analisi di immagini provenienti da due o più telecamere inquadranti la stessa scena da posizioni diverse. La ricostruzione 3D viene suddivisa in due fasi consecutive: la prima, basata sul principio di somiglianza, è finalizzata al reperimento, nelle due (o più) immagini, di punti omologhi (cioè che si riferiscono al medesimo punto fisico), la seconda, basata sui parametri di calibrazione di ogni singola camera (intrinseci), sulle loro posizioni reciproche (estrinseci) e su opportuni algoritmi, permette di effettuare la ricostruzione vera e propria determinando, per ogni n-upla di punti omologhi, il passaggio da coordinate 2D in pixel a coordinate 3D in millimetri.

Utilizzando un modello semplificato, rappresentato in figura 1.3, si può capire come questo sistema agisce. Dato un oggetto puntiforme nello spazio, la distanza tra le telecamere permette di misurare una disparità della posizione dell'oggetto nelle immagini acquisite dalle due telecamere.



(a)



(b)

Figura 1.3: Schema base di un sistema di visione stereoscopica semplificato (a) e schema in cui è evidenziato l'accoppiamento di punti omologhi (b).

Conoscendo la distanza tra i due punti di vista la posizione dell'oggetto può essere calcolata e rappresentata dai due angoli α e β. Con questi il calcolo della

profondità z è facilmente calcolabile tramite la formula trigonometrica:

$$z = \frac{x}{\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}} \tag{1.1}$$

La difficoltà maggiore nella visione stereoscopica è il problema della corrispondenza: dato un punto in un'immagine, trovare lo stesso punto nell'immagine dell'altra telecamera comporta una complessità algoritmica notevole.

È chiaro come la ricerca di punti corrispondenti sia resa difficile dalla presenza di occlusioni (punti visibili in alcune immagini ma nascosti in altre a causa di oggetti presenti nella scena), distorsioni fotometriche (superfici non perfettamente lambertiane) e distorsioni proiettive (fenomeno della prospettiva associato a due differenti punti di vista relativi allo stesso oggetto). Non è possibile stabilire un'accurata profondità non avendo un algoritmo di corrispondenza complesso e robusto.

1.2.2 Sistemi ottici attivi

I sistemi ottici attivi, caratterizzati dall'utilizzo di una fonte di illuminazione di tipo artificiale, si suddividono in tecniche a tempo di volo (time of flight) e metodi basati su luce strutturata (triangolazione laser, interferometria).

Interferometria

Il funzionamento consiste nel proiettare diversi patterns luminosi, come barre di luce parallele ed equispaziate o anelli concentrici sulla scena e filtrare l'acquisizione attraverso il medesimo pattern leggermente ruotato, così da generare frange di interferenza dalla cui analisi è possibile effettuare la ricostruzione 3D.

Il metodo è molto accurato ma per acquisire una singola nuvola di punti, il sistema richiede numerose iterazioni successive di patterns con accuratezza crescente al fine di raffinare sempre più il modello ricostruito, ciò comporta un tempo di acquisizione lungo limitando così l'utilizzo di questa tecnologia a soggetti statici. La figura 1.4 visualizza un esempio di pattern luminoso.

Luce Strutturata

Il principale sistema di acquisizione che appartiene a questa categoria si basa sulla triangolazione laser. Esso è composto da un proiettore, in grado di generare una lama laser, ed una telecamera, solidali alla medesima struttura ed opportunamente orientati così che i due componenti rispettivamente illuminino ed inquadrino la scena da due differenti posizioni. La misura è effettuata intersecando il raggio ottico della camera con la lama laser proiettata sull'oggetto ed effettuando calcoli

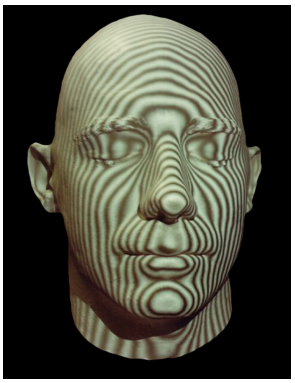


Figura 1.4: Interferometria: proiezione della griglia sul misurando.

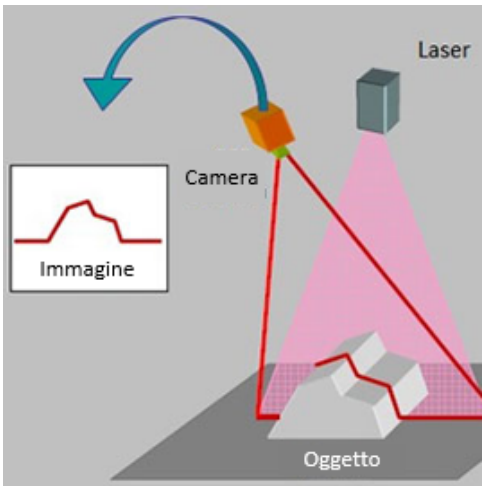


Figura 1.5: Principio di funzionamento della triangolazione attiva.

trigonometrici di triangolazione. Il dispositivo necessita di una preliminare calibrazione al fine di rilevarne il modello geometrico ma presenta il vantaggio di non dover individuare punti omologhi e permette di ottenere la ricostruzione molto accurata di un intero oggetto. Ogni immagine acquisisce un singolo profilo, per una ricostruzione 3D completa dell'oggetto servono più profili. La necessità di acquisire un elevato numero di immagini da diversi punti di vista rende la ricostruzione molto lenta e determina l'impossibilità di effettuare misure dinamiche.

Al fine di ridurre i tempi di ricostruzione, così da poter effettuare acquisizioni in ambiti real-time di scene in movimento, viene sostituito l'emettitore di lama laser con un dispositivo in grado di proiettare sulla scena un pattern casuale ma noto di spot luminosi ottenuto dalla diffrazione, mediante opportuno filtro, di un singolo raggio infrarosso. Il reticolo è deformato dalla scena ed assume curvature e dimensioni in relazione alla posizione degli oggetti presenti. L'acquisizione

del pattern deformato, effettuata mediante una telecamera sensibile solamente alla radiazione infrarossa generata dall'illuminatore, viene confrontata con un modello indeformato di riferimento acquisito in fase di calibrazione ad una distanza nota e memorizzato all'interno del sensore. A seguito di questa operazione, le coordinate 3D di ciascun punto vengono estratte mediante una procedura basata sulla triangolazione geometrica. Questa tecnologia è implementata in un dispositivo commerciale chiamato Kinect e presentato da Microsoft nel 2009.

Figura 1.6: Dispositivo Microsoft Kinect.

Tempo di volo

Questa tecnologia permette di effettuare misure puntuali di distanza e si divide in due sottocategorie: tempo di volo impulsato e ad onda continua (CW).

La prima tecnologia è concettualmente molto simile a quella utilizzata nei sonar. Viene misurato il tempo t trascorso tra l'invio di un'onda e la sua ricezione a seguito di una riflessione sulla scena. La distanza d tra dispositivo e bersaglio, pari alla metà di quella realmente percorsa dall'onda. La differenza tra le due tecnologie sta nel tipo di onda, a differenza del sonar, in cui si utilizza un'onda sonora, si utilizza un'onda luminosa. La distanza è calcolabile attraverso la relazione:

$$d = \frac{c \cdot t}{2} \tag{1.2}$$

in cui c è pari alla velocità della luce nell'aria. Si noti che, essendo presente nella formula la velocità della luce, un minimo errore nella misura del tempo si ripercuote con effetti importanti sulla misura della distanza. Per questo motivo sono necessari, soprattutto per piccole distanze, contatori molto accurati e costosi.

Nei sistemi ad onda continua (o continuous wave CW), invece, il segnale emesso viene modulato sinusoidalmente (o a onda quadra) alla frequenza f e lo spazio tra emettitore e bersaglio è calcolato misurando lo sfasamento ϕ tra segnale emesso e ricevuto, convertendolo in distanza mediante la seguente formula:

$$d = \frac{c}{4\pi f}\phi \tag{1.3}$$

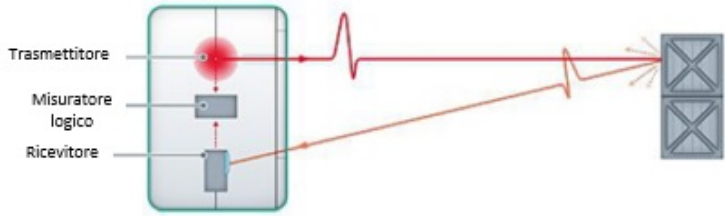


Figura 1.7: Principio di funzionamento di un dispositivo ToF impulsato.

La scansione di un'intera scena o di un oggetto 3D può essere effettuata per mezzo di misure ripetute, a distanze angolari note, così da acquisire un numero elevato di punti.

Ciò è reso possibile in strumenti di misura chiamati laser scanner, ed utilizzando un sistema di movimentazione in grado di orientare opportunamente il raggio laser. In particolare, all'interno dell'apparecchio è presente uno specchio posto in rotazione per mezzo di un motore elettrico, attorno ad un asse orizzontale, così da deflettere il raggio laser all'interno di un semipiano. Contemporaneamente il dispositivo ruota attorno ad un asse verticale permettendo così di ottenere, grazie ai due movimenti combinati, un insieme di punti 3D, detto nuvola di punti. Uno strumento di questo tipo, utilizzato in una delle applicazioni presentate nella tesi, è lo scanner laser Stonex X300. Con questa tecnica però, l'acquisizione risulta un processo lento attraverso il quale è impossibile effettuare misure dinamiche.

Questo problema può essere superato utilizzando illuminatori a luce IR (circa 800 nm) diffusa e sensori a matrice (spesso telecamere IR) grazie ai quali si ha la possibilità di acquisire l'intera scena a frequenze intorno a 30 Hz. Negli ultimi anni, la disponibilità sul mercato di dispositivi elettronici a basso costo ha permesso di realizzare telecamere low cost basate sulla tecnologia ToF ad onda continua. Un esempio di telecamera a tempo di volo è costituito dal Kinect V2, dispositivo commercializzato da Microsoft a partire dall'anno 2014 che presenta caratteristiche tecniche dichiarate dal costruttore nettamente superiori rispetto alla precedente versione. Prima di descrivere più dettagliatamente la telecamera Kinect V2, che sarà lo strumento principale nelle applicazioni di questa tesi, verrà presentato nella prossima sezione il modello matematico di un dispositivo a tempo di volo ad onda continua.

1.3 Telecamera a tempo di volo

Le telecamere a tempo di volo presentano al proprio interno una telecamera sensibile all'infrarosso con sensore composto da una matrice rettangolare di fotori-

velatori. Misurando numerose distanze puntuali sincrone in diversi punti sparsi della scena e allocandole appositamente in una matrice, è possibile acquisire la scena sotto forma di nuvola di punti 3D. Conoscendo per ogni punto acquisito la distanza lungo Z, nel sistema di riferimento solidale alla camera, e le relative coordinate u e v in pixel nella matrice del sensore, è possibile, a valle di un opportuno processo di calibrazione, atto ad estrarre i parametri intrinseci, applicare un modello di telecamera e ricavare le relative coordinate X ed Y. Dopo questo procedimento sono note le posizioni 3D dei punti acquisiti ed è quindi possibile ricostruire la nuvola di punti della scena.

La possibilità di acquisire l'intera scena con più frame al secondo rende possibile l'utilizzo di questa tecnologia anche in processi dinamici e quindi di acquisire anche oggetti in movimento.

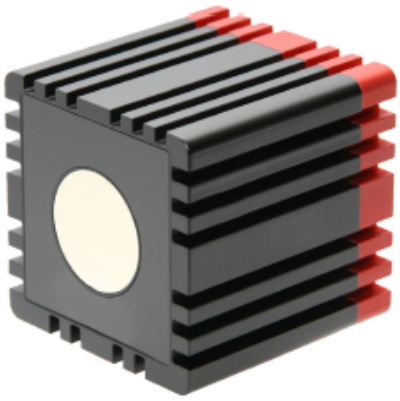


Figura 1.8: Esempio di telecamera a tempo di volo commerciale.

Fino a pochi anni fa le uniche telecamere a tempo di volo disponibili sul mercato, a causa del loro elevato costo, erano indirizzate ad un bacino di utenza ristretto ed il loro utilizzo era limitato all'ambito industriale e della ricerca. Inoltre, esse disponevano di ridotti angoli di visione e di scarsa risoluzione.

Ad esempio la telecamera Mesa Imaging SwissRanger 4000 (SR4000), riportata nell'immagine 1.8. Essa presenta un range di misura ampio 5 metri, un campo di vista di 69×55^o ed una risoluzione di 176×144 pixel. È in grado di acquisire fino a 30 frame al secondo (fps), e viene venduta a circa 9000 dollari.

A partire dagli ultimi mesi dell'anno 2013, Microsoft ha realizzato e commercializzato un dispositivo chiamato Kinect V2 basato su un sensore tof CW a basso costo (meno di 150 dollari) con accuratezze di misura soddisfacenti. Di questo dispositivo è stata effettuata, nel VBLab, una qualificazione metrologica nei mesi precedenti dell'inizio di questa tesi da Andrea Corti [10].

Nei prossimi paragrafi si descriveranno due modelli matematici su cui si basano le telecamere a tempo di volo. Il completo funzionamento di queste infatti può essere compreso solamente analizzando sia il modello matematico di una telecamera che quello di un dispositivo a tempo di volo.

1.3.1 Modello matematico di una telecamera

La telecamera è uno strumento in grado di acquisire immagini 2D. I componenti principali di una telecamera digitale moderna sono la parte ottica (obiettivo) composta da una o più lenti mobili oppure fisse, il sensore sensibile ad una determinata porzione dello spettro luminoso, ed un circuito elettronico di controllo. I fotoni, provenienti da un'opportuna fonte di illuminazione e riflessi dagli oggetti presenti nella scena, vengono catturati dall'obiettivo e focalizzati sul sensore, il quale, composto da una matrice di pixel, li converte in elettroni così da ottenere, grazie al circuito di controllo, un dato memorizzabile ed elaborabile digitalmente.

Un'immagine digitale, quindi, non è altro che una matrice ordinata contenente in ogni cella un valore numerico proporzionale alla quantità di fotoni acquisita dal singolo pixel nell'intervallo di tempo. Nella scala di grigi a 8 bit si rappresenta con il valore zero un pixel nero, cioè che ha catturato pochi (idealmente zero) fotoni, mentre con il valore 255 (pari a $2^8 - 1$) un pixel bianco, il quale è stato colpito da un numero molto elevato di fotoni. Tra i modelli geometrico-matematici presenti in letteratura, relativi a telecamere proiettive, il più comunemente utilizzato è chiamato *pinhole camera model* mostrato in figura 1.9.

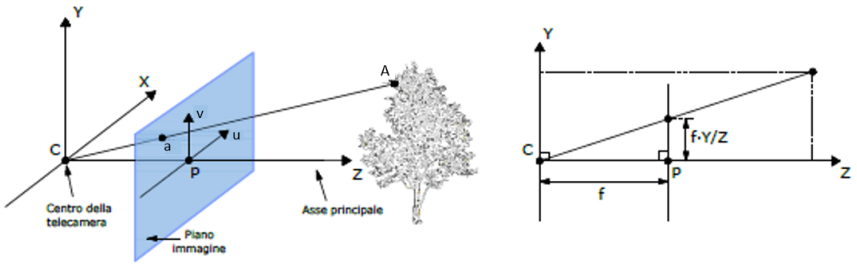


Figura 1.9: Geometria del modello lineare di camera pinhole.

In riferimento all'immagine del modello, si definisca C il centro ottico della telecamera e in esso si posizioni un sistema di riferimento XYZ definito da una terna destra cartesiana avente asse Z uscente dalla telecamera. Su quest'ultimo ad una distanza f detta focale, viene posto un secondo sistema di riferimento allineato (u, v), centrato sul punto p ed allineato con il precedente. Gli assi uv individuano un piano ortogonale a Z detto piano dell'immagine. La retta passante per C e p è detta asse principale della camera.

A partire da questo modello è possibile derivare la correlazione tra un punto A qualsiasi della scena ed il suo corrispondente a giacente sul piano immagine. Se si associa al generico punto dello spazio X un vettore colonna contenente le sue coordinate $(X; Y; Z)^T$, è possibile derivare una mappatura da uno spazio euclideo in R^3 in uno spazio euclideo in R^2.

$$(X, Y, Z)^T \rightarrow (u, v)^T$$

La relazione tra realtà e immagine 2D può essere espressa con la seguente formula lineare

$$(\lambda u, \lambda v, \lambda)^T = (Xf, Yf, Z)^T$$

e con la seguente formula matriciale:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

dove λ rappresenta un fattore di scala il cui valore è pari a Z. La matrice 3×4 che si ottiene è chiamata matrice di proiezione ed è indicata con la lettera $\mathbf{K}$. Il modello fin qui proposto è troppo semplice per essere utilizzato in applicazioni reali: è necessario perciò completarlo tenendo conto delle principali non idealità. In primo luogo si deve, infatti, notare che la posizione dell'origine del sistema di coordinate dei pixel non sempre coincide con il punto principale P, localizzato al centro dell'immagine. Risulta quindi necessario una conversione del sistema di coordinate. In questo caso, modificando la matrice di proiezione, si otterrebbe:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

dove c_x e c_y sono le coordinate del punto P nel nuovo sistema di riferimento. È possibile ora definire la matrice K:

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Inoltre i pixel delle telecamere non sono sempre perfettamente quadrati. Siccome normalmente le immagini vengono misurate in pixel, ciò rende diverso il fattore di scala orizzontale e verticale del sensore. Dimensioni non quadrate dei

pixel si traducono in un numero diverso di pixel per unità di lunghezza che comporta una scalatura diversa nella proiezione del punto lungo le due direzioni. Per tener conto di questo fenomeno la forma generale della matrice di calibrazione delle telecamere digitali diventa quindi :

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

dove $f_x = f \cdot m_x$ e $f_y = f \cdot m_y$ e m_x e m_y esprimono rispettivamente il numero di pixel per unità di lunghezza lungo x e lungo y.

Per far sì che il modello assuma maggiore attendibilità, è necessario considerare la possibilità che i pixel possiedano lati non perpendicolari tra loro, introducendo un ulteriore parametro chiamato skew e definito come:

$$s = \cos(\alpha) \cdot f$$

dove α è l'angolo formato dai lati del pixel. Generalmente il parametro s assume valore nullo essendo l'angolo α pari a $\pi/2$. Infine la matrice $\mathbf{K}$ può essere riscritta nella forma:

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Il principale vantaggio del pinhole camera è che descrive solamente relazioni lineari tra i parametri della telecamera. In molti casi tale modello offre una buona approssimazione della realtà. Non risulta, però, sufficientemente accurato nei casi in cui sono presenti delle distorsioni ottiche considerevoli. Tali distorsioni possono presentarsi in diversi modi, ma generalmente sono individuabili come delle non-linearità del sistema di visione. Il modello viene quindi modificato ulteriormente per avvicinarsi maggiormente alla geometria reale di una telecamera, nella quale i punti del mondo, nel piano immagine e il centro ottico non sono più collineari (ovvero collegati da una linea retta) come nel caso ideale. Le linee dell'immagine, che dovrebbero essere rette e parallele tra loro sono invece curvate in maniera tanto più accentuata quanto più si trovano ai bordi dell'immagine, come in figura 1.10a.

Tale distorsione viene chiamata radiale e può verificarsi in due modalità, a seconda del segno della distorsione; a botte oppure a puntaspilli 1.10b.

É necessario dunque correggere questa distorsione in modo da riportare il fenomeno di formazioni delle immagine nuovamente ad un caso lineare. Il processo per effettuare la correzione in coordinate pixel risulta:

(a) (b)

Figura 1.10: Immagine soggetta a distorsioni radiali a botte (a) e rappresentazione delle diverse tipologie di distorsioni radiali (b).

$$\widehat{x} = x_c + L(r)(x - x_c) \qquad \widehat{y} = y_c + L(r)(y - y_c)$$
$$\text{con} \qquad r^2 = (x - x_c)^2 + (y - y_c)^2$$

dove $(x; y)$ sono le coordinate immagine misurate, $(\widehat{x}; \widehat{y})$ sono le coordinate corrette e $(x_c; y_c)$ è il centro delle distorsioni radiali. La funzione $L(r)$ è definita solo per valori positivi di r e viene solitamente risolta come uno sviluppo di un'equazione di Taylor $L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + \dots$ I coefficienti di distorsione $k_1, k_2, k_3, \dots$ sono considerati parte della calibrazione interna della telecamera.

Un modello più accurato per trattare tali aberrazioni ottiche venne introdotto da Brown nel 1966 [11], chiamato modello *"Plumb Bob"* (piombino). Indicando con x_n le coordinate normalizzate di telecamera (x, y) la correzione delle coordinate diventa:

$$x_d = ()1 + k_2 r^2 + k_4 r^4 + k_6 r^6) x_n + dx \tag{1.4}$$

dove d_x è il termine che definisce lo spostamento introdotto dalle distorsioni tangenziali:

$$d_x = \begin{bmatrix} 2p_i xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy \end{bmatrix} \tag{1.5}$$

Si è infine giunti al modello completo per descrivere la telecamera. Questo è caratterizzato da 16 GDL, separabili in intrinseci ed estrinseci. Tali parametri possono essere determinati tramite operazioni di calibrazione. Il modello completo possiede quindi questi parametri:

5 parametri intriseci:

- f_x e f_y chiamati distanze focali ed espressi lungo x e y.
- c_x e c_y relativi alle coordinate del centro ottico.
- s chiamato skew e proporzionale all'angolo α presente nei lati del pixel.

6 parametri estrinseci

- r_1, r_2, r_3 utilizzati per esprimere la rotazione tra SDR camera e SDR mondo;
- t_1, t_2, t_3 coordinate del vettore traslazione del centro camera nel SDR mondo.

5 parametri di distorsione:

- k_2, k_4, k_6 relative alle distorsioni radiali;
- p_1, p_2 relativi alle distorsioni tangenziali.

Non sempre però è necessario ricorrere all'utilizzo del modello completo; in alcuni casi, infatti, esso introduce una complessità tale da rivelarsi eccessiva per l'applicazione in esame. Spesso le distorsioni sono trascurabili ed in particolare è abbastanza frequente che non si considerino i coefficienti p_1 e p_2 relativi alle distorsioni tangenziali ed il coefficiente k6 di distorsione radiale.

1.3.2 Modello matematico di un dispositivo a tempo di volo a onda continua.

Il modello base di un sistema di misura lineare ottico a tempo di volo continuous wave è composto da un emettitore di luce (un diodo o un led), con lunghezza d'onda contenuta nello spettro dell'infrarosso (IR), un sensore (fotodiodo) opportunamente filtrato e sensibile alla luce IR ed un circuito elettronico che controlli il segnale inviato ed acquisisca i dati provenienti dal ricevitore [12]. In figura 1.11 è presente uno schema utilizzato per descrivere il modello matematico di un dispositivo ToF a onda continua. Un raggio luminoso, modulato sinusoidalmente alla frequenza f_{mod}, viene rilevato da un fotodiodo a seguito di una riflessione da parte del bersaglio. Si nota come il segnale ricevuto (blu) presenti ampiezza a minore rispetto al segnale di partenza (rosso), poiché la diffusività della superficie

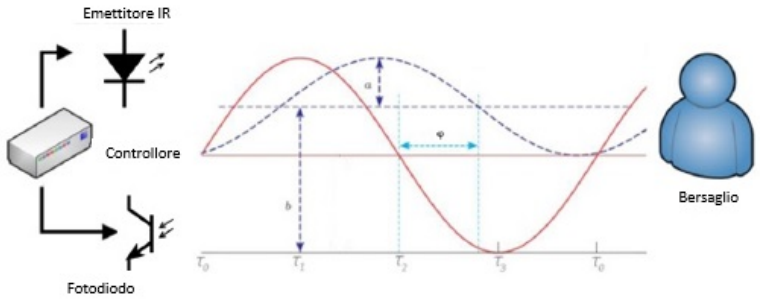


Figura 1.11: Schema di funzionamento di un dispositivo ToF a onda continua.

riflettente disperde parte della luce ricevuta e come sia presente un offset costante b causato dal contributo della luce ambientale non bloccate dal filtro IR montato sull'ottica della camera. Infine, è chiaramente visibile uno sfasamento ϕ proporzionale alla distanza tra emettitore e bersaglio. Per questo motivo la misura della distanza è ricavata dallo sfasamento.

La luce sorgente illumina per un breve periodo al fine di misurare il ritardo di fase associato al segnale riflesso, una tecnica comunemente utilizzata prevede di acquisire 4 valori di ampiezza (Q_1, Q_2, Q_3, Q_4) relativi al segnale ricevuto, in istanti temporali $(\tau_0, \tau_1, \tau_2, \tau_3)$ sfasati 90^o l'uno dall'altro e sincronizzati col periodo del segnale inviato. Usando questa tecnica, l'angolo di fase ϕ e la distanza d possono essere calcolate dalle formule:

$$\phi = \arctan\left(\frac{Q_3 - Q_4}{Q_1 - Q_2}\right) \tag{1.6}$$

$$d = \frac{c}{4\pi f_{mod}}\phi$$

Analizzando l'equazione (1.6) si nota come i termini $(Q_3 - Q_4)$ e $(Q_1 - Q_2)$ riducono l'effetto di un eventuale offset di misurazione. Inoltre il rapporto $\frac{Q_3 - Q_4}{Q_1 - Q_2}$ ha lo scopo di normalizzare i valori e eliminare gli effetti dovuti alla variazione di luce, dovute a differenti riflettività del bersaglio. Segue allora che la misura l'ampiezza dell'intensità luminosa misurata a e l'offset b sovrapposto al segnale ricevuto possono essere calcolati da:

$$a = \frac{\sqrt{((Q_1 - Q_2)^2 + (Q_3 - Q_4)^2)}}{2}$$

$$b = \frac{Q_1 + Q_2 + Q_3 + Q_4}{4}$$

L'ampiezza dell'onda riflessa a e l'offset b influenzano l'accuratezza della misura di distanza. La varianza della misura di distanza infatti può essere approssimata

tramite la formula:

$$b = \frac{c}{4\sqrt{2}\pi f_{mod}} \cdot \frac{\sqrt{a+b}}{c_d a}$$

dove c_d, il contrasto di modulazione, è un valore che descrive la capacità del sensore a tempo di volo di distinguere due diversi fotoelettroni. Analizzando l'equazione si può comprendere come un aumento dell'ampiezza dell'onda riflessa a, della frequenza di modulazione f_{mod}, del contrasto di modulazione c_d e una diminuzione dell'offset b implica un aumento dell'accuratezza del sistema.

Poiché la frequenza f_{mod} è nell'ordine di 10^6 Hz mentre la frequenza di acquisizione f_{acq}, con cui vengono restituiti i valori di distanza, è nell'ordine di decine di Hertz, il sistema è in grado di effettuare numerose misure ripetute della fase ϕ e restituire il valore medio migliorando l'accuratezza del sistema.

Il fatto che i sistemi ToF a onda continua si basino sulla misura della fase porta una notevole limitazione in termini di distanza massima misurabile. La fase deve essere contenuta in 2π perché non si verifichino effetti di ambiguità, quindi la distanza massima misurabile, detta anche distanza ambigua, risulta essere:

$$d_{amb} = \frac{c}{2f}$$

Questa limitazione è chiaramente estendibile riducendo la frequenza di modulazione e diminuendo però l'accuratezza della misura. Invece di accettare questo compromesso, sistemi ToF di nuova generazione utilizzano tecniche a frequenza multipla per estendere la distanza senza ridurre la frequenza di modulazione.

Le tecniche a frequenza multipla fondano la propria teoria di funzionamento sull'utilizzo di due o più frequenze con le quali viene emesso il segnale luminoso, ciascuna delle quali è caratterizzata da una propria distanza ambigua e restituisce un valore di distanza. Tramite il set di misure della distanza, effettuate mediante l'utilizzo di una diversa frequenza modulante, il sistema attribuisce la stima della posizione dell'oggetto misurato. Nella semplice ipotesi di un sistema basato su doppia modulazione, si otterranno due valori di distanza d_1 e d_2, rispettivamente attribuibili alle frequenze di modulazione f_1 e f_2, ottenuti tramite le formule:

$$d_1 = \frac{c}{4\pi f_1}(\phi + h2\pi)$$

$$e$$

$$d_2 = \frac{c}{4\pi f_2}(\phi + k2\pi)$$

con h e k interi e positivi.

Il sistema determina, tramite un opportuno algoritmo, i valori h e k tali per cui le distanze d_1 e d_2 risultino uguali.

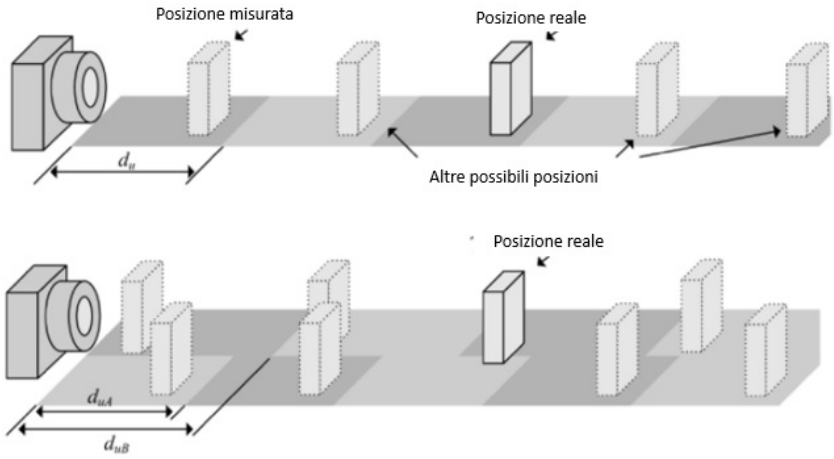


Figura 1.12: In alto, modello del sistema utilizzando un'unica frequenza di modulazione e, in basso, principio di estensione della distanza di misura utilizzando una tecnica a frequenza multipla.

La tecnologia ToF fino ad ora descritta è limitata a singole misura puntiformi di distanza. Come già descritto nella sezione precedente, è possibile usare questa tecnica per la ricostruzione di nuvole di punti 3D deflettendo il raggio luminoso mediante specchi rotanti. Questo richiede, però, dei tempi di acquisizione elevati che non permettono l'acquisizione real time.

Il problema può essere risolto sostituendo il laser IR con un emettitore a luce diffusa, ed il singolo fotorivelatore con un sensore matriciale. I dispositivi così descritti vengono chiamati telecamera a tempo di volo.

1.4 Microsoft Kinect V2

L'importanza che Microsoft ripone in questo sensore è comprensibile da alcune scelte commerciali. Nasce come accessorio integrato alla XBox One al fine di permettere il controllo di videogame attraverso gesti e movimenti del corpo, senza l'ausilio di alcun controller o altro dispositivo impugnato o indossato dal giocatore. Nella maggior parte dei videogame prodotti però è necessario un controller per assicurarne la giocabilità, rendendo il Kinect V2 poco utilizzato e desiderato dalla gran parte degli utenti. La decisione di Microsoft di integrare nella XBox One il Kinect V2 ha fatto lievitare il costo commerciale della console Microsoft di circa 100 euro, portando così una disparità di prezzo tra questa e la PlayStation 4, la principale console concorrente. Sapendo che, tolto il Kinect, le prestazioni delle due console sono circa le stesse, Microsoft ha assunto un rischio integrandolo nella propria console, poiché il consumatore medio non interessato al Kinect avrebbe

potuto considerare la differenza di costo tra le console eccessiva e deviare la propria scelta.

Questo rischio è giustificato dalla possibilità di ripartire i costi fissi di produzione, cioè quelli non direttamente influenzati dal crescere del volume di produzione. Integrando il Kinect V2 nella console ne si assicura la vendita, quindi un certo volume di produzione, abbassando così i costi fissi e di conseguenza i costi di produzioni totali, del Kinect V2.

La rischiosa scelta ha permesso lo sviluppo di questa tecnologia migliorando le prestazioni di misura e rendendo il dispositivo alla portata di tutti.

Figura 1.13: Microsoft Kinect V2: componenti nell'immagine a sinistra e sistema di riferimento a destra.

Nel dettaglio, esso si basa su una telecamera tof CW a variazione di fase composta da un emettitore IR realizzato mediante un array di 3 laser dotati di apposito filtro diffusore e da una telecamera monocromatica sensibile alla radiazione infrarossa con risoluzione di 512×424 pixel e campo di vista orizzontale ampio 70^o e verticale ampio 60^o. Il range di misura si estende, lungo l'asse Z uscente dalla camera, da 500 mm a 4500 mm (tabella 1.1).

Inoltre, all'interno dell'oggetto, sono presenti una telecamera RGB con risoluzione pari a 1080×1920 pixel ed un array di 4 microfoni.

Dato che il dispositivo presenta un cavo di collegamento terminato con un connettore proprietario, al fine di fornire alimentazione elettrica, stabilire la connessione ad un PC, e poter dunque acquisire i dati, viene venduto (a circa 50 dollari) un apposto kit composto da un convertitore AC-DC e da un adattatore USB 3.0, che permette la trasformazione da Kinect V2 for Xbox a Kinect V2 for Windows. Al fine di ottenere il corretto funzionamento della periferica su Windows sono richiesti i requisiti di sistema mostrati nella tabella 1.2.

Presso il laboratorio Visual Bricks del Politecnico di Milano, dove sono stati effettuati gli studi descritti in questa tesi, nell'ultimo anno è stata effettuata una qualificazione metrologica delle accuratezze del sensore Kinect V2 [10].

Come già accennato, in questo lavoro è stato sviluppato un software per la ricostruzione di scene utilizzando il sistema operativo Ubuntu, in modo da ottenere

Caratteristiche principali del Kinect V2.		
Risoluzione Camera IR	[pix]	512x424
Risoluzione Camera RGB	[pix]	1080x1920
Frame rate massimo	[Hz]	30
Campo visivo	[^{o}]	$70(H)$x$60(V)$
Range di misura	[mm]	500x4500
Dimensioni	[mm]	250x66x67
Peso	[g]	966
Collegamento		USB 3.0
Alimentazione		Adattatore per rete elettrica
Altro		4 microfoni

Tabella 1.1: Kinect V2 for Windows - principali caratteristiche.

Requisiti minimi di sistema per l'utilizzo del Kinect V2.	
CPU	Intel Core i7 3.1 GHz (or higher)
RAM	4 GB (or more)
GPU	Direct X 11 supported
USB	USB 3.0
Language	C++,VB.Net,HTML

Tabella 1.2: Requisiti minimi di sistema.

un software open source e non essere vincolati dalle licenze d'uso di altri sistemi operativi. Mentre in Windows esistono programmi che restituiscono una nuvola di punti colorata acquisita dal sensore Kinect V2, in Ubuntu è unicamente possibile ricavare singolarmente le immagini a colori, le immagini IR e le immagini di profondità utilizzando delle librerie open sources, chiamate "*libfreenect2*". Le informazioni delle immagini a colori e quelle nelle immagini di profondità devono essere assemblate tenendo conto della matrice $\mathbf{K}$ definita nel paragrafo 1.3.1, in modo da ricavare delle nuvole di punti colorate. La tabella 1.3 riporta i valori di calibrazione relativi alla telecamera IR del dispositivo Kinect V2 utilizzato nel presente lavoro.

Parametri intriseci di calibrazioni.		
Parametri	**Valori**	**Simboli**
Lunghezza focale X	257.615	c_x
Lunghezza focale Y	201.567	c_y
Centro Ottico X	365.102	f_x
Centro Ottico Y	365.102	f_y
Distorsione radiale 2^o ordine	0.092957	k_2
Distorsione radiale 4^o ordine	-0.27154	k_4
Distorsione radiale 6^o ordine	0.095053	k_6

Tabella I.3: Parametri intrinseci di calibrazione relativi alla telecamera IR.

Capitolo 2

SLAM: Simultaneous Location And Mapping

Nel capitolo seguente si darà una descrizione della tecnica di Simultaneous Localization And Mapping (SLAM). Dopo una breve introduzione, vengono descritte le due principali strutture per l'approccio alla tecnica, la rete bayesiana dinamica e il Graph-Based. Si concentra poi l'attenzione su quest'ultima poiché la sua struttura verrà analizzata e sviluppata durante il percorso di questa tesi.

Costruire una mappa di un ambiente e simultaneamente posizionarsi in esso senza la mappa stessa, è una capacità essenziale per la navigazione di un sistema mobile in una scena non conosciuta. Questo problema chiamato *Simultaneous Location And Mapping*, o più semplicemente SLAM, è uno dei più popolari argomenti di ricerca nella movimentazione autonoma di robot da oltre due decadi.

Attraverso l'elaborazione delle misure ricavate durante l'esplorazione da diversi tipi di sensori (quali per esempio GPS od odometri), viene stimato probabilisticamente la traiettoria percorsa (*localization*), e allo stesso tempo ricostruita una mappa dell'ambiente (*mapping*) grazie a informazioni ricavate da ulteriori sensori (quali per esempio telecamere o scanner a tempo di volo). Nella letteratura sono disponibili una grande varietà di soluzioni al problema. Questi approcci possono essere classificati in *filtering* e *smoothing*.

L'approccio di tipo *filtering* modellizza il problema mediante una stima in tempo reale dello stato del sistema, dove lo stato è composto dalla posizione del robot e dalla mappa. La stima è rifinita incorporando le nuove informazioni quando queste diventano disponibili. Le tecniche più popolari di questo tipo sono il filtro di Kalman [13], filtri particolari [14] [15] [16] o filtri di informazione [17] [18].

L'approccio di tipo *smoothing*, invece, stima tutta la traiettoria del sistema mobile dal set completo delle misurazioni. Questi approcci sono chiamati anche

problemi di Full-SLAM, e tipicamente dipendono da tecniche di minimizzazione dell'errore quadratico. Un modo intuitivo per affrontare il problema SLAM è tramite la formulazione graph-based: la risoluzione del problema SLAM comporta la costruzione di un grafico i cui nodi rappresentano la posa del robot e gli edges, cioè i collegamenti tra i nodi, indicano un vincolo tra i nodi stessi, dovuto ad una misurazione del sensore. Ovviamente alcuni vincoli possono essere contraddittori in quanto le osservazioni sono affette da rumore. Una volta che il grafico è costruito, il problema cruciale diventa quello di trovare una configurazione di nodi che sia consistente con le misurazioni. Questo implica la risoluzione di un problema di minimizzazione dell'errore.

Scopo del paragrafo successivo sarà quello di introdurre il problema dello SLAM in forma probabilistica e sintetizzare lo stato dell'arte del metodo graph-based SLAM.

2.1 Formulazione probabilistica dello SLAM

A causa del "rumore" presente nell'immagine e nella nuvola di punti 3D, entrambe acquisite dalla telecamera 3D, il problema SLAM può essere descritto efficacemente tramite un modello probabilistico. Si assume che un robot su ruote si muova in un ambiente sconosciuto, lungo una traiettoria descritta da una sequenza casuale di variabili $x_{(1:T)} = x_1, ..., x_T$. Mentre si muove, acquisisce una sequenza di misure odometriche, come $u_{(1:T)} = u_1, ..., u_T$ e informazioni percepite dell'ambiente $z_{(1:T)} = z_1, \ldots, z_T$. La risoluzione del problema dello SLAM consiste nello stimare la probabilità della traiettoria del robot $x_{(1:T)}$ e la mappa m dell'ambiente, date tutte le acquisizioni ed una posizione iniziale x_0:

$$p(x_{1:T}, m | z_{1:T}, u_{1:T}, x_0)$$

La posizione iniziale x_0 definisce la posizione della mappa e può essere scelta arbitrariamente. La stima di questa probabilità è difficilmente trattabile senza una struttura che definisca il problema dello SLAM. Questa struttura emerge da due presupposti certi e comuni:

- Ambiente statico (Static World Assumption): l'ambiente rimane immutato durante tutta la procedura di mapping. Questo presupposto è chiaramente fondamentale, anche se si cerca sempre più di rendere robusto la tecnica anche con piccoli cambiamenti dell'ambiente, come può avvenire durante l'acquisizione di una strada percorsa da passanti.

- Assunzione di Markov: un processo stocastico gode della proprietà di Markov se la distribuzione di probabilità condizionata degli stati futuri del processo dipende solamente dallo stato corrente e non da quelli passati.

Esistono due modi principali per descrivere la struttura del problema SLAM: tramite una rete Bayesiana dinamica (DBN) o il Graph-Based [2], entrambi saranno presentati nei prossimi paragrafi.

2.2 Rete Bayesiana Dinamica

Una rete bayesiana è un modello grafico che descrive il processo stocastico tramite un grafico orientato. Questo grafico è composto da un nodo per ogni variabile, e da frecce tra i nodi che identificano una dipendenza tra essi.

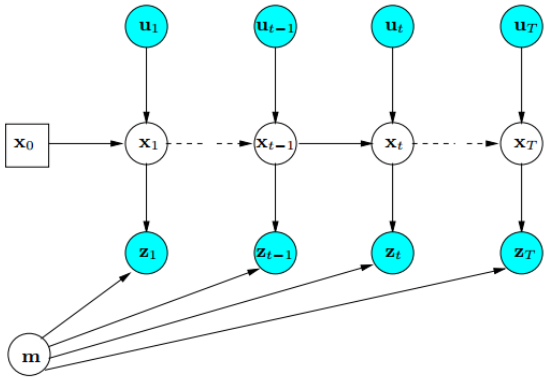


Figura 2.1: Rete Bayesiana dinamica di un processo SLAM.

Nella figura 2.1, è possibile distinguere nodi blu che rappresentano le variabili osservate ($z_{(1:T)}$ *e* $u_{(1:T)}$), cioè quelle misurate, e nodi bianchi, le variabili latenti che devono essere ricavate dalle variabili osservate. Le variabili latenti $x_{(1:T)}$ e m modellizzano la traiettoria del robot e la mappatura dell'ambiente.

I collegamenti della rete bayesiana dinamica seguono uno schema descritto da un modello di transizione e un modello di osservazione.

- Il modello di transizione $p(x_t \mid x_{(t-1)}, u_t)$ è rappresentato dalle due frecce in ingresso a x_t e identifica la probabilità che un robot al tempo t sia in x_t sapendo che la sua posizione nel tempo $t-1$ era in $x_{(t-1)}$ e avendo acquisito una misura odometrica u_t.
- Il modello di osservazione $p(z_t \mid x_t, m_t)$, rappresentato attraverso le frecce entranti in z_t identifica la probabilità di rappresentare l'acquisizione z_t nella mappa, sapendo che il robot si trova nella posizione x_t.

L'acquisizione z_t dipende solamente dalla posizione corrente x_t e dalla mappa statica m. Esprimere lo SLAM come una rete bayesiana dinamica rende visibile

la sua struttura temporale. Per questo è utilizzata in letteratura per descrivere i processi di tipo filtering.

Le variabili $u(t)$ rappresentano nello schema le grandezze acquisite che permettono di stimare le posizioni ad ogni istante. Nell'esempio analizzato finora, esse consistono nelle misure odometriche ottenute integrando le informazioni degli encoder posti sulle ruote. Nel nostro caso invece $u(t)$ sono rappresentate dalle nuvole di punti acquisite poiché tramite esse, e la posizione x_{t-1}, è possibile stimare la posizione $x(t)$. La variabile $z(t)$ invece identifica la percezione della scena che si avrebbe nella posizione $x(t)$.

2.3 Graph-Based

Una rappresentazione alternativa alla rete bayesiana dinamica è la formulazione graph-based del problema SLAM, che ne sottolinea la struttura spaziale. In un Graph-Based SLAM le pose del robot sono modellizzate tramite nodi in un grafico ed identificate tramite la loro posizione nell'ambiente. I legami spaziali tra le pose, che vengono formati grazie alle informazioni risultanti dalla rappresentazione sulla mappa o da misure odometriche successive, sono rappresentate da frecce tra i nodi.

Ogni nodo nel grafico rappresenta la posizione e la misura acquisita in quella posizione. Un collegamento fra due nodi rappresenta un legame spaziale, relativo alle due pose del robot, che consiste in una distribuzione di probabilità della trasformazione relativa tra le due pose. Queste trasformazioni sono dovute o a misure odometriche tra posizioni temporalmente sequenziali del robot, o all'allineamento tra nuvole spazialmente vicine. Dopo aver costruito il grafico, si cerca di trovare la configurazione delle pose del robot che meglio soddisfa i legami [19]. In uno SLAM di tipo graph-based il problema è quindi diviso in fasi:

- Costruzione del grafico (Front-End): si costruisce il grafico dalle misurazioni svolte, questa fase è fortemente dipendente dal sensore.
- Ottimizzazione del grafico (Back-End): consiste nel determinare la miglior configurazione delle pose data la struttura del grafico, dati cioè i collegamenti tra i nodi nel grafico. Questa parte fa affidamento su una rappresentazione astratta dei dati che il sensore ha acquisito.

2.4 SLAM Front-End

Lo SLAM Front-End può essere diviso in due fasi, la *Registrazione* e il *Loop Closure Detection*.

A) Registrazione

L'allineamento iterativo tra tutte le nuvole di punti temporalmente vicine è fondamentale per avere una prima e approssimata rappresentazione della scena acquisita. Lo scopo della registrazione è dunque quello di stimare posizione e orientazione di viste acquisite separatamente, in un sistema di coordinate globali, di allinearle e mescolarle insieme in una singola nuvola di punti in modo che le aree di intersezione coincidano.

Nella letteratura, il metodo utilizzato consiste nello stimare una trasformazione approssimata tra due nuvole di punti consecutive e nel rifinire tale stima in un secondo passaggio.

Gli algoritmi per allineare due nuvole di punti sono molteplici e descritte, per l'importanza di questa fase, nel capitolo 3.

B) Loop Closure Detection

Il *loop closure detection* consiste nel rilevare quando il sistema ritorna in una posizione passata dopo aver percorso una certa traiettoria. La rilevazione dei *loop closure* è una parte importante della fase Front-End, senza questo infatti il grafico risulterebbe una catena lineare che collega unicamente le pose acquisite consecutivamente nel tempo. I loop, invece, rappresentano legami tra nodi che non sono temporalmente adiacenti. Una volta che il loop è stato rilevato la nuova corrispondenza tra i due nodi del loop può essere usata come una vincolo addizionale del grafo utilizzato per la risoluzione dello SLAM.

Nella letteratura le tecniche di loop closure possono essere classificate, come descritto in [20] in 3 categorie principali.

1. *Map to Map*: come indica il nome, è usata in metodi che dipendono dalla costruzione di sotto-mappe di un unico ambiente, sono calcolate le corrispondenze tra sotto-mappe ed unite tramite una trasformazione.

2. *Image to Map*: la nuvola di punti più recente è comparata con le features dell'intera mappa in modo da trovare corrispondenze su di essa. La posa della nuvola di punti quindi è determinata in relazione con la mappa di features fino a quel momento acquisita.

3. *Image to Image*: si cercano corrispondenze tra la più recente nuvola di punti e quelle acquisite in precedenza.

Il metodo più utilizzato è il rilevatore di loop di tipo image to image.

2.5 SLAM Back-End

La registrazione tra acquisizioni successive è un buon metodo per tracciare la posizione di un robot solamente per una breve distanza. Il rumore presente nelle acquisizioni e l'incertezza conseguente alla stima della trasformazione di spostamento, infatti, causano un errore crescente che si accumula all'aumentare della lunghezza della catena. Questo porta ad una mappa con scostamenti importanti dalla realtà. Svolgendo unicamente la fase front-end, quindi, l'errore accumulato tra la posa stimata e la posa reale diverge all'aumentare del numero di nuvole acquisite.

Il ruolo dello SLAM Back-End è di ottimizzare la mappa, riducendo questo errore, utilizzando il grafo fornito dalla fase Front-End. Il grafo è composto da n vertici, in ognuno dei quali è allocata l'acquisizione ad una certa posa, e da legami che rappresentano una relazione tra le pose. Questa tecnica di ottimizzazione globale del grafo cerca di stimare al meglio tutte le pose per costruire una mappa consistente dell'ambiente acquisito.

2.5.1 Formulazione del problema

Per una migliore comprensione viene presentato il problema nell'ambito bidimensionale. Si consideri un robot dotato di un scanner 3D, che si sposta lungo un percorso e attraversa $n+1$ pose $X_0, \dots X_n$. Ad ogni posa X_i viene acquisita una scansione laser a 360^o dell'ambiente circostante. In figura 2.2 è visibile il modello del sistema.

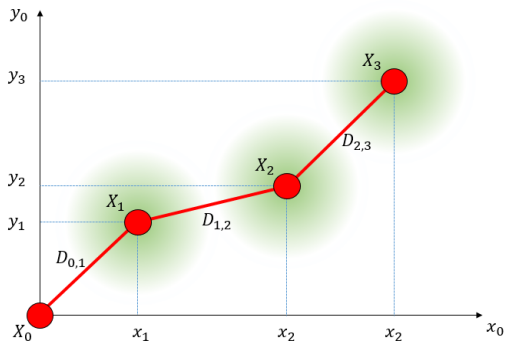


Figura 2.2: Modello Graph-Based di un sistema a 2 gradi di libertà

Allineando le acquisizione consecutive, è possibile ottenere una serie di relazioni tra queste pose. Nella rete risultante, le pose sono rappresentate come nodi e le relazioni come linee di collegamento tra essi.

Data una rete di nodi $X_0, \ldots, X_n$ e di collegamenti $D_{i,j}$, lo scopo è quello di stimare tutte le pose in modo da formare una mappa consistente della scena.

Per semplificare, l'equazione di misura viene considerata lineare:

$$D_{i,j} = X_i - X_j$$

L'osservazione $\overline{D}_{i,j}$ della reale differenza è modellizzata come $\overline{D}_{i,j} = D_{i,j} + \Delta D_{i,j}$, dove l'errore $\Delta D_{i,j}$, è una distribuzione gaussiana variabile random con media zero e una varianza conosciuta $C_{i,j}$.

La stima della massima verosimiglianza è utilizzata per approssimare la posa ottima X_i. Sotto l'assunzione che tutti gli errori nelle osservazioni siano gaussiani e distribuiti in modo indipendente, massimizzare la probabilità $P(D_{i,j}|\overline{D}_{i,j})$ è l'equivalente di minimizzare la seguente distanza di Mahalanobis [21]:

$$W(x) = \sum_{i,j} (D_{i,j} - \overline{D}_{i,j})^T C_{i,j}^{-1} (D_{i,j} - \overline{D}_{i,j}) \tag{2.1}$$

Questo significa trovare la soluzione a:

$$X^* = \underbrace{argmin}_{X} W(x)$$

Il valore minimo della funzione $W(x)$ non è importante, ciò che importa è il valore di X, X^*, per cui la funzione risulta minima.

2.5.2 LUM

Il metodo LUM, proposto da Lu e Milios [22], fornisce una soluzione al problema Back-End di ottimizzazione della rete. L'obbiettivo dell'algoritmo è ricavare la mappa dell'ambiente basandosi sui legami tra le diverse pose. Tali legami devono essere corretti, poiché influenzati dall'errore di misura, per ottenere una mappa consistente.

Senza perdita di generalità è possibile assumere che tutti i nodi della rete siano connessi tra di loro da $D_{i,j}$. Nel caso in cui una connessione tra il nodo i-esimo e il nodo j-esimo la corrispondenza $C_{i,j}^{-1}$ viene impostata a zero. Considerando un semplice caso lineare, l'equazione 2.1 diventa:

$$\mathbf{W} = \sum_{0 \leq i \leq j \leq n} (X_i - X_j - \overline{D}_{i,j})^T C_{i,j}^{-1} (X_i - X_j - \overline{D}_{i,j}) \tag{2.2}$$

Per minimizzare questa equazione, è definito un sistema di coordinate impostando un nodo come punto di riferimento. Per semplicità si utilizza $X_0 = 0$ e gli altri nodi $X_1, \ldots, X_n$ denotano le pose relative a X_0. Usando la matrice di

incidenza $\mathbf{H}$, che contiene i legami tra le diverse pose, l'equazione concatenata delle relazioni di misura tra i nodi D è scritta come:

$$\mathbf{D} = \mathbf{HX},$$

con $\mathbf{X}$ la concatenazione di X_1 fino a X_n. La distanza di Mahalanobis può essere scritta come:

$$\mathbf{W} = (\overline{\mathbf{D}} - \mathbf{HX})^T \mathbf{C}^{-1} (\overline{\mathbf{D}} - \mathbf{HX}).$$

La concatenazione di tutte le osservazioni $\overline{D}_{i,j}$ forma il vettore $\overline{\mathbf{D}}$ e $\mathbf{C}$ è una matrice di blocchi diagonali costituiti dalle matrici di covarianza. La soluzione $\mathbf{X}$ che minimizza l'equazione 2.2 e la sua covarianza $\mathbf{C_X}$ è data da:

$$\mathbf{X} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{C}^{-1} \overline{\mathbf{D}}$$

$$\mathbf{C_X} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1}.$$

La matrice $\mathbf{G} = \mathbf{H}^T \mathbf{C}^{-1} \mathbf{H}$ e il vettore $\mathbf{B} = \mathbf{H}^T \mathbf{C}^{-1} \overline{\mathbf{D}}$ semplificano l'equazione. $\mathbf{G}$ è composta dalle sottomatrici:

$$G_{i,i} = \sum_{j=0}^{n} C_{i,j}^{-1}$$

$$G_{i,j} = C_{i,j}^{-1} \quad (i \neq j)$$

La matrice $\mathbf{B}$ è ottenuta da:

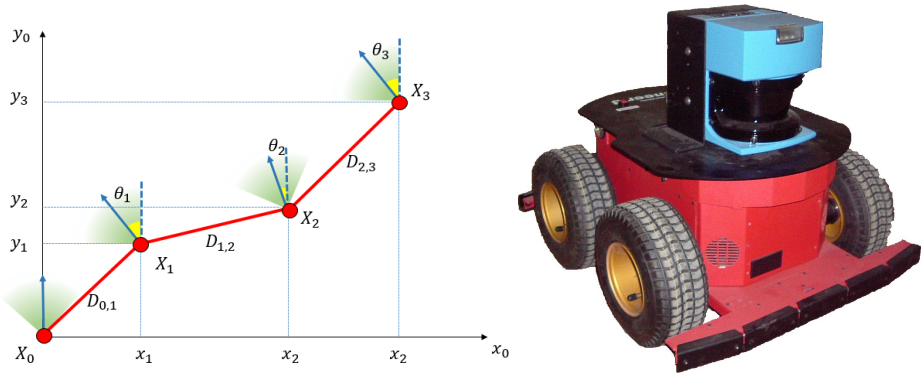
$$B_i = \sum_{j=0, j \neq i}^{n} C_{i,j}^{-1} \overline{D}_{i,j}$$

Per quando scritto la stima della posizione lineare ottima si limita ad un sistema di equazioni lineari:

$$\mathbf{GX} = \mathbf{B}$$

Per lo scopo di questa tesi però non è possibile limitare il modello al caso lineare a due gradi di libertà (x, y). Lu e Milios propongono una soluzione per i sistemi a tre gradi di libertà $(x, y, \theta)^T$, come per esempio può essere un robot, rappresentato nell'immagine 2.3, in grado di muoversi lungo un piano nella scena con un sensore che non ha la possibilità di acquisire la scena a 360^o, e con gli angoli di rollio e beccheggio fissi.

Nella ricostruzione di scene 3D, anche avere 3 gradi di libertà risulta un vincolo. Una qualità importante di un dispositivo utilizzato per la ricostruzione di



(a) (b)

Figura 2.3: Modello Graph-Based di un sistema a 3 gradi di libertà (a), e laser scanner con campo di acquisizione orizzontale limitato (b).

.

una scena è la possibilità di spostarsi in essa senza imporre vincoli sul movimento, poiché si possono creare delle zone d'ombra nell'acquisizione perdendo così dettagli della scena. Una zona d'ombra è un'area scura proiettata su una superficie da un corpo che, interponendosi tra la superficie stessa e l'emettitore, impedisce il passaggio della luce e quindi l'acquisizione. Per limitare le zone d'ombra l'utente deve essere libero di muoversi a suo piacimento nella scena. Si ha allora la necessità di estendere questo metodo da 3 $(x, y, \theta)^T$ a 6 $(x, y, z, \theta_x, \theta_y, \theta_z)$ gradi di libertà.

L'estensione del LUM a 6 gradi di libertà è descritta in [23]. Anche con l'utilizzo di 6 gradi di libertà la soluzione risulta essere un sistema di equazioni lineari $\mathbf{GX} = \mathbf{B}$ dove G è una matrice simmetrica positiva $6n \times 6n$. L'introduzione di ulteriori tre gradi di libertà rende la soluzione computazionalmente più complessa.

Capitolo 3

Registrazione di nuvole di punti

In questo capitolo si approfondisce la descrizione sullo SLAM Graph-Based, riportata nel capitolo 2, analizzando la registrazione. La trattazione propone i tre principali metodi utilizzati per l'allineamento di nuvole di punti: registrazione manuale, con Iterative Closest Point e features based. Dopo una loro descrizione generale, viene posta l'attenzione sulle fasi che caratterizzano l'ICP e per ognuna di esse vengono presentati algoritmi differenti. Sono infine ampiamente descritti i metodi per l'estrazione e la descrizione di punti di interesse 2D e 3D, fondamentali per la registrazione features based.

Il problema dell'allineamento di due o più nuvole di punti acquisite nello stesso ambiente e con diversi punti di vista, è conosciuto sotto il nome di *registrazione*. Lo scopo della registrazione è quello di stimare posizione e orientazione di viste acquisite separatamente, in un sistema di coordinate globali, di allinearle e mescolarle insieme in una singola nuvola di punti in modo che le aree di intersezione si sovrappongano perfettamente. In figura 3.1 è rappresentato un possibile esempio di registrazione.

La registrazione di nuvole di punti è fondamentale per uno svariato numero di applicazioni relativi alla computer vision: navigazione robotica in un ambiente non strutturato, fotogrammetria, archeologia digitale, architettura e in generale ogni volta che si vuole ricostruire una scena registrando più scansioni singole. Questa ampia utilizzazione motiva una ricerca intensa negli anni passati, con lo scopo di ottenere un allineamento accurato per nuvole di punti 3D e di rendere la registrazione un processo sempre più automatico.

Una nuvola di punti è una serie di dati strutturati P usata per rappresentare un gruppo di punti multidimensionali $p \in R^n$. In una nuvola di punti 3D, gli elementi sono costituiti principalmente da un vettore di tre coordinate geometriche X, Y e Z. Se sono presenti ulteriori informazioni, come ad esempio il colore, informazioni

(a) Prima nuvola di punti

(b) Seconda nuvola di punti

(c) Nuvola di punti registrata

Figura 3.1: Esempio di Registrazione

sulla normale locale della superficie o la curvatura, i punti $p \in P$ sono costituiti da un vettore con più elementi.

Data una nuvola di punti P con punti $p \in P$, e una nuvola Q con punti $q \in Q$, il problema della registrazione risiede nel trovare corrispondenze tra P e Q, stimare una trasformazione T che, quando applicata a P, allinea tutte le coppie di punti $(p_i \in P, q_i \in Q)$. Queste corrispondenze sono per la maggior parte delle volte non conosciute, necessitano quindi di essere determinate mediante un algoritmo. Per allineare due nuvole di punti in modo univoco bastano tre corrispondenze. Date un numero di corrispondenze maggiore di tre, ci sono differenti metodi, per calcolare la trasformazione ottima, che minimizzano un errore metrico.

Nelle prossime sezioni vengono ampiamente sviluppati i tre principali metodi per l'allineamento di nuvole di punti: la registrazione manuale, l'Iterative Closest Point, e la registrazione basata su features. Durante la descrizione di ogni singolo algoritmo viene anche riportato il nome della funzione che ne permette l'utilizzo

nelle librerie PCL, tramite la simbologia ($\rightarrow funzione$).

3.1 Registrazione Manuale

In questi ultimi anni la necessità di allineare nuvole di punti ha reso possibile lo sviluppo di algoritmi che rendono la registrazione un processo più veloce e automatico. Ancora oggi, però, la registrazione manuale continua ad essere utilizzata quando le nuvole da allineare risultano poche. Se invece, come nel caso studiato, devono essere allineate decine di nuvole di punti ne è sconsigliato l'utilizzo, poiché l'intero processo impiegherebbe molto tempo. Un corretto allineamento può essere ricavato principalmente in due modi.

Il primo consiste in traslazioni e rotazioni successive rispetto ad un sistema di riferimento. In questo caso l'utente deve impostare la lunghezza (o l'angolo), l'asse e il verso su cui applicare la trasformazione. Il sistema di riferimento può essere globale, cioè comune a tutte le nuvole, o locale, proprio della nuvola trasformata. Questo primo processo risulta essere poco efficace e per questo poco utilizzato.

Il secondo metodo per la registrazione manuale consiste nell'impostare manualmente dei legami tra punti di due diverse nuvole che rappresentano lo stesso punto reale di una scena. Questi legami sono detti corrispondenze. L'utente può identificare i punti che determinano la corrispondenza indicando l'indice che definisce il punto nella nuvola o tramite "click diretti" del mouse sui punti delle nuvole mostrate dal visualizzatore. Come già accennato, per un corretto allineamento, sono necessarie almeno tre corrispondenze e, minimizzando la distanza tra esse, viene ricavata la matrice di rototraslazione cercata.

La registrazione manuale risulta essere un processo lento e scomodo poiché necessita di un operatore che applichi le trasformazioni corrette o imposti delle corrispondenze tra punti di due diverse nuvole. In questi ultimi anni la ricerca dell'automatizzazione del processo ha portato allo sviluppo di diversi algoritmi che offrono una soluzione al problema. Possono essere distinte due categorie che definiscono la struttura algoritmica per la registrazione:

1. Algoritmi di registrazione Iterativi o Iterative Closest Point (ICP).
2. Algoritmi di registrazione basati su Features.

3.2 Iterative Closest Point

L'Iterative Closest Point (ICP) è uno strumento di registrazione presentato da Besl [3] nel 1992. Questo algoritmo permette di trovare iterativamente una matrice di rototraslazione ottima tra due nuvole di punti $p_i \in P$ e $q_j \in Q$ minimizzando l'errore tra le aree sovrapposte.

La struttura dell'algoritmo Iterative Closest Point può essere suddivisa in quattro fasi:

1. *Selection*: estrarre un sottoinsieme di punti dalle due nuvole.
2. *Correspondence Estimation*: calcolare le corrispondenze tra P e Q.
3. *Correspondence Rejection*: scartare le corrispondenze inesatte.
4. *Trasformation Estimation*: calcolare una trasformazione che minimizza la distanza tra i punti corrispondenti.

Mentre la prima fase viene effettuata un'unica volta durante lo sviluppo dell'algoritmo, le ultime tre vengono ripetute iterativamente. Il risultato tipicamente converge ad una trasformazione ottima desiderata.

Sotto varie ipotesi, come una perfetta sovrapposizione, l'allineamento converge ad un minimo globale. Il lato negativo dell'ICP è che se le nuvole di punti sono solo parzialmente sovrapponibili, è possibile che l'algoritmo converga ad un minimo locale inesatto portando così ad un allineamento non corretto. Questo accade soprattutto se le due nuvole sono state acquisite in posizioni molto diverse. Per questo sono necessari metodi per scartare le false corrispondenze in modo da evitare minimi locali e migliorare la convergenza. Inoltre se la nuvola è già allineata in modo grezzo, l'ICP è un algoritmo efficiente e robusto per raffinare l'allineamento tra nuvole di punti.

Input: Due nuvole di punti: $P = \{p_i\}$ e $Q = \{q_i\}$
Una traslazione iniziale: T_0.
Output: La trasformazione corretta, T, che allinea P con Q.

```
1  T ← T_0;
2  P* = Selection_point_from_P
3  Q* = Selection_point_from_Q
4  while non converge do
5  |  for i ← 1 to N do
6  |  |  m_i ← Corrispondence_Estimation_per_q_i*;
7  |  |  if Reject_Corrispondence(m_i, q_i*) then
8  |  |  |  w_i ← 0;
9  |  |  else
10 |  |  |  w_i ← 1;
11 |  |  end
12 |  end
13 |  T ← Transformation_Estimation;
14 end
```

Figura 3.2: Struttura dell'ICP.

L'algoritmo in figura 3.2 mostra il codice semplificato di un ICP. In input si hanno due nuvole di punti: $P = \{p_i\}$, $Q = \{q_j\}$ acquisite in uno stesso ambiente. La funzione restituisce un output che risulta la corretta T, che allinea la nuvola di punti P con Q.

In figura 3.2, $m_i \in P^*$ e (m_i, q_i^*) è una coppie di corrispondenza tra le nuvole P^* e Q^*. Il valore w_i può essere usato per dare un peso d'importanza alle coppie, in questo caso assume valore 1 se la corrispondenza trovata è corretta, 0 altrimenti.

Per ognuna delle fasi presentate in precedenza è necessaria la scelta di un metodo. Vi sono infatti differenti algoritmi per ogni fase che permettono diverse combinazioni di ICP. Si presentano ora le quattro fasi che caratterizzano l'ICP e i più importanti algoritmi per ognuna di esse.

3.2.1 Selection

In base alla applicazione ed al sensore utilizzato, le nuvole di punti possono contenere molti punti. Calcolare le corrispondenze per ogni punto, valutarne la correttezza e ricavare da esse una trasformazione ottima può diventare un calcolo computazionalmente oneroso. Inoltre le informazioni sono spesso ridondanti o non necessarie per la registrazione. Per questo, registrare solamente un sottoinsieme di punti delle nuvole originali può raggiungere risultati simili con un carico computazionale minore. In questa sezione saranno presentati alcuni metodi per estrarre un sottoinsieme limitato di punti rappresentativi della scena.

Filtraggio ROI (Region Of Interest)

Non sempre tutti i punti acquisiti da una scena sono utili per una registrazione. Se per esempio la scena acquisita risulta essere un'intera stanza ma ciò che interessa è la ricostruzione di un particolare oggetto, può risultare utile ridurre la nuvola in modo da avere in essa unicamente i punti che lo identificano. L'utente può eliminare una serie di punti non interessanti che risultano essere all'esterno di un volume di interesse, definito da tre intervalli, uno per ogni asse x, y e z di un sistema di riferimento, i cui estremi possono essere impostati.

É importante notare che, per una nuvola di punti proiettabile, risulta possibile applicare lo stesso ragionamento, utilizzando una regione di interesse bidimensionale, sui pixel della camera definiti dalla coppia (u, v), presentata nella sezione 1.3.1.

Filtraggio Outliers

Gli outliers sono punti nel cui intorno non sono presenti altri punti. È possibile che, nella nuvola acquisita dal Kinect V2, anche se con una minor probabilità rispetto ad una scansione laser, siano presenti punti che non rappresentano l'ambiente reale poiché sono creati da errori di misura. Questi, detti appunto outliers, sporcano le caratteristiche locali dei punti, come le normali di superficie o i cambi di curvatura, e causano insuccessi nella registrazione. Molte di queste irregolarità possono essere risolte tramite un'analisi statica dell'intorno di un punto, scegliendo, tramite diversi criteri, l'esclusione del punto stesso.

Il filtro *Statistical Outlier Removal* [24], in figura 3.3, è basato sul calcolo della distribuzione delle distanze tra il punto e i suoi vicini. Per ogni punto p_i, l'algoritmo calcola la distanza tra il punto stesso e i suoi k punti vicini. Assumendo che il risultato sia una distribuzione Gaussiana con una media e una deviazione standard, tutti i punti la cui distanza media non risulta compresa in un intervallo definito da una distanza globale media e una deviazione standard soglia, possono essere considerati outliers e cancellati quindi dalla nuvola di punti. Il valore di distanza globale media è calcolato sommando le distanze medie di ogni punto tra esso e i suoi vicini, e dividendo per il numero di punti della nuvola. Il parametro impostabile dall'utente, che risulta essere decisivo per la definizione degli outliers, è il valore della deviazione standard soglia. ($\rightarrow StatisticalOutlierRemoval$)

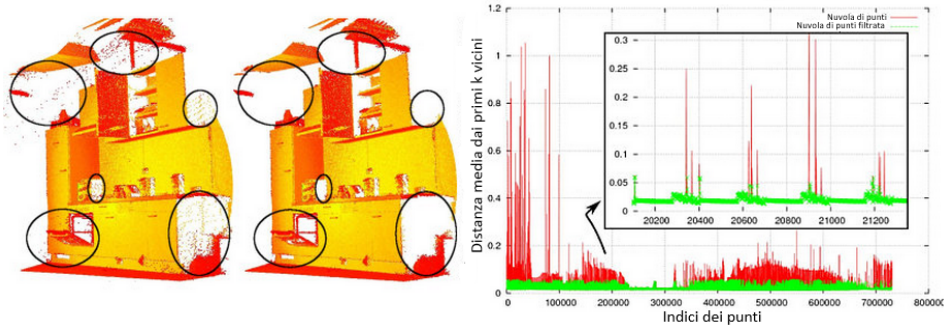


Figura 3.3: Metodo Statistical Outlier Removal.

Un altro filtro spesso usato è il filtro *Radius Outlier Removal* [25]. L'utente specifica un numero di vicini che ogni punto deve avere entro un certo raggio per rimanere nella nuvola di punti. L'immagine 3.4 aiuta a visualizzare come agisce il filtro e come cambi il risultato al variare dei parametri. Se per esempio è richiesto un solo vicino per far rimanere il punto nella nuvola, allora solo il punto giallo verrà rimosso. Se i vicini richiesti sono 2, sia il giallo che il verde saranno rimossi. ($\rightarrow RadiusOutlierRemoval$)

Nelle figure 3.5 è possibile visualizzare le immagine filtrate con questi algoritmi tramite il software scritto durante la tesi. È chiaro come i parametri inseriti modifichino drasticamente il risultato, si è quindi lasciata la possibilità di cambiare questi parametri nel programma. All'utente rimane la difficoltà di stimare tali valori.

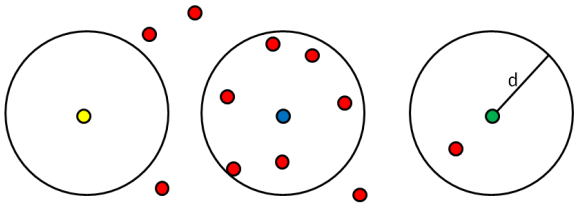
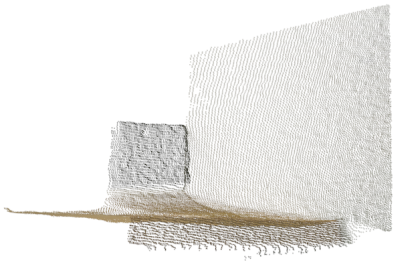


Figura 3.4: Metodo Radius Outlier Removal.

(a) Nuvola di punti.

(b) Nuvola filtrata con Statistical OutlierRemoval.



(c) Nuvola filtrata con Radius Outlier Removal.

Figura 3.5: Confronto tra metodi di Filtraggio Outliers.

Campionamento

Il campionamento è un metodo per selezionare un campione rappresentativo di punti della nuvola. Nella letteratura esistono vari metodi per campionare i punti che dovranno essere studiati nelle fasi successive dell'ICP. É necessario che il metodo utilizzato sia appropriato per la particolare condizione in esame, come il tipo di oggetto, presenza di occlusioni e presenza di rumore.

La ragione dell'impiego di questo metodo è ridurre il numero di punti aumentando le prestazioni dell'ICP, in termini di velocità. Riducendo il numero di punti, infatti, viene ridotto il numero di corrispondenze possibili usate per registrare le nuvole. Il trade-off tra campionamento e numero di iterazioni può migliorare di molto i risultati. Le tipologie di campionamento sono diverse:

Utilizzo di tutti i punti [3]: questo metodo tiene conto di tutti i punti in qualsiasi condizione. In situazioni dove esistono complesse variazioni di superfici è utile trattenere tutti punti, in modo da rilevare ogni singolo cambiamento di forma. In questo caso, però, la ricerca dei punti vicini, e quindi delle corrispondenze, può rivelarsi troppo ampia e computazionalmente costosa. Molti punti, come quelli appartenenti ad un piano, potrebbero introdurre informazioni che possono risultare ridondanti e dunque non necessarie per un corretto allineamento.

Campionamento Uniforme [26]: Tramite questo campionamento viene creata una griglia di piccoli parallelepipedi, chiamati anche voxel, in modo che in essa sia immersa la nuvola di punti considerata. Poi, in ogni voxel, tutti i punti presenti vengono approssimati con il centroide del parallelepipedo. Anche se questo approccio risulta essere poco più lento, viene utilizzato il centroide, e non il centro del voxel, poiché rappresenta la superficie in maniera più accurata. ($\rightarrow VoxelGrid$)

Campionamento Random [27]: il metodo consiste, come dice il nome, nell'applicare un campionamento random tra i punti della nuvola. Come dimostrato in [28], i campionamenti uniforme e random hanno le stesse prestazioni. L'unico vantaggio teorico di quest'ultimo è forzare il metodo correspondence estimation a scegliere coppie diverse per ogni iterazione, il che riduce la probabilità di scegliere la stessa incorretta corrispondenza, evitando di portare la convergenza ad un minimo locale. ($\rightarrow RandomSample$)

Normal Space Sampling [28]: Rusinkiewicz utilizza un criterio di selezione dei punti che predilige le aree della nuvola con dettagli chiamato normal space sampling. Le aree della nuvola 3D che hanno maggior dettagli contengono anche normali con direzioni differenti; perciò queste possono essere usate come una guida per controllare il numero di campionamenti per una certa area. Se dunque le normali dei punti in tale area risultano essere simili, come in un piano, il

campionamento sarà meno fitto. Al contrario se le direzioni delle normali risultano essere differenti, il campionamento sarà più fitto, poiché nell'area saranno presenti forme geometriche particolari. Nuvole con pochi elementi descrittivi possono essere comunque registrate se questi non vengono persi durante il campionamento. ($\rightarrow NormalSpaceSampling$)

Covariance sampling : il metodo seleziona i punti in base a quanto la nuvola risultante possa essere stabile per la registrazione tramite ICP con una copia di se stessa. Come dimostrato in [29] è possibile infatti determinare la stabilità di una forma, e quindi di regioni di nuvole, analizzando gli autovalori e gli autovettori della matrice di covarianza derivata dai punti appartenenti ad essa. ($\rightarrow CovarianceSampling$)

Nella figura 3.6 è possibile visualizzare le immagini filtrate con questi algoritmi tramite il software implementato durante la tesi. Per poter confrontare al meglio i metodi di campionamento, le nuvole di punti filtrate presentano un numero di punti uguale tra loro e circa dimezzato rispetto alla nuvola in ingresso.

Pur avendo un numero di punti uguali le nuvole sono notevolmente differenti. Si può notare come nel campionamento uniforme ci sia una certo ordine nella disposizione dei punti, cosa che invece non avviene nel campionamento random. I punti sul piano verticale del Normal Space Sampling sono molto più radi, poiché le direzioni delle normali dei punti appartenenti ad esso sono simili.

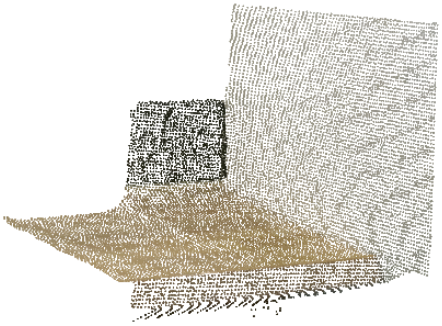
3.2.2 Correspondence Estimation

La Correspondence Estimation è il processo che accoppia ogni punto p_i appartenente alla nuvola P con un rispettivo punto q_j appartenente alla nuvola Q. Una corrispondenza sarà definita da una coppia di indici (i, j) che indicano una coppia di punti (p_i, q_j). Il metodo più utilizzato per il calcolo delle corrispondenze è quello di portare a termine una ricerca esaustiva attraverso tutti i punti di Q del punto più vicino ad ogni singolo punto di P. Questo algoritmo però è computazionalmente dispendioso se la nuvola è formata da molti punti. Per questo sono state implementate strutture per la ricerca rapida chiamate kd-tree [30] e octree [31].

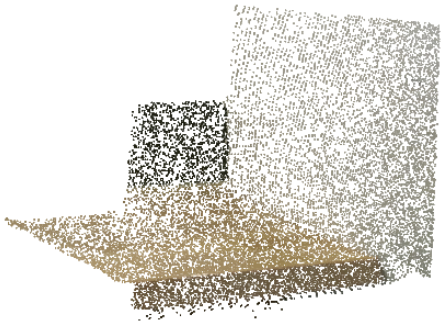
K-d tree

Il k-d tree, in figura 3.7, è una struttura di dati usata per organizzare un certo numero di punti in uno spazio k-dimensionale che, per il nostro scopo, cioè la ricerca di punti in una nuvola, verrà trattata nel caso tridimensionale. Si può pensare al k-d tree come a un albero binario in cui ogni livello divide i punti

(a) Nuvola di punti in ingresso.

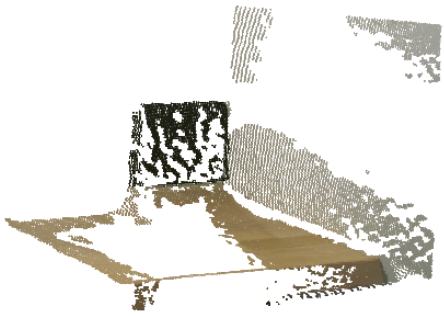


(b) Nuvola con campionamento Uniforme, Voxelgrid.



(c) Nuvola con campionamento Random.

(d) Nuvola campionate con Normal Space Sampling.



(e) Nuvola campionata con Covariance Sampling

Figura 3.6: Confronto tra metodi di campionamento.

lungo un piano allineato agli assi. Nel caso in cui $k > 3$ la divisione è effettuata tramite un iperpiano.

Il piano scelto per la divisione è la mediana dei punti del sottoalbero rispetto alle coordinate sull'asse per generare il piano separatore. Questo equivale a ordina-

re i punti in ordine crescente rispetto alla coordinata scelta, e posizionare la metà inferiore dei punti nel sottoalbero di sinistra e la metà superiore nel sottoalbero di destra. Avendo scelto la mediana, i due sottoalberi conterranno esattamente lo stesso numero di punti, garantendo che l'albero finale sia bilanciato, cioè con foglie a circa la stessa profondità. Un albero bilanciato garantisce una profondità logaritmica e quindi anche tempi per l'attraversamento logaritmici. Detto n il numero di punti della nuvola, il carico computazionale del k-d tree è $O(n \log n)$. $(\rightarrow KdTree)$

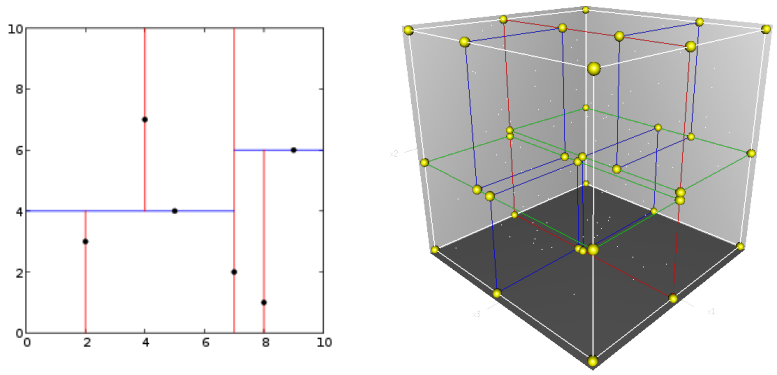


Figura 3.7: Esempi di K-d tree: bidimensionale (a) e tridimensionale (b).

Octrees

L'octree, in figura 3.8, è utilizzato per organizzare un certo numero di punti contenuti in uno spazio 3-dimensionale. L'octree è costituito da un albero binario i cui nodi rappresentano un cubo. Lo spazio di ogni nodo può essere diviso in otto cubi equivalenti, tramite dei piani paralleli agli assi x, y e z passanti per il centro del cubo stesso. Anche in questo caso la struttura di ricerca dei vicini si limita allora ad una ricerca in un albero binario. Esistono due metodi principali per costruire un octree:

1. *Capacità limitata:* ogni nodo può contenere al più un numero massimo di punti. Se quindi in un nodo vi sono più punti di quanto permetta il valore soglia questo viene diviso in otto cubi, nella modalità già descritta. In questo modo le foglie contengono un numero di punti inferiore al valore prefissato, ma non si ha garanzia sulla profondità della struttura ad albero dell'octree.

2. *Dimensione limitata*: ogni nodo interno non può avere il lato più piccolo di un valore prefissato. L'algoritmo si ferma se il nodo ha lato minore del valore

fissato o non contiene punti, altrimenti si procede ricorsivamente. Questa strategia è quella implementata nelle PCL.

Anche se il tempo per formare un octree è inferiore rispetto al k-d tree, grazie alla strategia triviale di creazione, la struttura non risulta bilanciata, la sua profondità infatti varia a seconda della distribuzione dei punti nella nuvola.

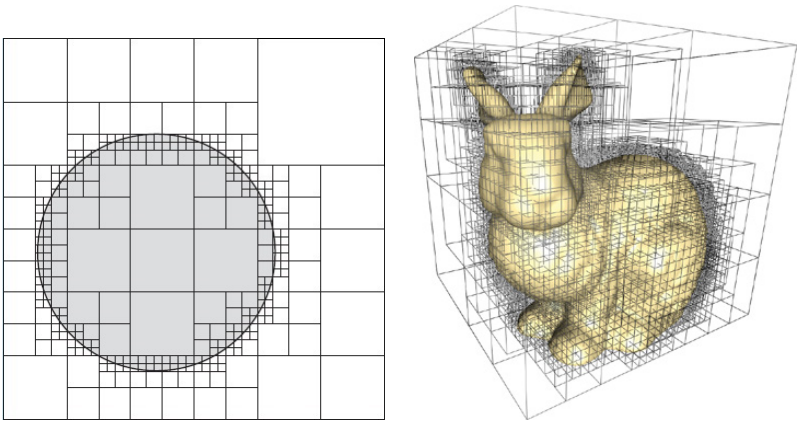


Figura 3.8: Esempio di Octree (b) e una sua rappresentazione bidimensionale (a).

Dopo aver descritto due diverse strutture per l'organizzazione dei punti e dunque per la ricerca rapida di corrispondenze, spostiamo l'attenzione sugli algoritmi per la stima delle corrispondenze. Nelle PCL sono implementate delle funzioni diverse per effettuare la fase di correspondence estimation, le quali verranno ora presentate.

Correspondence Estimation Base
Questo algoritmo consiste nel costruire un sistema ad albero impostabile, octree o k-d tree, e cercare il punto q_i appartenente alla nuvola target Q che ha distanza minima dal punto p_j della nuvola source P. ($\rightarrow Correspondence Estimation$)

Correspondence Estimation Back Projection
Dopo aver costruito un sistema ad albero, si cercano i k punti vicini q_j, appartenenti alla nuvola target Q, al punto p_i della nuvola source P. Per ognuno di questi punti si calcola il coseno dell'angolo θ tra la normale del punto stesso e la normale del punto in esame p_i:

$$cos(\theta) = normal_x(p_i) * normal_x(q_j) + normal_y(p_i) * normal_y(q_j) + normal_z(p_i) * normal_z(q_j)$$

dove $normal_x(p_i)$ identifica la normale in direzione dell'asse x del punto p_i. Si calcola poi la distanza tenendo conto della distanza spaziale punto-punto e della compatibilità della direzione della normale.

$$dist = \sqrt{(x(p_i) - x(q_j))^2 + (y(p_i) - y(q_j))^2 + (z(p_i) - z(q_j))^2} * (2 - (cos(\theta))^2$$

dove $x(p_i)$ è la coordinata x del punto p_i in un sistema di riferimento comune alle due nuvole di punti. Si imposta infine la corrispondenza tra p_i e il punto che ha distanza descritta minore da esso. Con questo algoritmo si tiene conto non solo della distanza euclidea ma anche della somiglianza della normale alla superficie in quei punti. ($\rightarrow CorrespondenceEstimationBackProjection$)

Correspondence Estimation Normal Shooting
Anche con questo algoritmo si calcola un insieme K composto dai k punti vicini q_j, appartenenti alla nuvola target Q, al punto p_i della nuvola source P. Per ogni punto di q_j appartenente a K si calcola una matrice V (1×3) :

$$pt_x = x(q_j) - x(p_i)$$
$$pt_y = y(q_j) - y(p_i)$$
$$pt_z = z(q_j) - z(p_i)$$
$$V = (pt_x, pt_y, pt_z)$$

ed una matrice $N(normal_x(p_i), normal_y(p_i), normal_z(p_i))$ relativa al punto p_i. Ora viene calcolata una matrice 3×1, C attraverso il prodotto vettoriale dei due vettori precedenti, $C = N \times V$. Infine si calcola la distanza utilizzando il valore assoluto di C e si imposta la corrispondenza tra il punto $q_j \in K$, che ha distanza da p_i minore, e il punto p_i stesso. Grazie alla geometria lineare, si è impostata una corrispondenza tra il punto p_i e il punti q_j nella nuvola target che ha una distanza perpendicolare minima alla retta che passa per il punto p_i e ha direzione della normale calcolata in esso. ($\rightarrow CorrespondenceEstimationNormalShooting$)

Correspondence Estimation Organized Projection
Questo algoritmo utilizza i parametri intrinseci ed estrinseci del sensore per proiettare la nuvola di punti source nella nuvola di punti target. É possibile trovare il

punto q_j più vicino a p_i, e quindi impostare la corrispondenza, tramite una semplice approssimazione dovuta alla proiezione. Anche se il risultato non è preciso come nei precedenti, è molto più veloce ed evita di utilizzare strutture ad albero addizionali. Per questo metodo è necessario che la nuvola di punti target sia organizzata. Una nuvola di punti organizzata ha una struttura simile ad un'immagine (o matrice), in cui i dati sono divisi in righe e colonne. In tale nuvola i vicini di un punto possono essere trovati utilizzando gli indici della struttura matriciale. ($\rightarrow CorrespondenceEstimationOrganizedProjection$)

Infine si sottolinea la possibilità presente nei primi tre metodi descritti, di calcolare le corrispondenze reciproche iterando due volte lo stesso algoritmo, scambiando la nuvola source con la target (e viceversa) e verificando l'univocità delle corrispondenze. Questo approccio anticipa la fase della correspondence rejection poiché elimina tutte le corrispondenze i cui accoppiamenti non sono univoci, rendendo la registrazione più robusta.

3.2.3 Correspondence Rejection

La correspondence rejection ha lo scopo di filtrare le corrispondenze calcolate durante la fase precedente. É possibile infatti che tra di esse siano presenti corrispondenze errate. Una corrispondenza risulta errata se lega punti che non si riferiscono allo stesso elemento della scena. L'utilizzo di queste corrispondenze per la stima della matrice di rototraslazione tra le nuvole può compromettere la buona riuscita dell'allineamento. Risulta necessario quindi eliminare le corrispondenze errate prima che introducano un errore nella fase successiva. Gli algoritmi di questa fase non sono mutualmente esclusivi ma è possibile utilizzarli in serie. Saranno presentati ora i metodi più utilizzati implementati nelle PCL.

Correspondence rejection basato sulla distanza. Questo metodo filtra tutte le coppie di punti che hanno una distanza Euclidea tra loro maggiore di una data soglia, come suggerito in [28]. ($\rightarrow CorrespondenceRejectorDistance$)

Correspondence rejection basato sulla distanza media. Anche questo algoritmo, come il precedente, si basa sul calcolo di una distanza. La soglia in questo caso però non è fissa, ma viene calcolata in base alla mediana delle distanze tra i due set di corrispondenze. Questo approccio considera la distribuzione di distanze tra le coppie di punti e adatta la soglia in base a quanto le nuvole di punti sono vicine, più sono vicini, più la soglia sarà bassa. Si usa la mediana e non la media perché risulta essere più robusta ai punti isolati. ($\rightarrow CorrespondenceRejectorMedianDistance$)

Correspondence Rejection di coppie con target multipli. Con gli algoritmi presentati fino ad ora, ogni punto della prima nuvola ha una sola corrispondenza

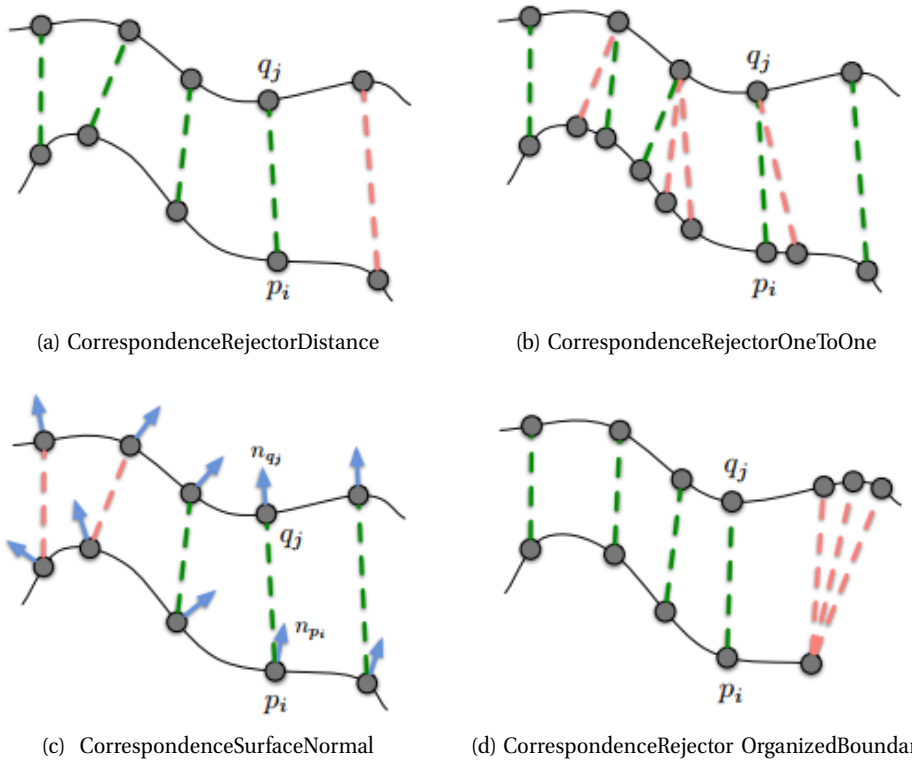
nella seconda. Questo però non implica che ogni punto della seconda debba avere una sola corrispondenza con la prima. É possibile, quindi, che siano presenti corrispondenze multiple dovute al fatto che le due nuvole non siano identiche. Con questo algoritmo vengono eliminate tutte le corrispondenze multiple in modo da avere un rapporto biunivoco, $1:1$, tra i punti, da cui il nome dell'algoritmo datogli dalle PCL. ($\rightarrow CorrespondenceRejectorOneToOne$)

Correspondence Rejection con RANSAC. Utilizza l'algoritmo RANSAC (RANdom SAmple Consensus) [32] per stimare la trasformazione per un dato set di corrispondenze. Poi elimina alcune di queste basandosi sulla distanza Euclidea tra i punti dopo l'applicazione della trasformazione calcolata sulla nuvola di punti sorgente. Questo algoritmo è efficace nel limitare la possibilità che l'allineamento converga ad un minimo locale, poiché ad ogni ciclo produce corrispondenze leggermente diverse. ($\rightarrow CorrespondenceRejectorSampleConsensus$)

Correspondence Rejection basato sulla compatibilità delle normali. Con questo algoritmo vengono eliminate le corrispondenze, relative ai punti le cui normali differiscono di un angolo maggiore di una certa soglia. In questo modo è possibile eliminare anche corrispondenze errate, giudicate corrette da altri algoritmi che usano la distanza fra i punti. Ovviamente il calcolo delle normali porta ad un onere computazionale maggiore.
($\rightarrow CorrespondenceRejectorSurfaceNormal$)

Correspondence Rejection basato sui bordi di superficie. Quando due nuvole di punti rappresentano superfici che sono parzialmente sovrapposte, accettare la corrispondenze tra punti sui bordi può introdurre errori. Per rilevare questi punti sui bordi, è possibile sfruttare la natura organizzata delle nuvole di punti ed eliminare le corrispondenze che contengono punti nei cui intorni siano presenti discontinuità di profondità. Muovendo una finestra lungo la mappa di profondità è possibile controllare se siano presenti al suo interno un numero di punti che abbiano profondità sufficientemente simile al punto centrale. ($\rightarrow CorrespondenceRejectorOrganizedBoundary$)

È possibile inoltre saltare questa fase, assumendo però che non esista nessun punto isolato e che tutte le corrispondenze siano corrette. Questa opzione infatti permette di tenere in considerazione tutte le corrispondenze trovate nella fase precedente per ricavare la trasformazione. Nella figura 3.9 sono presenti dei modelli grafici che rappresentano il funzionamento dei vari algoritmi di Correspondence Rejector.



(a) CorrespondenceRejectorDistance

(b) CorrespondenceRejectorOneToOne

(c) CorrespondenceSurfaceNormal

(d) CorrespondenceRejector OrganizedBoundary

Figura 3.9: Correspondence rejection: le "buone" corrispondenze in verde vengono mantenute mentre le "cattive" corrispondenze in rosso vengono eliminarte per migliorare la convergenza.

3.2.4 Transformation Estimation

La trasformation estimation è la stima di una trasformazione rigida $\mathbf{T}$ che minimizza l'errore tra le coppie di punti. In questi anni sono stati implementati numerosi metodi, ma esistono principalmente due errori metrici da minimizzare: point-to-point e point-to-plane.

$$E_{point-to-point}(\mathbf{T}) = \sum_{k=1}^{N} w_k(\|\mathbf{T}\ p_k - q_k\|)^2$$

$$E_{point-to-plane}(\mathbf{T}) = \sum_{k=1}^{N} w_k(\|\mathbf{T}p_k - q_k\| \cdot n_{qk})^2$$

dove (p_k, q_k) è la k-esima delle N coppie di corrispondenza tra la nuvola

source e la target e il valore opzionale w_k può essere usato per dare un peso d'importanza alle coppie.

Errore metrico standard point-to-point
L'errore metrico nell'algoritmo ICP standard, utilizzato per la prima volta da Arun [33] nel 1987, è il point-to-point. Esistono vari modi per minimizzare questo errore, le PCL utilizzano principalmente un metodo basato sulla decomposizione ai valori singolari (SVD) ($\rightarrow TransformationEstimationSVD$). Un altro metodo utilizzato per minimizzare l'errore point-to-point si basa sulla stima attraverso i metodi di Levenberg-Marquardt. ($\rightarrow TransformationEstimationLM$)

Errore metrico point-to-plane
Chen e Medioni [34] hanno introdotto l'errore metrico point-to-plane. Utilizza la distanza tra il punto p_k nella nuvola source e il piano descritto dal punto q_k, appartenente alla nuvola target, e la sua normale alla superficie n_{qk}. L'errore è minimizzato tramite algoritmi non lineari come Levemberg-Marquadt. ($\rightarrow TransformationEstimationPointToPlane$)

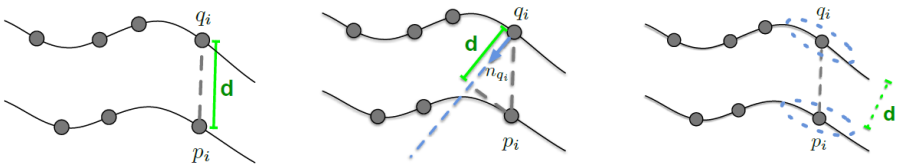
Errore ai minimi quadrati point-to-plane
Nelle PCL è implementato anche un metodo iterativo che accumula gli errori point-to-plane di tutte le corrispondenze in una matrice $\mathbf{A}$ e stima la rotazione e la traslazione risolvendo un sistema lineare della forma $\mathbf{A}^T\ \mathbf{A}v = \mathbf{A}^T b$, con v una vettore di sei elementi rappresentanti i parametri di trasformazione ottimali da trovare [35]. ($\rightarrow TransformationEstimationPointToPlaneLLS$)

Errore metrico pesato point-to-plane
É possibile migliorare la convergenza assegnando un coefficiente di peso ad ogni corrispondenza. I pesi delle coppie di punti possono essere visti come una ulteriore correspondence rejection, poiché modificano l'influenza delle corrispondenze nella fase di minimizzazione. Il peso può essere in funzione della distanza point-to-point o point-to-plane, in funzione dell'angolo tra le normali corrispondenti ai punti o in funzione di un modello di rumore del sensore che si sta utilizzando. Molte pubblicazioni considerano l'aggiunta di un peso nelle corrispondenze un beneficio per la robustezza e la convergenza dell'algoritmo trasformation estimation. Le PCL offrono una funzione per utilizzare l'errore pesato point-to-plane. ($\rightarrow TransformationEstimationPointToPlaneWeighted$))

Errore metrico Generalizzato
Segal [36] propone un errore metrico che generalizza gli errori point-to-point e point-to-plane. Il metodo usa le covarianze dell'intorno del punto ($\mathbf{\Sigma}_k^P$ e $\mathbf{\Sigma}_k^Q$) per allineare le superfici piuttosto che i punti stessi:

$$E_{Generalized-ICP}(\mathbf{T}) = \sum_{k=1}^{N} d_k^{(\mathbf{T})^T} (\mathbf{\Sigma}_k^Q + \mathbf{T}\mathbf{\Sigma}_k^P \mathbf{T}^T)^{-1} d_k^{(\mathbf{T})}$$

Dove $d_k^{(\mathbf{T})} = q_k - \mathbf{T}p_k$ è la distanza point-to-point delle corrispondenze (p_k, q_k), $\mathbf{T}$ è la matrice di trasformazione, e $\mathbf{T}^T$ la sua trasposizione. Nelle PCL, l'algoritmo Generalized-ICP è implementato come fosse una registrazione completa, include cioè il calcolo delle covarianze $\mathbf{\Sigma}_k^P$ e $\mathbf{\Sigma}_k^Q$ per tutti i punti della nuvola source P e la nuvola target Q. ($\rightarrow GeneralizedIterativeClosestPoint$)



(a) Errore Point to Point. (b) Errore Point to Plane. (c) ICP Generalized.

Figura 3.10: Metodi basati sulla minimizzazioni di errori.

In figura 3.10 vengono proposti i modelli degli algoritmi che si occupano della minimizzazione di diversi tipi di errori.

Per la grande importanza dell'algoritmo nella computer vision, nelle PCL sono già implementati diversi tipi di Iterative Closest Point che sfruttano le diverse tipologie di trasformation estimation. In ognuno di questi, l'utente può modificare il metodi di correspondence estimation e correspondence rejection permettendo svariate soluzione di personalizzazione; di default è impostato l'algoritmo correspondence estimation base, ovvero $CorrespondenceEstimation$.

3.2.5 I criteri di termine

Per un algoritmo di registrazione ricorsivo, devono essere necessariamente definiti dei criteri di terminazione dello stesso. Nell'ICP, oltre a impostare semplicemente un numero di iterazioni massimo, possono essere specificati diversi criteri. Nelle librerie PCL sono disponibili i seguenti.

Massimo numero di iterazioni

Questa soglia deve essere impostata in base alla complessità di registrazione; è ragionevole pensare che la registrazione di due nuvole di punti lontane tra loro necessiterà un maggior numero di iterazione rispetto a due nuvole discretamente preallineate. Da una certa iterazione k in poi, l'errore cessa di diminuire e converge ad un valore non nullo che dipende dalle caratteristiche delle nuvole. Impostare

unicamente questo criterio di termine ad un valore n, implica che l'algoritmo continua fino ad esattamente n iterazioni. Con $n < k$ l'allineamento non è ottimale, ma con $n > k$ si hanno $n - k$ iterazioni che aumentano inutilmente il tempo di processo. ($\rightarrow setMaximumIterations$)

Errore quadratico medio assoluto

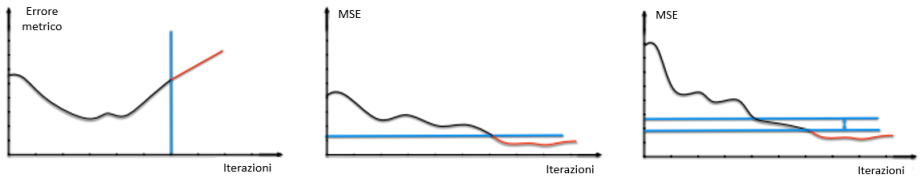
Non essendo a conoscenza del numero di iterazioni ottimali, si è pensato di ribaltare il problema. Con questo metodo si termina l'algoritmo quando l'errore tra due nuvole allineate risulta inferiore ad un certo valore soglia.

L'errore tra due nuvole di punti è calcolato utilizzando l'errore quadratico medio della somma delle distanze tra ogni punto della nuvola source e il punto più vicino della nuvola target. Sapere a priori un buon valore soglia risulta, però, difficile, poiché dipende dalle caratteristiche rappresentate in entrambe le nuvole. Per questo problema si è introdotto un ulteriore criterio di termine.
($\rightarrow setTransformationEpsilon$)

Errore quadratico medio relativo

Questo metodo non utilizza un valore soglia assoluto dell'errore, poiché la definizione stessa di questo risulta difficile, bensì un valore soglia tra la differenza degli errori risultanti da due iterazioni consecutive. Sia e_i l'errore di allineamento in uscita dell'i-esima iterazione, e_{i+1} quello in uscita dell'i+1-esima iterazione e sia h un valore soglia fissato. L'operazione viene terminata se risulta $e_i - e_{i+1} < h$.
($\rightarrow setEuclideanFitnessEpsilon$)

In figura 3.11 vengono rappresentati i criteri di termine dell'ICP.



(a) Massimo numero di iterazioni.

(b) Errore quadratico medio assoluto.

(c) Errore quadratico medio relativo.

Figura 3.11: Criteri di termine dell'ICP.

3.3 Registrazione Features Based

Gli algoritmi ICP, per loro natura, richiedono un allineamento iniziale per evitare la convergenza ad un minimo locale inesatto. Una soluzione a questo problema può essere trovata utilizzando più volte ricorsivamente lo stesso algoritmo ICP o

calcolando una trasformazione iniziale da corrispondenze impostate dall'utente. Entrambi i metodi però risultano essere troppo lenti per la maggior parte delle applicazioni.

In questi ultimi anni, per trovare una soluzione al problema, si sono ripresi concetti utilizzati per la registrazione di immagini 2D. Per un buon allineamento non sono necessarie tante corrispondenze ma buone corrispondenze. Bastano infatti tre corrispondenze corrette per stimare la matrice di rototraslazione da applicare.

Nella registrazione bidimensionale si utilizzano algoritmi che estraggono dalle immagini una serie di particolari pixel identificativi della scena acquisita, basandosi sulla variazione dell'intensità luminosa attorno ad esso. Da questi pixel di interesse, chiamati keypoints, vengono estratte delle informazioni che ne descrivono più dettagliatamente l'intorno. Gli elementi descrittivi delle due scene, chiamati features, vengono comparati tra loro e vengono calcolate delle corrispondenze tra keypoints aventi un intorno simile. Questo metodo di registrazione è chiamato features based, poiché si basa su confronti tra features.

La stessa struttura algoritmica può essere utilizzata, a meno di opportune modifiche, per la registrazione di nuvole di punti 3D. La registrazione features based può essere schematizzata in alcuni passi:

1. *Keypoints estimation*: da entrambe le nuvole di punti, si identificano i punti di interesse che meglio rappresentano la scena;
2. *Feature descriptors estimation*: ogni keypoints viene descritto tramite un set di features;
3. *Correspondence estimation*: si stimano delle corrispondenza tra keypoints delle nuvole, basandosi sulla somiglianza tra i loro features;
4. *Correspondence rejection*: è necessaria una fase in cui vengono scartate le errate corrispondenze che contribuirebbero negativamente al processo di registrazione;
5. *Trasformation estimation*: dal rimanente set di corrispondenze corrette, si stima la trasformazione tra le due nuvole di punti.

Nell'immagine 3.12 è rappresentato lo schema features based. Mentre la fase di correspondence rejection e trasformation estimation sono analoghi all'ICP, la correspondence estimation è differente. Non si confrontano più le singole caratteristiche del punto ma, tramite i descrittori, gli intorni dei due punti.

Figura 3.12: Schema della registratione features based.

Per l'importanza di questo metodo di registrazione nei prossimi due paragrafi verranno presentati non solo i keypoints e descrittori 3D, ma anche keypoints e descrittori 2D, per completezza.

3.4 KeyPoints e Features 2D

Uno dei passi fondamentali della registrazione di immagini nel 2D o di nuvole di punti nel 3D consiste nell'identificare punti notevoli, chiamati keypoints, e descrivere essi tramite features, cioè una raccolta di dati che identificano il singolo keypoint. Quello che si vuole ottenere sono delle misure locali sparse che catturano e descrivono l'essenza della nuvola di punti, o dell'immagine, in ingresso.

Per raggiungere questo scopo è necessario che si soddisfino due importanti criteri:

- l'algoritmo per l'estrazione di features deve risultare *ripetibile e accurato*, in modo che le stesse caratteristiche siano estratte da due nuvole di punti, o immagini, che mostrano lo stesso oggetto.
- le features devono essere *distintive*, la descrizione di diversi dati in ingresso deve risultare differente.

In generale un algoritmo che determina il contenuto informativo dell'immagine è composto da due passi principali:

- Detection: rilevazione di caratteristiche puntuali o locali salienti nell' immagine.
- Description: descrizione delle caratteristiche trovate attraverso un descrittore robusto, compatto e identificativo.

In questo capitolo verranno presentati i principali algoritmi di detection e description nell'ambito 2D, cercando di far risaltare, oltre che la modalità con cui agiscono, anche le necessità che hanno portato a questo processo evolutivo nella Computer Vision.

Gli algoritmi presentati vengono definiti 2D poichè hanno come input un'immagine e non una nuvola di punti.

3.4.1 Harris corner detector

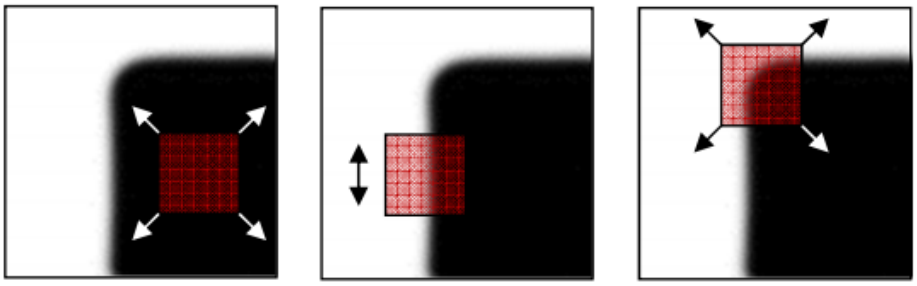
Vista l'enorme quantità di dati contenuti in un'immagine, è fondamentale avere a disposizione un algoritmo che a partire da questi dati riesca a riconoscere e selezionare solo quelli di interesse per lo scopo specifico che ci si è predisposti. Una regione d'immagine espressa in scala di grigi può essere classificata, come si può vedere nella figura 3.13, in base alla variazione d'intensità presente in essa:

- *Flat*: nella regione non vi sono presenti variazioni nell'intensità dell'immagine.
- *Edge*: nella regione si rilevano cambiamenti d'intensità significativi nella direzione ortogonale alle edge.
- *Corner*: nella regione si rilevano cambiamenti d'intensità significativi in tutte le direzioni.

L'algoritmo Harris corner detector rileva gli angoli nell'immagine, che diventeranno i keypoints che vogliamo estrarre nell'immagine stessa.

Sia I un'immagine espressa in scala di grigi. Siano inoltre u e v le coordinate sul piano corrispondenti rispettivamente alla direzione orizzontale e verticale. Volendo rilevare variazioni significative dell'immagine I, viene spontaneo esprimere la I stessa attraverso lo sviluppo di Taylor che per semplicità verrà troncato al primo ordine. Sia perciò x_0 un qualsiasi punto all'interno del dominio dell'immagine ed h un piccolo vettore spostamento, allora negli intorni di x_0 vale

$$I(x_0 + h) \approx I(x_0) + (\nabla I(x_0))^T h$$



(a) Flat (b) Edge (c) Corner

Figura 3.13: Classificazione di regioni d'immagine tramite la variazione d'intensità.

dove con $\nabla I(x_0)$ abbiamo indicato il gradiente dell'immagine nel punto x_0 cioè:

$$\nabla I(x_0) = \begin{bmatrix} \frac{\partial I}{\partial u}(x_0) \\ \frac{\partial I}{\partial u}(x_0) \end{bmatrix}$$

Si noti che l'immagine è descritta su di un dominio discreto, si ha quindi: $\frac{\partial I}{\partial u} = I_u(x_0) = I_u(u,v) = I(u+1,v) - I(u,v)$ e $\frac{\partial I}{\partial v} = I_v(x_0) = I_v(u,v) = I(u,v+1) - I(u,v)$. La variazione dell'intensità dell'immagine in un intorno di x_0 può perciò essere scritta come:

$$I(x_0 + h) - I(x_0) \approx (\nabla I(x_0))^T h$$

Per rilevare un corner non è interessante il segno di variazione, interessa soltanto che tale variazione sia considerevole. Consideriamo quindi il quadrato della funzione.

$$(I(x_0 + h) - I(x_0))^2 \approx ((\nabla I(x_0))^T h)^2 = h^T \nabla I(x_0)(\nabla I(x_0))^T h \tag{3.1}$$

L'immagine a nostra disposizione sarà un'immagine tipicamente affetta da rumore: sarà perciò opportuno stimare $(I(x_0 + h) - I(x_0))^2$ mediando i valori che si trovano per tale indice in un intorno Ω_{x_0} di x_0 stesso:

$$(I(x_0 + h) - I(x_0))^2 \approx \sum_{x \in \Omega_{x_0}} w(x - x_0)(I(x + h) - I(x))^2 \tag{3.2}$$

dove $w(x - x_0)$ è una funzione di peso, tipicamente una gaussiana. Si noti che l'equazione (3.1) continua a valere anche per un generico punto x:

$$(I(x + h) - I(x))^2 \approx ((\nabla I(x))^T h)^2 = h^T \nabla I(x)(\nabla I(x))^T h$$

perciò sostituendola in (3.2) otteniamo:

$$(I(x_0+h)-I(x_0))^2 \approx \sum_{x\in\Omega_{x0}} w(x-x_0)h^T\nabla I(x)(\nabla I(x))^T h = E(x_0).$$

Lo spostamento h non dipende dalla x, per cui l'equazione diventa:

$$E(x_0) = h^T \sum_{x\in\Omega_{x_0}} w(x-x_0)h^T\nabla I(x)(\nabla I(x))^T h = E(x_0)h = h^T C h$$

Infine notando che la sommatoria può essere anche portata all'interno della parentesi si ottiene:

$$C = \sum_{x\in\Omega_{x_0}} w(x-x_0)\nabla I(x)(\nabla I(x))^T = \sum_{x\in\Omega_{x_0}} w(x-x_0)\begin{bmatrix} I_u^2(x)) & I_u(x))I_v(x)) \\ I_u(x))I_v(x)) & I_v^2(x)) \end{bmatrix} =$$

$$= \begin{bmatrix} \sum_{x\in\Omega_{x_0}} w(x-x_0)I_u^2(x)) & \sum_{x\in\Omega_{x_0}} w(x-x_0)I_u(x))I_v(x)) \\ \sum_{x\in\Omega_{x_0}} w(x-x_0)I_u(x))I_v(x)) & \sum_{x\in\Omega_{x_0}} w(x-x_0)I_v^2(x)) \end{bmatrix}$$

La matrice C così ottenuta è simmetrica e semi-definita positiva, può perciò essere decomposta, tramite la scomposizione ai valori singolari, in:

$$C = USU^T = U\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} U^T$$

Ricordando l'interpretazione geometrica della scomposizione ai valori singolari questo ci dice che:

1. Quando $\sigma_1 \approx 0$ e $\sigma_2 \approx 0$ non si hanno variazioni di intensità, l'intorno di x_0 è allora classificabile di tipo flat.
2. Quando $\sigma_1 \gg 0$ e $\sigma_2 \approx 0$ allora nei dintorni di x_0 vi è una variazione d'intensità in direzione u_1, in prossimità di x_0 passerà allora un edge nella direzione ortogonale a u_1.
3. Quando $\sigma_1 \gg 0$ e $\sigma_2 \gg 0$ allora nei dintorni del punto si ha un corner poiché in entrambe le direzioni principali si ha una variazione sostanziale dell'intensità dell'immagine.

Per diminuire la complessità di calcolo al posto di applicare la scomposizione ai valori singolari è possibile determinare le proprietà dell'intorno del punto x_0 tramite la traccia e il determinante della matrice C.

$$R(x_0) = det(C(x_0)) - k \cdot tr^2(C(x_0))$$

dove k è un'opportuna costante, e sapendo che $trace(C(x_0)) = \sigma_1 + \sigma_2$ e $det(C(x_0)) = \sigma_1\sigma_2$ è semplice dimostrare che se $R(x_0) \gg 0$ si è in presenza di un corner mentre se $R(x_0) \ll 0$ si è in presenza di un edge.

3.4.2 SIFT: Scale Invariant Feature Transform

L'algoritmo SIFT (Scale Invariant Features Transform) è stato presentato per la prima volta nel 2002 da David G.Lowe [37]. Lo scopo è di riconoscere caratteristiche invarianti alla rotazione e a differenti scale, parzialmente invarianti al cambio di illuminazione e al punto di vista della telecamera. I passi principali dell'algoritmo di estrazione delle SIFT sono i seguenti:

- *Individuazione degli estremi locali nello scale-space*: si cercano punti interessanti su tutte le scale e posizioni dell'immagine utilizzando una funzione DoG (Difference of Gaussian). L'approccio utilizzato è quello del filtraggio in cascata (cascade filtering approach), che consente di determinare le posizioni e la scala delle feature candidate ad essere punti chiave e che, in un secondo momento, vengono caratterizzate con maggior dettaglio.

- *keypoints localization*: per ciascun punto candidato viene costruito un modello dettagliato per determinarne posizione e scala. I punti vengono inoltre selezionati secondo misure di stabilità.

- *Orientation assignment*: per ottenere l'invarianza rotazionale, viene associata una o più orientazioni per ogni keypoint, calcolate in base alla direzione del gradiente dell'immagine. Tutte le operazioni seguenti saranno eseguite tenendo conto delle informazioni dell'immagine trovate in precedenza, posizione, orientazione e scala per ogni keypoints.

- *keypoints descriptor*: a partire dai gradienti locali dell'immagine, alla scala selezionata e nell'intorno del punto chiave, viene costruito il descrittore.

La quantità di features è particolarmente importante per decretare l'appartenenza di una regione all'ambiente. I descrittori locali trovati con il SIFT permettono alle posizioni relative tra i features di cambiare significativamente con piccoli cambiamenti nel descrittore. Questo approccio non solo permette un accoppiamento affidabile anche in caso di distorsioni, ma è anche robusto alle variazioni del punto di vista della scena per superfici non piane.

Scale extrema detection

Il primo passo per il rilevamento di keypoints è quello di identificare le posizioni e le scale che possono essere assegnate per diversi punti di vista dello stesso oggetto. Lo scale-space è una rappresentazione multiscala di un'immagine. È ottenuta attraverso la convoluzione del segnale con una famiglia di kernel gaussiano con deviazione standard σ variabile.

Lo scales-pace di un'immagine è definita come una funzione, $L(x, y, \sigma)$, che è prodotta dalla convoluzione di una Gaussiana con scala variabile, $G(x, y, \sigma)$, con un'immagine, $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Dove $*$ è l'operatore di convoluzione in x e y, e:

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

In questo modo si ha un filtro gaussiano con σ variabile che proietta l'immagine $I(x, y)$ in uno spazio tridimensionale $L(x, y, \sigma)$.

Per trovare dei punti interessanti nello scale-space viene utilizzata la funzione differenza di gaussiane, o DoG (difference of gaussian), che si calcola come differenza di due funzioni scale-space separati da un fattore moltiplicativo k della scala:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Ci sono diverse ragioni che supportano l'utilizzo di questa funzione. Per prima cosa la funzione DoG è molto veloce da calcolare a partire dallo scale-space, basta effettuare la differenza tra immagini convolute vicine tra loro. Inoltre, la DoG fornisce una buona approssimazione del Laplaciano della Gaussiana normalizzato rispetto al fattore di scala $\sigma^2\nabla^2 G$. La normalizzazione del Laplaciano con il fattore σ^2 è richiesta per l'invarianza al fattore di scala. In seguito, Mikolajczyk [38] ha dimostrato che i massimi e minimi di $\sigma^2\nabla^2 G$ producono features più stabili rispetto a quelli ottenuti con il gradiente, l'Hessiano, o la funzione Harris-Corner. La relazione tra D e $\sigma^2\nabla^2 G$ può essere compresa tramite l'equazione di diffusione del calore:

$$\frac{\partial G}{\partial \sigma} = \sigma^2 \nabla^2 G.$$

Da questo si evince che $\nabla^2 G$ può essere calcolato come approssimazione alle differenze finite di $\frac{\partial G}{\partial \sigma}$, usando la differenza delle due vicine scale, una di $k\sigma$ e l'altra di σ:

$$\sigma^2 \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

e quindi:

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G. \tag{3.3}$$

Questo dimostra che quando la funzione differenza di gaussiane di un livello di scala differisce di un fattore costante da un livello di scala differente, essa incorpora già la normalizzazione σ^2 necessaria perché il Laplaciano sia invariante rispetto alla scala. Il fattore $(k - 1)$ nell'equazione 3.3 è costante in tutte le scale, e quindi non influenza la localizzazione degli estremi.

L'idea che sta alla base della rappresentazione scale-space è quella di creare una famiglia di segnali nei quali i dettagli vengono progressivamente soppressi, un esempio in figura 3.14.

Figura 3.14: Scale-space di un'immagine.

Per velocizzare la costruzione dello scale-space, l'immagine viene sottocampionata ad intervalli regolari e nuovamente filtrata, dando origine ad insiemi di immagini detti ottave. L'immagine di ogni ottava ha associato un valore di deviazione standard della Gaussiana usata per la convoluzione dell'immagine stessa σ, pari al doppio rispetto alla corrispondente immagine nell'ottava precedente.

L'immagine iniziale viene convoluta con varie gaussiane producendo immagini separate da un fattore k nello scale-space. Ogni ottava viene divisa in un numero intero s di intervalli tali che $k = 2^{\frac{1}{s}}$. Dopo aver processato un'intera ottava, l'immagine che ha $\sigma = 2\sigma_0$ (σ_0 deviazione standard relativa alla prima immagine dell'ottava) viene sottocampionata di un fattore 2. In ogni ottava occorre calcolare $s + 3$ immagini per permettere una corretta estrazione degli estremi locali. Nell'esempio in figura 3.15, ogni ottava viene divisa in $s = 2$ intervalli. Si calcolano $s + 3 = 5$ immagini con valori di deviazione standard della gaussiana σ crescente di un fattore $k = 2^{\frac{1}{s}} = 2^{\frac{1}{2}}$.

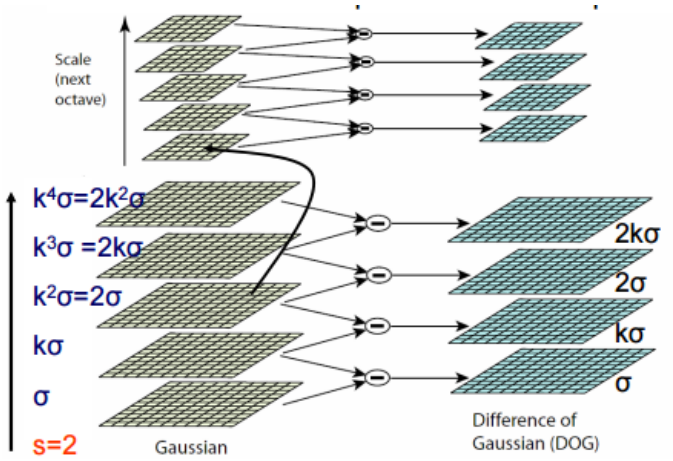


Figura 3.15: Creazione di uno Scale-Space.

Ovviamente $k^s = 2$ per cui il valore di sigma iniziale viene raddoppiato ed è proprio questa l'immagine da sottocampionare per l'ottava successiva.

Come si può capire ci sono diversi parametri che influenzano l'individuazione di keypoints SIFT. Per prima cosa, la ricerca di keypoints a scale multiple è ottenuta costruendo una Gaussiana scale-space. Come già descritto, la Gaussiana scale-space è una collezione di immagini ottenute con un progressivo smoothing dell'immagine principale, che è analogamente riduce gradualmente la risoluzione dell'immagine. La costruzione dello scale-space, e quindi la buona riuscita dell'algoritmo, è influenzata dai seguenti parametri che devono essere impostati alla creazione dell'algoritmo SIFT:

- il *numero di ottave*. Aumentando la scala di un'ottava significa raddoppiare la grandezza del kernel di smoothing, il cui effetto è di circa dimezzare la risoluzione dell'immagine. Il numero di ottave massime è: $log_2(min(largezza, altezza))$.

- l'*indice della prima ottava*. Comunemente si usa partire dall'ottava ad indice nullo che incomincia con la risoluzione dell'immagine (1 dimezza la risoluzione).

- il *numero di livelli per ottava*. Ogni ottava è divisa in un numero di scales intermedi (3 di default). Aumentando questo numero teoricamente si ha una localizzazione dei keypoint più precisa, ma in pratica può essere che rende la selezione instabile a causa del rumore accentuandolo.

È fondamentale inoltre definire il *minimo valore di contrasto* perché un punto sia accettato come keypoints.

Localizzazione accurata dei keypoints

Una volta trovato un punto candidato comparando un pixel con quelli vicini, il passo successivo è quello di testarne la robustezza, scartando i punti con poco contrasto o localizzati lungo edge. Nel 2002 Brown e Lowe [37] hanno utilizzato una funzione quadratica 3D su campioni di punti locali per determinare la posizione interpolata del massimo per aumentare la stabilità dell'algoritmo. Il loro approccio usa l'espansione di Taylor ai termini quadratici della funzione scale-space, $D(x, y, k\sigma)$ centrata sul punto esaminato:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{3.4}$$

dove D e le sue derivate sono valutate nell'intorno del punto preso in considerazione e $\mathbf{x} = (x, y, \sigma)^T$ è l'offset da questo punto. La locazione dell'estremo, $\hat{\mathbf{x}}$, è determinato annullando la derivata della funzione stessa rispetto a $\mathbf{x}$:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}} \tag{3.5}$$

Se l'offset $\hat{\mathbf{x}}$ assume un valore maggiore di 0.5 in qualunque direzione, significa che l'estremo giace più vicino a un altro punto. In questo caso il punto campione viene cambiato e l'interpolazione viene ricalcolata secondo le nuove coordinate. Nella formula (3.6) la funzione valutata all'estremo, $D(\hat{\mathbf{x}})$, è utile per eliminare gli estremi con poco contrasto tramite un valore soglia h, cioè $|D(\hat{\mathbf{x}})| > h$.

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}} \tag{3.6}$$

Eliminazione degli edge

Per avere la stabilità, non è sufficiente eliminare i keypoints con basso contrasto. La funzione Differenza di Gaussiane ha una risposta molto alta lungo i bordi anche se le locazioni lungo essi possono essere poco determinate e quindi instabili, anche con poco rumore.

Ricordiamo che un edge è tale se la funzione Differenza di Gaussiane in quel punto ha una curvature principale ampia ma esigua nella direzione perpendicolare. Le curvature principali può essere calcolata grazie alla matrice Hessiana 2x2, $\mathbf{H}$, calcolata nella locazione e scala del keypoints in esame:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

Le derivate sono calcolate come differenze tra i punti campione adiacenti. Gli autovalori di H sono proporzionali alle curvature principali di D. Per rilevare un

edge, non serve quindi calcolare l'autovalore, è sufficiente sapere il rapporto tra essi. Sia α l'autovalore di valore assoluto maggiore e β quello di valore assoluto inferiore:

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta$$
$$Det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

Nelle rare volte in cui il determinante è negativo, le curvature hanno segno differente, per cui il punto viene scartato non essendo un estremo. Sia r il rapporto tra l'autovalore con il valore assoluto maggiore e quello di valore assoluto minore, valga quindi $\alpha = r\beta$. Si ha:

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})^2} = \frac{(\alpha+\beta)^2}{r\beta^2} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{3.7}$$

che dipende solo dal rapporto degli autovalori e non dal valore individuale. $(r+1)^2/r$ è al minimo quando due autovalori sono uguali e aumenta all'aumentare di r. Quindi, per verificare che il rapporto tra le curvature principali sia al di sotto di una soglia, r si deve semplicemente verificare che

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})^2} < \frac{(r+1)^2}{r}$$

Questa relazione è computazionalmente leggera e permette di non prendere in considerazione gli edges come keypoints, rendendo più stabile questa parte di algoritmo.

Assegnamento dell'orientamento principale

Ad ogni punto chiave viene associata una direzione basandosi sulle proprietà locali dell'immagine, in questo modo il descrittore del keypoint può essere reso invariante alla rotazione calcolandolo relativamente al suo orientamento. La scala associata al punto d'interesse è utilizzata per selezionare l'immagine filtrata da una funzione Gaussiana, $L(x, y)$, che ha valore di scala più vicino a quello del punto stesso, in modo che tutti i calcoli siano invarianti alla scala. Per ogni keypoint campione viene calcolata la magnitudo del gradiente $m(x, y)$ e l'orientamento $\theta(x, y)$ usando la differenza tra pixel:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = tan^{-1}\Big(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\Big)$$

Si forma così un istogramma dell'orientamento dei gradienti dei punti in un intorno del keypoint. L'istogramma dell'orientamento ha 36 bin che coprono tutto l'intervallo di 360° delle possibili direzioni. Ogni campione che si aggiunge all'istogramma è pesato per il modulo del suo gradiente e per una finestra gaussiana circolare con raggio σ 1,5 volte rispetto a quello della scala del keypoint. I picchi nell'istogramma degli orientamenti corrispondono alle direzioni principali dei gradienti locali. Si rileva il picco più alto e ogni altro picco che risulti entro l'80% del picco maggiore. Per locazioni che hanno più picchi di modulo simile, vengono creati più keypoints per la stessa locazione e per la stessa scala ma con orientamenti differenti. Solo circa il 15% dei punti risultano presentare orientamenti multipli ma questi contribuiscono significativamente alla stabilità delle corrispondenze. Infine, una parabola viene fatta passare per i tre valori dell'istogramma più vicini ad ogni picco per trovare con maggiore accuratezza la posizione di ogni picco. Dai test risulta che l'assegnamento dell'orientamento rimane accurato nel 95% dei casi anche dopo l'aggiunta del 10% di pixel rumorosi.

Descrittore locale

Per il calcolo dei descrittori vengono prima di tutto calcolati magnitudo ed orientazione del gradiente nei punti che si trovano in un intorno del keypoint in esame. Per ottenere invarianza rotazionale le coordinate del descrittore e le orientazioni del gradiente vengono ruotate relativamente a quelle del punto chiave. Viene inoltre assegnato un peso alle magnitudo dei pixel in base ad una funzione Gaussiana con σ pari alla metà della larghezza della finestra del descrittore. Questo è illustrato tramite una finestra circolare nella parte sinistra dell'immagine. Lo scopo di tale finestra gaussiana è evitare sostanziali cambiamenti nel descrittore per piccoli cambiamenti della posizione della finestra e dare meno importanza ai gradienti lontani dal centro del descrittore, poiché questi sono più soggetti a errori di allineamento.

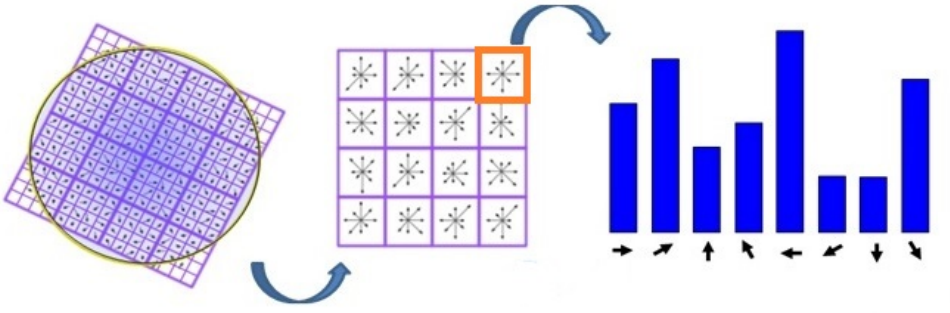


Figura 3.16: Schema del calcolo del descrittore SIFT.

Si considera poi una griglia di 16x16 pixel intorno al keypoint nell'immagine $L(x, y, \sigma)$ in cui tale punto è stato individuato. La griglia è suddivisa in 4x4 regioni, ciascuna di 4x4 pixel. Come si vede nella parte destra della figura 3.16, in ognuna di tali regioni viene costruito un istogramma delle orientazioni dei gradienti, normalizzati rispetto all'orientazione principale del keypoint. Ogni istogramma ha 8 bin, quindi il descrittore risultante ha 128 elementi (4x4x8).

3.4.3 SURF: Speeded Up Robust Features

L'algoritmo SURF è stato implementato nel 2006 da Herbert Bay, Tinne Tuytelaars e Luc Van Gool [39] con l'intento di comparare in performance e robustezza l'algoritmo SIFT ed essere più veloce computazionalmente. Questo scopo è stato raggiunto grazie a una serie di approssimazioni e accorgimenti sui filtri usati, sulla modalità di costruzione della piramide scale-space e sulla struttura del descrittore stesso.

L'idea su cui si basa questo algoritmo è quella di sostituire i filtri DoG con box filters che possono essere applicati sfruttando il concetto di immagine integrale, definita da Viola e Jones [40], e riducendo il carico computazionale.

Immagine integrale

L'immagine integrale consiste in una rappresentazione matriciale dell'immagine originale, in cui ogni elemento in posizione (x, y) contiene la somma dei livelli di intensità di tutti i pixel aventi coordinate inferiori o uguali a (x, y). Sia I un'immagine a toni di grigio. L'immagine integrale $\mathbb{I}$ del pixel (x, y) è così definita:

$$\mathbb{I}(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x, y)$$

Grazie all'immagine integrale è possibile, con un costo computazionale di quattro somme, ottenere la sommatoria di una qualunque parte rettangolare dell'immagine I:

$$\sum_{y=y_0}^{y_1} \sum_{x=x_0}^{x_1} I(x, y) = \mathbb{I}(x_1, y_1) + \mathbb{I}(x_0, y_0) - \mathbb{I}(x_1, y_0) - \mathbb{I}(x_0, y_1) \qquad (3.8)$$

Oltre a poter ottenere la sommatoria dei livelli d'intensità di grigio per ogni regione rettangolare dell'immagine, è possibile ottenere facilmente una veloce implementazione dei box filters, particolari filtri spiegati nel paragrafo successivo.

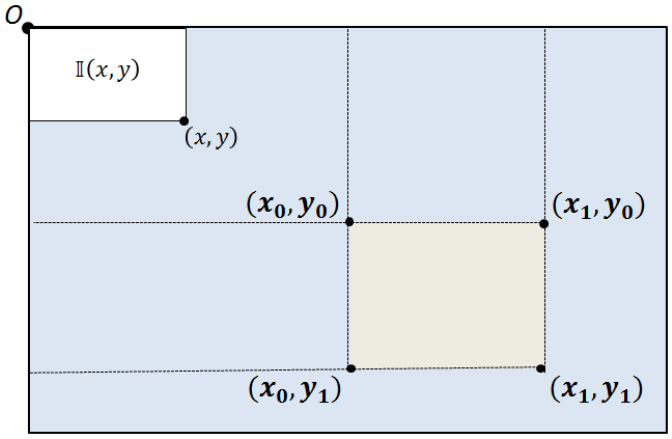


Figura 3.17: Immagine rappresentativa della formula 3.8.

Fast-Hessian Detector

Il rilevatore del SURF si basa sulla matrice Hessiana. Dato un punto $\mathbf{x} = (x, y)$ in un'immagine I, la matrice Hessiana $\mathbf{H}(\mathbf{x}, \sigma)$ nel punto $\mathbf{x}$ alla scala σ è definita come:

$$\mathbf{H}(\mathbf{x}, \sigma) \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{yx}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

Dove $L_{xx}(\mathbf{x}, \sigma)$ è la convoluzione della derivata di secondo ordine della Gaussiana $\frac{\partial^2}{\partial x^2} g(\sigma)$ con l'immagine I nel punto $\mathbf{x}$, e allo stesso modo con $L_{xy}(\mathbf{x}, \sigma)$ e $L_{yy}(\mathbf{x}, \sigma)$. La funzione gaussiana è ottima per l'analisi scale-space, ma nella pratica ha bisogno di essere discretizzata e limitata, come si vede nell'immagine 3.18, per operare su immagini digitali, creando quindi effetti dovuti all'aliasing. È possibile utilizzare una tecnica in cui non è necessario applicare in modo iterativo il medesimo filtro all'output dell'operazione precedente, come avviene nello scale-space. Questa tecnica si basa sull'utilizzo dei box filters. Questi infatti approssimano le derivate di secondo ordine delle Gaussiane e possono essere calcolati molto velocemente tramite l'uso dell'immagine integrale.

Chiamiamo queste approssimazioni D_{xx}, D_{yy} e D_{xy}. La localizzazione dei punti di interesse si basa sul calcolo del determinante della matrice Hessiana approssimata:

$$det(\mathbf{H}_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2$$

Dove il valore w è un fattore di peso necessario per approssimare meglio la gaussiana. Inoltre, la risposta del filtro deve essere normalizzata rispetto alla sua dimensione.

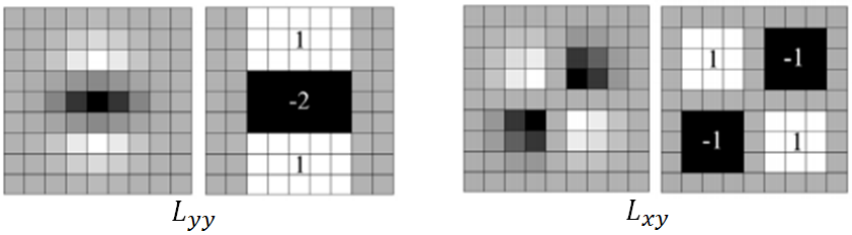


Figura 3.18: Da sinistra a destra, la funzione gaussiana di secondo ordine L_{yy} e la sua approssimazione, la funzione gaussiana di secondo ordine L_{xy} e la sua approssimazione.

Lo scale-space è solitamente implementato come una piramide di immagini, le quali sono ripetutamente filtrate tramite una funzione Gaussiana. Grazie all'impiego di box filters e alle immagini integrali, non è necessario applicare iterativamente lo stesso filtro su un altro livello prefiltrato, ma si possono applicare all'immagine originale più filtri di diverse grandezze in parallelo. Perciò l'analisi scale–space è effettuata ingrandendo le dimensioni del filtro e non riducendo iterativamente le dimensioni dell'immagine, come si vede in figura 3.19. Lo scale-space viene per semplicità diviso in ottave. Per ogni nuova ottava le dimensioni del filtro raddoppiano.

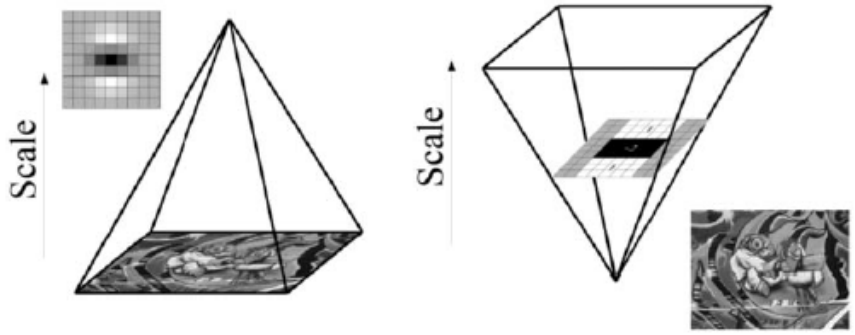


Figura 3.19: A sinistra si può notare come nel SIFT l'immagine viene iterativamente sottocampionata e scalata. A destra invece è mostrato come nel SURF è il filtro che, ad ogni ottava, viene ingrandito.

Il descrittore SURF

Il descrittore SURF si basa su simili proprietà del SIFT. Il primo passo consiste nel fissare un'orientazione riproducibile basata sulle informazioni nell'intorno circolare del punto di interesse, successivamente si allinea una regione quadrata all'orientazione trovata, estraendo da essa il descrittore del punto. Per informazione generale, esiste anche un ulteriore descrittore chiamato U-SURF che risulta invariante alla

rotazione e più leggero nel carico computazionale ma è più prestante solamente se la camera non ruota attorno all'asse z uscente da essa. Questa lo porta ad essere un descrittore non utilizzabile per lo scopo di questa tesi.

Assegnamento dell'orientazione

Per rendere il descrittore invariante alla rotazione, è necessario identificare un'orientazione riproducibile per il punto di interesse. A questo scopo, innanzitutto, viene calcolata la risposta alla wavelet di Haar, dei particolari box filters, nella direzione x e y, nell'intorno circolare di raggio $4s$, dove s è la scala associata al keypoint.

Una volta che le risposte alla wavelet sono calcolate e pesate con una gaussiana centrata nel punto di interesse, esse vengono rappresentate nello spazio, come si può vedere in figura 3.20: la risposta orizzontale viene rappresentata sull'asse delle ascisse, mentre quella verticale sull'asse delle ordinate. L'orientazione dominante è stimata calcolando la somma di tutte le risposte con una finestra di orientamento mobile, avente un angolo di $60°$ e viene assunta come orientamento locale. Il vettore più lungo tra tutti quelli calcolati con le diverse finestre definisce l'orientamento del punto di interesse.

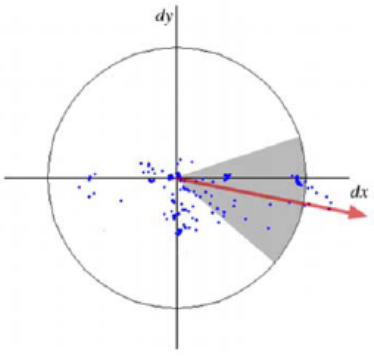


Figura 3.20: Rappresentazione del calcolo dell'orientamento. Il settore grigio è la finestra mobile.

Componenti del descrittore

Per l'estrazione del descrittore, il primo passo consiste nel costruire una regione centrata attorno al punto di interesse, e orientata come descritto nella paragrafo precedente. Per la versione U-SURF questo passo non è necessario in quanto si presuppone che la telecamera sia per lo più orizzontale.

Come rappresentato in figura 3.21, la regione è divisa in 4x4 sottoregioni quadrate, in modo da preservare le informazioni spaziali, per ognuna delle quali viene calcolata la risposta di Haar Wavelet sia nella direzione orizzontale, chiamata d_x, sia in quella verticale, chiamata d_y. Per rendere il descrittore più robusto ad eventuali deformazioni geometriche e ad errori di localizzazione, le risposte d_x e d_y vengono pesate da una funzione gaussiana centrata nel punto di interesse.

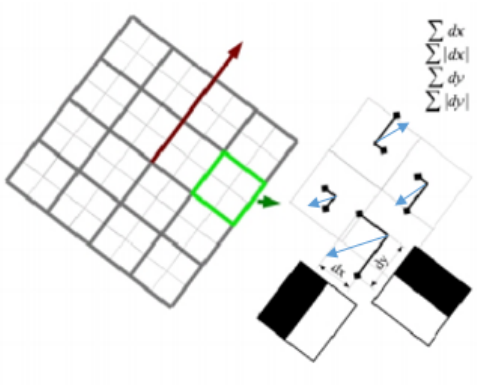


Figura 3.21: Schema di calcolo del descrittore SURF.

Le risposte alla wavelet d_x e d_y sono sommate per ogni regione, formando una prima serie di vettori caratteristici. Per mantenere informazioni sulla polarità dei cambiamenti di intensità vengono anche estratte le somme dei valori assoluti, $|d_x|$ e $|d_y|$. Ogni regione è definita quindi da un vettore v di 4 elementi $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ che dipende dalla struttura dell'intensità nell'intorno del keypoints. Risulta quindi per ognuna delle 4x4 regioni, un vettore descrittivo formato da 64 elementi, mentre nel SIFT erano 128. Esempi del vettore v al variare dell'area intorno al keypoints, sono riportati nell'immagine 3.22.

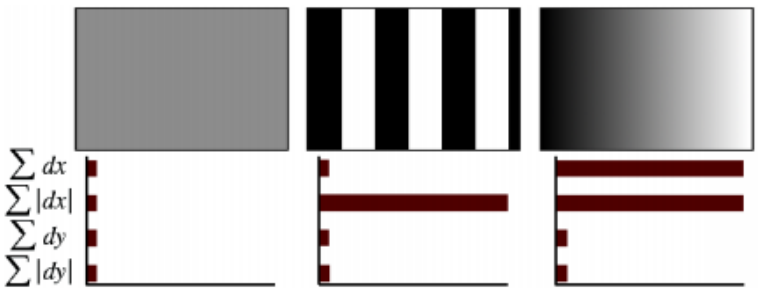


Figura 3.22: Esempi dei valori del vettore v a seconda del contenuto dell'area intorno al keypoints.

3.4.4 Rilevatori per il real time: FAST e AGAST

Gli algoritmi SIFT e SURF hanno avuto molto successo per la loro robustezza. Questa è raggiunta a discapito di un carico computazionale e un consumo di memoria elevato. Sia il rilevatore Harris, che si basa sul calcolo dei gradienti, sia i rilevatori SIFT e SURF, che si basano sul calcolo dell'Hessiano, sono computazionalmente troppo pesanti per sostenere il real time. Per questo si è cercato di implementare nuovi algoritmi di rilevamento dei corner più veloci.

FAST: Features From Accelerated Segment Test

L'algoritmo FAST [41] valuta la differenza di luminosità dei pixel appartenenti al cerchio di Bresenham di raggio r attorno al punto considerato. Se n pixel contigui sono tutti più intensi o tutti meno intensi del nucleo di almeno un valore soglia t, allora il pixel del nucleo è considerato un keypoint.

Solitamente viene utilizzato un cerchio di Bresenham di raggio 3 pixels in modo da comparare 16 pixels al valore d'intensità luminosa del nucleo, come è possibile vedere in figura 3.23. Il pixel in esame è un keypoint se almeno S pixels continui sono tutti più intensi o tutti meno intensi del nucleo di almeno un valore soglia t. Il valore S definisce la massima ampiezza dell'angolo rilevato. Con $S = 8$ è possibile rilevare gli edge. Mantenendo invece S elevato, $S = 12$, non si considerano gli edge e cresce la ripetibilità del corner detector.

Il valore t definisce la differenza minima di intensità luminosa usata per il confronto tra uno qualsiasi dei pixel e il nucleo; esso controlla quindi la sensibilità della risposta. Con un valore di t elevato si rilevano pochi ma definiti corners, con un valore basso si rilevano molti corners, anche quelli a basso gradiente, rendendo così poco stabile l'algoritmo.

Il test per queste condizioni può essere ottimizzato esaminando i pixels 1, 9, 5 e 13 per scartare più velocemente i pixel candidati; un corner esiste solamente se tre di questi pixels sono tutti più intensi, o tutti meno intensi, del nucleo di un valore soglia t.

Questo tipo di corner detector permette di usare l'intensità dei 16 pixel per formare un vettore caratteristico che definisce il keypoints e può essere quindi utilizzato come feature.

Ci sono alcune limitazioni per questo algoritmo: in primo luogo, con $S < 12$, i punti trovati dall'algoritmo sono troppi; inoltre l'ordine in cui viene valutata la luminosità dei 16 pixel determina la velocità dell'algoritmo. Per fornire una soluzione a questi due problemi il FAST utilizza un approccio di *machine learning*.

Questo approccio opera in due fasi. Nella prima viene selezionata una serie di immagini per l'allenamento, in ognuna delle quali viene utilizzato l'algoritmo

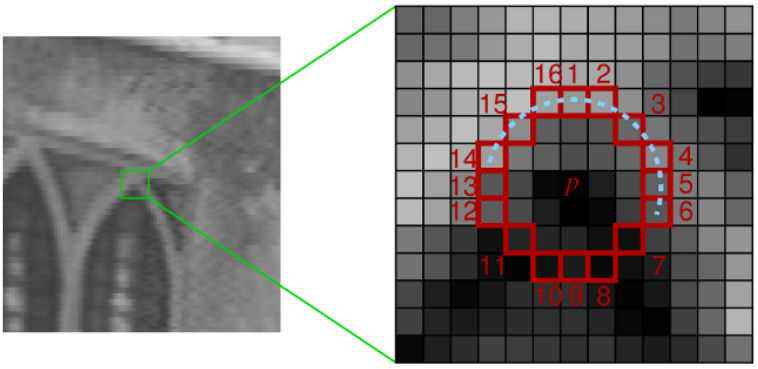


Figura 3.23: Cerchio di Bresenham nell'algoritmo FAST.

di corner detection per determinare i punti di interesse, prendendo un pixel alla volta e valutando tutti i 16 pixel nel cerchio descritto in precedenza.

Per ogni punto p, ogni posizione appartenente al cerchio $x \in 1, 2, 3, \ldots, 16$ può assumere 3 stati: più luminoso di p, più scuro di p o simile a p. Matematicamente si ha:

$$S_{p \to x} = \begin{cases} d, & I_{p \to x} \leq I_p - t \quad \text{(più scuro)} \\ s, & I_p - t < I_{p \to x} < I_p - t \quad \text{(simile)} \\ b, & I_p + t \leq I_{p \to x} \quad \text{(più chiaro)} \end{cases}$$

dove $S_{p \to x}$ è lo stato, $I_{p \to x}$ è l'intensità del pixel x e t è il valore soglia.

Detto P la serie di tutti i pixel delle immagini prese in considerazione, è possibile dividere P in tre differenti sottoinsiemi P_d, P_s, P_b dove:

$$P_d = p \in P : S_{p \to x} = d;$$
$$P_s = p \in P : S_{p \to x} = s;$$
$$P_b = p \in P : S_{p \to x} = b;$$

Nella seconda fase un algoritmo decisionale ad albero, l'algoritmo ID3, viene applicato alle 16 posizioni x per ottenere il massimo guadagno di informazione. Alla fine di questo processo viene restituita un ordine di importanza tra le posizioni $x \in \{1, 2, 3, \ldots, 16\}$. Valutando i 16 pixel in quest'ordine, derivante dall'albero di decisione, si accelera notevolmente la rilevazione di keypoints.

Infine si calcola una funzione V per ogni corner rilevato e viene applicata una soppressione dei non massimi per rimuovere i corners aventi nelle vicinanze un altro corner con V maggiore. Il parametro V è dato da:

$$V = max \left(\sum_{x \in S_{bright}} |I_{p \to x} - I_p| - t, \sum_{x \in S_{dark}} |I_p - I - p \to x| - t \right)$$

con:

$$S_{bright} = \{x | I_{p \to x} \geq I_p + t\}$$
$$S_{dark} = \{x | I_{p \to x} \leq I_p - t\}$$

In altre parole V consiste nella somma dei valori assoluti della differenza tra i pixel in un'area limitrofa ed il pixel centrale.

AGAST: Adaptive and Generic Accelerated Segment Test

L'algoritmo FAST, pur essendo computazionalmente rapido, ha dei lati negativi:

- Se la camera viene ruotata, la configurazione dei pixel di un corner cambia considerevolmente. Perciò l'algoritmo risulta ottimizzato alla scena fino a quando la camera non ruota. Ogni movimento porta l'algoritmo di rilevazione di corners ad essere più lento.
- Alcune configurazioni di corner possono essere state persi durante l'allenamento, ciò permette che siano presenti risposte di falsi positivi e falsi negativi.
- L'albero decisionale deve essere calcolato per ogni nuovo ambiente

L'algoritmo AGAST risulta essere simile al FAST. La differenza risiede nel tipo di tecnica utilizzata per la costruzione dell'albero decisionale. Questo algoritmo propone una tecnica di calcolo di un duplice albero decisionale generico che non deve essere adattato ad un nuovo ambiente. Combinando due alberi, il corner detector si adatta automaticamente all'ambiente e fornisce un risultato rapido preservando la ripetibilità dell'algoritmo FAST. Per un maggior approfondimento di questo algoritmo si rimanda a [42].

3.4.5 Descrittori Binari: BRIEF

I descrittori tradizionali sono formati da un vettore di un certo numero di valori a virgola mobile. Per creare questo vettore ed eseguire in seguito i confronti è necessario un carico computazionale che rende questi descrittori difficilmente utilizzabili per le applicazioni real-time, specialmente se la piattaforma utilizzata è uno smartphone o un computer con un processore lento. Nonostante il descrittore utilizzato nel SURF accelera il calcolo utilizzando immagini integrali, in alcune applicazioni non risulta essere abbastanza veloce. Inoltre SIFT e SURF sono protetti da brevetto commerciale.

Per soddisfare la domanda di velocità e limitato carico computazionale sono nati alcuni algoritmi che propongono di ridurre le dimensioni del descrittore senza diminuire la proprietà discriminativa. L'idea è di codificare la maggior parte delle

informazioni di una zona in una stringa binaria usando solo il confronto della intensità nelle immagini. Questo può essere fatto molto velocemente, inoltre il matching tra due "patch" può essere fatto utilizzando una singola istruzione, come la distanza di Hamming pari alla somma della operazione XOR tra le due stringhe binarie. Questo è esattamente quello che viene fatto nei descrittori binari.

BRIEF: Binary Robust Independent Elementary Features

Il primo descrittore binario è stato realizzato da Calonder nel 2010 [43]. Le parti di un'immagine possono essere efficacemente classificate confrontando l'intensità di un piccolo numero di coppie di pixel.

Definiamo il test τ in un intorno quadrato $\mathbf{p}$ con finestra $S \times S$ del keypoint come:

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{altrimenti} \end{cases}.$$

dove $\mathbf{p}(\mathbf{x})$ è l'intensità del pixel alla posizione $\mathbf{x} = (u, v)^T$. Scelto un set di n_d coppie $(\mathbf{x}, \mathbf{y})$, è possibile definire un'unica stringa di bit che definisce l'intorno $\mathbf{p}$, tramite il descrittore BRIEF così definito:

$$f_{n_d}(\mathbf{p}) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i)$$

dove solitamente n_d prende il valore di 512, formando una stringa lunga 512 bit.

Per ottenere questa maschera di coordinate è necessario eseguire dei test sulle immagini per ottenere la combinazione che massimizza le rilevazioni corrette, e cioè il test τ. Il problema di questo metodo risiede nel fatto di dover compiere studi precedenti dell'ambiente. É necessario un addestramento ogni volta che l'ambiente cambia.

3.4.6 BRISK: Binary Robust Invariant Scalable Keypoints

L'algoritmo BRISK presentato per la prima volta da Leutenegger, Chli e Siegwart [44], è un metodo per la rilevazione dei keypoints, identificazione dei descrittori e matching. Secondo quanto dimostrato dagli autori è computazionalmente molto veloce, mentre per quanto riguarda l'accuratezza è paragonabile a SIFT e SURF.

Per capire meglio la struttura dell'algoritmo esso viene diviso in tre fasi:

- Identificazione dei Keypoints nello scale-space;
- Descrizione dei Keypoints;

- Matching dei descrittori.

Identificazione dei Keypoints nello scale-space

Questa parte dell'algoritmo è ispirata dall'AGAST. Con lo scopo di essere invarianti alla scala, qualità cruciale per un keypoint, l'algoritmo BRISK ricerca i massimi non solo nell'immagine piana, ma, a differenza dell'AGAST, anche nello scale-space, utilizzando il parametro V, introdotto nel paragrafo 3.4.4, come misura di importanza.

Nella struttura del BRISK, i livelli della piramide dello scale-space consistono in n ottave c_i e n intra-ottave d_i per $i = 0, 1, \ldots, n-1$ e solitamente $n = 4$. Le ottave sono formate dimezzando progressivamente la campionatura dell'immagine originale, corrispondente a c_0. Ogni intra-ottava d_i è posizionata tra il layer c_i e $c_i + 1$ come illustrato in figura 3.24.

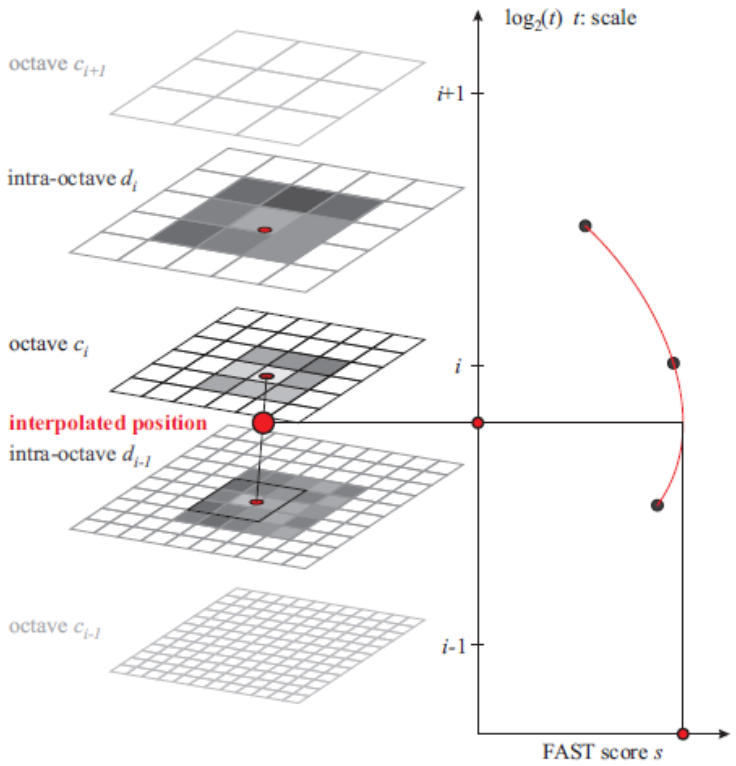


Figura 3.24: Rilevamento dei punti di interesse nello scale-space: un keypoints è identificato nell'ottava c analizzando gli 8 punti adiacenti e negli strati immediatamente contigui sopra e sotto.

La prima intra-ottava d_0 si ottiene sottocampionando l'immagine originale c_0 di un fattore di $1,5$, mentre il resto degli altri strati intra-ottava sono derivati da successivi sottocampionamenti di un fattore 2. Quindi detta t la scala si avrà $t(c_i) = 2^i$ e $t(d_i) = 2^i \cdot 1,5$.

Sia il FAST che l'AGAST usano diverse maschere per il rilevamento di keypoints. Nel BRISK si utilizza la maschera 9-16, cioè almeno 9 pixel consecutivi nel cerchio di 16 pixel devono essere tutti sufficientemente più luminosi, o meno luminosi, del pixel centrale. Inizialmente, l'algoritmo applica il rilevatore FAST 9-16 per ogni ottava e intra-ottava separatamente usando lo stesso valore soglia t per identificare regioni potenziali di interesse. I punti appartenenti a queste regioni sono sottoposti a soppressione dei non massimi nello scale-space. La Figura 3.24 mostra come vengono individuati i punti di interesse nello scale-space: un keypoint viene identificato all'ottava c_i, sia analizzando i valori V di rilevanza dei suoi 8 vicini, sia valutando i valori V relativi ai vicini negli strati immediatamente sopra e immediatamente sotto. In tutti e tre gli strati di interesse, il punto con rilevanza massima locale, cioè avente V maggiore, viene precisato a livello di pixel ed in seguito, tramite l'adattamento ad una parabola 1D lungo l'asse delle scale, viene interpolata la scala esatta del keypoint.

Descrizione dei Keypoints

Dato un insieme di keypoints, il descrittore BRISK è composto da una stringa binaria formata concatenando i risultati di semplici test di confronto della luminosità. L'idea è ripresa dal BRIEF ed è risultata molto efficiente. In BRISK viene identificata la direzione principale di ogni keypoint per consentire che i descrittori siano normalizzati rispetto all'orientazione principale e quindi raggiungere l'invarianza di rotazione, questo è fondamentale per la robustezza generale.

Pattern di Campionamento

Il descrittore BRISK fa uso di un pattern per campionare i punti vicini al keypoint. In figura 3.25 è rappresentato il pattern di campionamento con $N = 60$ punti: i piccoli cerchi blu indicano le posizione in cui viene effettuato il campionamento, mentre i cerchi rossi tratteggiati più grandi, centrati sui punti campionati, hanno raggio σ corrispondente alla deviazione standard del kernel gaussiano usato per smussare i valori d'intensità degli stessi punti campionati.

Stima della direzione principale

Considerando un keypoint k, una delle $N(N-1)/2$ coppie di punti campione $(\mathbf{p}_i, \mathbf{p}_j)$ e i rispettivi valori di intensità smussata dei punti $I(\mathbf{p}_i, \sigma_i)$ e $I(\mathbf{p}_j, \sigma_j)$, è possibile stimare il gradiente locale $\mathbf{g}(\mathbf{p}_i, \mathbf{p}_j)$:

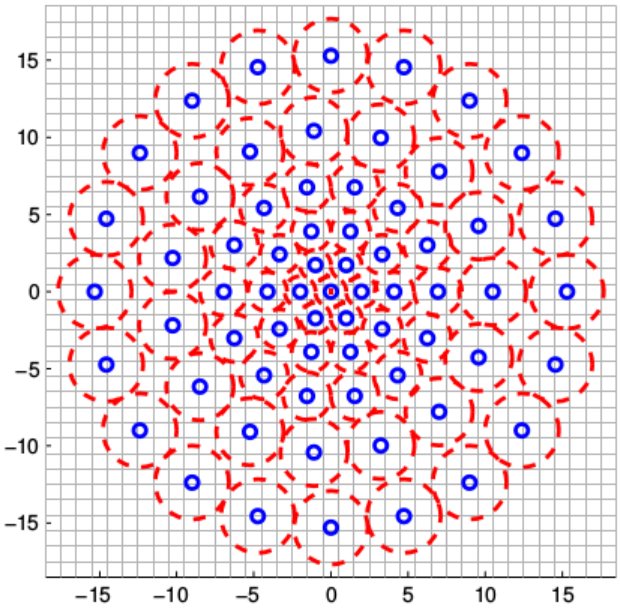


Figura 3.25: Pattern del descrittore BRISK.

$$\mathbf{g}(\mathbf{p}_i, \mathbf{p}_j) = (\mathbf{p}_j - \mathbf{p}_i) \cdot \frac{I(\mathbf{p}_j, \sigma_j) - I(\mathbf{p}_i, \sigma_i)}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}$$

Sia $\mathcal{A}$ l'insieme di tutte le coppie dei punti di campionamento:

$$\mathcal{A} = \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathbb{R}^2 \times \mathbb{R}^2 \mid i < N \wedge j < i \wedge i, j \in \mathbb{N}\}$$

definiamo un sottoinsieme delle coppie a distanza breve $\mathcal{S}$ e un altro sottoinsieme delle L coppie a distanza lunga $\mathcal{L}$:

$$\mathcal{S} = \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{A} \mid \|\mathbf{p}_j - \mathbf{p}_i\| < \delta_{max}\} \subseteq \mathcal{A}$$
$$\mathcal{L} = \{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{A} \mid \|\mathbf{p}_j - \mathbf{p}_i\| > \delta_{min}\} \subseteq \mathcal{A}$$

Scorrendo le coppie di punti in $\mathcal{L}$ si stima le direzione principale del keypoint k come:

$$\mathbf{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \cdot \sum_{(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{L}} \mathbf{g}(\mathbf{p}_i, \mathbf{p}_j).$$

Le coppie a distanza lunga sono usate per questo calcolo, basandosi sul presupposto che i gradienti locali si annullino tra di loro e non siano quindi necessari alla determinazione del gradiente globale.

Costruzione del descrittore

Per la formazione di un descrittore binario normalizzato che tenga conto dell'invarianza alla rotazione e alla scala, BRISK applica lo schema di campionamento ruotato di un angolo $\sigma = \arctan 2(g_y, g_x)$, attorno al keypoint k. Il descrittore risulta un vettore di bit ricavato tramite il confronto delle intensità delle coppie di punti a distanza breve $(\mathbf{p}_i^\alpha, \mathbf{p}_j^\alpha) \in \mathcal{S}$ in modo tale che ogni bit b corrisponda a:

$$b = \begin{cases} 1, & \text{if } I(\mathbf{p}_j^\alpha, \sigma_j) > I(\mathbf{p}_i^\alpha, \sigma_i) \\ 0, & \text{altrimenti} \end{cases} \quad \forall (\mathbf{p}_i^\alpha, \mathbf{p}_i^\alpha) \in \mathcal{S}.$$

Per ogni coppia a distanza breve viene considerata l'intensità smussata dei punti campionati. Se l'intensità del primo punto nella coppia è maggiore di quella del secondo punto il bit del descrittore assumerà il valore "1", altrimenti "0". Il descrittore BRIEF è anch'esso ricavato tramite una comparazione di intensità luminosa, il BRISK però ha differenze fondamentali oltre alla pre-rotazione e all'invarianza di scala dello schema di campionamento. In primo luogo utilizza uno schema di campionamento deterministico, che comporta una densità uniforme dei punti campione ad un raggio fissato attorno al keypoint. Quindi non può succedere che lo smussamento Gaussiano distorga accidentalmente l'informazione dei confronti delle coppie, prendendo un punto campionato errato vicino. Inoltre, BRISK usa molti meno punti campionati per il confronto tra le coppie, un singolo punto partecipa in più confronti, limitando la complessità del calcolo dei valori di intensità. Con il pattern di campionamento e i valori di soglia presi in considerazione, si ottiene una stringa di 512 bits.

Matching dei descrittori

Il matching tra due descrittori è una semplice e veloce computazione della loro distanza di Hamming, come accade nel BRIEF. Un diverso numero di bit nei due descrittori è una misura della loro diversità. I calcoli si riducono a operazioni XOR, rendendo veloce ed efficiente l'algoritmo.

3.5 KeyPoints e Features 3D

Tutti gli algoritmi del paragrafo 3.4 trattano della rilevazione e descrizione di keypoint bidimensionale. Essi infatti necessitano come input delle immagini. Per lo studio proposto in questa tesi però è necessario estendere l'analisi su rilevatori e descrittori che gestiscono nuvole di punti tridimensionali.

Grazie a questi algoritmi, oltre che l'informazione del colore, è possibile sfruttare l'informazione della distanza, le normali delle superfici e la curvatura nel processo di registrazione, in modo da ottenere un allineamento consistente con la forma delle nuvole di punti.

In questa sezione si presenteranno dunque i principali keypoint detectors e descriptors.

3.5.1 HARRIS 3D e 6D

L'algoritmo Harris 3D usato nelle PCL si discosta dal più complesso e omonimo algoritmo descritto in [45]. Dallo script implementato nelle PCL si può notare come si utilizzi la struttura algoritmica dell'Harris 2D. L'unica grande differenza risiede nel prendere i valori delle normali invece dell'intensità luminosa.

Sia p_0 un qualsiasi punto appartenente alla nuvola di punti, Ω_0 un intorno di p_0 e T il numero di punti appartenenti all'intorno con valori finiti alle normali. Una volta calcolate le normali per ogni singolo punto p appartenente all'intorno, è possibile creare una matrice di covarianza E:

$$E = \frac{1}{T}\begin{bmatrix} \sum_{p\in\Omega_{p_0}}(n_x(p))^2 & \sum_{p\in\Omega_{p_0}}(n_x(p)*n_y(p)) & \sum_{p\in\Omega_{p_0}}(n_x(p)*n_z(p)) \\ \sum_{p\in\Omega_{p_0}}(n_y(p)*n_x(p)) & \sum_{p\in\Omega_{p_0}}(n_y(p))^2 & \sum_{p\in\Omega_{p_0}}(n_y(p)*n_z(p)) \\ \sum_{p\in\Omega_{p_0}}(n_z(p)*n_x(p)) & \sum_{p\in\Omega_{p_0}}(n_z(p)*n_y(p)) & \sum_{p\in\Omega_{p_0}}(n_z(p))^2 \end{bmatrix}$$

dove $n_x(p)$, $n_y(p)$, $n_z(p)$ sono rispettivamente le componenti nelle direzioni x, y, z della normale calcolata in p. La matrice E così ottenuta è simmetrica. É possibile ora ottenere, tramite semplici calcoli, il determinante e la traccia di E. Si definisce ora il valore intensità I tramite la formula usata da Harris 2D:

$$I(p_0) = 0.04 + det - 0.04 * trace * trace.$$

Se $I(p_0)$ è maggiore di un certo valore soglia, allora si verifica che p_0 sia il massimo relativo tra i punti del suo intorno. Se la condizione è soddisfatta, p_0 risulta un keypoint.

Si può notare che questo algoritmo è utilizzabile anche con nuvole di punti non colorate poiché agisce unicamente sulle normali calcolate nell'intorno di quel punto. Non si considera quindi informazione RGB del colore, che fornisce il Kinect v2.

Per questo motivo nasce l'algoritmo Harris 6D, chiamato così poiché la matrice di covarianza risulta una 6x6, che tiene in considerazione sia l'informazione sulle normali che l'informazione di luminosità. La matrice di covarianza risulta:

$$E_{RGB} = \frac{1}{T}\begin{bmatrix} A_1 & A_2 \\ A_2 & A_3 \end{bmatrix}$$

dove T è il numero di punti appartenenti all'intorno considerato, A_1 è la stessa matrice di covarianza E dell'algoritmo Harris 3D, A_3 è la matrice di covarianza dovuta ai gradienti di luminosità nelle 3 direzioni, e A_2 è la matrice di covarianza realizzata moltiplicando il valore dei gradienti e il valore delle normali in x, y e z. Detti $\lambda_1 \dots \lambda_6$ gli autovalori della matrice, perché il punto sia un keypoint deve essere massimo relativo e deve verificarsi la relazione:

$$min(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6) > \lambda_{threshold}$$

dove $\lambda_{threshold}$ è un valore soglia predefinito.

3.5.2 SIFT 3D

L'algoritmo SIRF 3D, chiamato anche THRIFT, è stato presentato nel 2007 [46]. Per riportare l'algoritmo di keypoint detector presentato nel SIFT in un ambiente 3D, una funzione di densità $f(x, y, z)$ viene approssimata campionando regolarmente lo spazio. Poi si costruisce lo scale space attraverso la funzione trovata, e si cerca il massimo locale del determinante Hessiano.

La Mappa Densa

La nuvola di punti è essenzialmente una serie di punti:

$$\chi = x_i \in \mathbb{R}^3$$

Si desidera approssimare una funzione di densità $f(x, y, z)$ da questi punti. Sia $n(B)$ il numero di punti dati nella regione $B \subseteq \mathbb{R}^3$. Si approssimi $f(B)$ in tale regione usando

$$\int_B f(\mathbf{x})d\mathbf{x} = n(B)$$

Vengono definiti i cubi di ugual dimensione $B = \{B_{ijk}\}_{(i,j,k)\in I \subset Z^3}$ e distribuiti regolarmente in ogni dimensione:

$$B_{i,j,k} = (x, y, z) \in \mathbb{R}^3 : \begin{cases} \alpha i \leq x < \alpha(i+1) \\ \beta j \leq y < \beta(j+1) \\ \gamma k \leq z < \gamma(k+1) \end{cases}$$

Si costruisce ora f come somma delle funzioni:

$$f(x,y,z) = \sum_{(ijk)\in I} D(i,j,k)\delta(x - X_{ijk}, y - Y_{ijk}, z - Z_{ijk})$$

dove $(X_{ijk}, Y_{ijk}, Z_{ijk}) = \bar{B}_{ijk}$ è il centro del cubo B_{ijk} e

$$D(i,j,k) = \frac{n(B_{ijk})}{\underset{(i,j,k)\in I}{\operatorname{argmax}}\Big\{n(B_{ijk})\Big\}}$$

è la mappa normalizzata densa. La funzione argmax ridà l'insieme dei valori (i,j,k) per i quali $n(B_{ijk})$ raggiunge il suo più alto valore. In pratica si opererà direttamente su D che rappresenta un array 3D di valori. Ogni elemento della matrice rappresenta la densità nella regione di spazio. È possibile applicare la convoluzione 3D a D nello stesso modo in cui si applica per le immagini 2D.

Scale-space Denso

Come spiegato in 3.4.2, il rilevatore di punti 2D costruisce tipicamente uno scale-space, per considerare un'ampia gamma di scale, convolvendo l'immagine con un kernel gaussiano di varianza crescente, il risultato è una piramide di immagini. Si applica lo stesso concetto per avere la mappa densa D in una gamma di scale. La mappa densa D è convoluta con una serie di kernel gaussiani 3D per ricostruire una piramide della mappa densa, in cui ogni livello rappresenta la scala $\sigma = k\sigma'$ dove σ' è la scala del livello subito al di sotto. Sia $L(x,y,z;\sigma)$ uno scale-space per D

$$L(x,y,z;\sigma) = (D \otimes g(\sigma))(x,y,z)$$

dove $g(\sigma)$ è la Gaussiana 3D con varianza σ:

$$g(x,y,z;\sigma) = exp\big(\tfrac{-x^2-y^2-z^2}{2\sigma^2}\big)$$

Il numero delle operazioni di downsample (il numero delle ottave) e il numero delle scale che sono generate tra 2 downsample (il numero di livelli per piano) sono parametri specificati dall'utente.

Selezione dei keypoints

Questo algoritmo al fine di migliorare la ripetibilità trova i punti nella mappa densa D in cui è presente un'ampia curvatura in tutte e tre le dimensioni, in modo da essere riconosciuti anche in condizioni differenti. Si utilizza il determinante della matrice Hessiana 3×3 per trovare i punti, perché è accurata ed è definita per

scale arbitrarie. Dato un punto $x = (x, y, z)$ e una scala σ, l'hessiano in $(\mathbf{x}, \sigma)$ è definito come:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{pmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) & L_{xz}(\mathbf{x}, \sigma) \\ L_{yx}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) & L_{yz}(\mathbf{x}, \sigma) \\ L_{zx}(\mathbf{x}, \sigma) & L_{zy}(\mathbf{x}, \sigma) & L_{zz}(\mathbf{x}, \sigma) \end{pmatrix}$$

dove:

$$L_{xx}(\mathbf{x}, \sigma) = \frac{\delta^2}{\delta x^2} L(\mathbf{x}, \sigma)$$

é la derivata parziale seconda nella direzione x della funzione scale space L, in maniera simile per le altre derivate parziali. In pratica si sono calcolati i termini dell'hessiano tramite convoluzione diretta di D con la derivata seconda parziale della gaussiana.

$$S_{xx}(x, \sigma) = D \otimes \frac{\delta^2}{\delta x^2} g(\sigma)$$

Questo permette di evitare la costruzione dello space scale e calcolare i termini dell'hessiano come operazioni separate. Si calcola quindi il $|\text{Det}(\mathcal{H})|$ per ogni punto nello scale space. Si applica un valore di soglia T nella risposta per eliminare punti con risposte deboli e un filtro per eliminare i non massimi. I punti rimanenti sono massimi locali di $|\text{Det}(\mathcal{H})|$ e diventano così keypoints.

Il rilevatore di punti di interesse può essere riassunto nella formula:

$$keypoints(\chi) = \underset{(x,\sigma)}{\text{argmax}} \, |\text{Det}(\mathcal{H}(\mathbf{x}, \sigma)|$$

3.5.3 ISS 3D: Intrinsic Shape Signatures

L'algoritmo Intrinsic Shape Signatures (ISS) [47] è un metodo i cui risultati dipendono molto dalla qualità di misurazione.

Per prima cosa si definisce una struttura intrinseca di riferimento F_i per ogni punto p_i utilizzando gli autovalori calcolati dalla matrice di dispersione. Questa fase è riassunta in alcuni passaggi.

1. Viene calcolato per ogni punto p_i un peso w_i inversamente proporzionale al numero di punti in un proprio intorno sferico di raggio $r_{density}$:

$$w_i = \frac{1}{\|\{p_j : |p_j - p_i| < r_{density}\}\|}.$$

   Questo peso è usato per compensare un campionamento irregolare dei punti 3D, dunque i punti che hanno pochi vicini contribuiscono maggiormente rispetto a quelli nelle regioni dense.

2. Si calcola una matrice di dispersione $cov(p_i)$ per p_i utilizzando tutti i punti p_j all'interno di una distanza r_{frame} che limita la grandezza della struttura:

$$cov(p_i) = \sum_{|p_i - p_j| < r_{frame}} w_j (p_j - p_i)(p_j - p_i)^T / \sum_{|p_i - p_j| < r_{frame}} w_j.$$

3. Si calcolano gli autovalori $\{\lambda_i^1, \lambda_i^2, \lambda_i^3\}$ in ordine di decrescente e i loro rispettivi autovalori $\{e_i^1, e_i^2, e_i^3\}$

4. Viene utilizzato p_i come origine e e_i^1, e_i^2, e il loro prodotto vettoriale $e_i^1 \bigotimes e_i^2$ come assi x, y e z rispettivamente per definire un sistema di coordinate 3D F_i appartenente a p_i che è chiamato struttura di riferimento intrinseca.

In questo algoritmo un punto p_i risulta un keypoint se: l'autovalore con il valore assoluto più basso λ_i^3 risulta maggiore di un certo valore soglia, ciò indica che esso possiede nel suo intorno grosse variazioni di coordinate tra i punti nelle tre dimensioni; e il rapporto tra gli autovalori risulta minore di una certa soglia $\lambda_i^2/\lambda_i^1 < \gamma_{21}$ e $\lambda_i^3/\lambda_i^2 < \gamma_{32}$, ciò permette di eliminare i punti che hanno aventi la stessa caratteristica dimensionale in due assi diversi e quindi riduce le ambiguità simmetriche.

3.5.4 NARF: Normal Align Radial Features

Fino ad ora abbiamo descritto metodi per la ricerca di corner 3D. Nel paper [48] è presentato l'algoritmo NARF, Normal Aligned Radial Features, che si occupa sia della detection che della description di keypoints. Questo è l'unico algoritmo 3D presentato che non utilizza in input una nuvola di punti; funziona infatti con *range image*.

Una range image è una comune immagine RGB nella quale l'informazione di distanza di un punto è codificata da un certo colore del pixel corrispondente. Considerando lo spettro della luce visibile, i punti più vicini al sensore saranno viola e i punti più lontani rossi. Siccome si utilizzano principalmente le nuvole di punti, le PCL forniscono delle funzioni in grado di convertire la nuvola di punti in image range.

Il metodo NARF si basa su due principi: i) un keypoints deve trovarsi in una posizione in cui la superficie risulti stabile, per assicurare una stima robusta della normale, e siano presenti sufficienti cambiamenti nelle vicinanze, ii) siccome abbiamo una visione parziale della scena è possibile concentrarsi sui bordi di un oggetto. I contorni di una forma in primo piano saranno abbastanza unici quindi possono essere usati per l'estrazione di punti di interesse. Per questo , si presenta ora il metodo per trovare i bordi di un oggetto nella scena.

Estrazione dei bordi

Un importante passo per ricavare features NARF è l'estrazione di bordi. I bordi appaiono come una discontinuità tra un oggetto e lo sfondo. In questo contesto, si possono definire tre tipi di punti interessanti: i *bordi dell'oggetto*, cioè i punti più esterni che appartengono all'oggetto, i *bordi d'ombra*, i punti sullo sfondo al confine con le occlusioni, e *i punti velo*, che sono i punti interpolati tra i bordi dell'oggetto e i punti ombra. I punti velo sono un fenomeno presente nelle acquisizioni 3D ottenuti tramite lidars.

In figura 3.26 vi è un esempio per comprendere i diversi tipi di punti. Consideriamo un quadrato piano in una nuvola di punti. La superficie chiaramente non fornisce punti interessanti, ma i quattro angoli ne definiscono la forma. Mentre i punti appartenenti ai bordi d'ombra non sono utili, la rilevazione dei punti velo è necessaria, essi non devono essere incorporati nel processo di estrazione features.

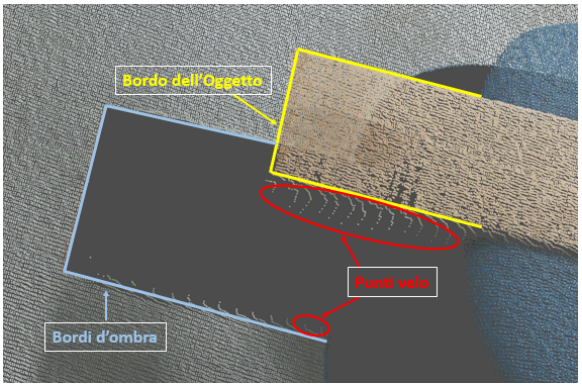


Figura 3.26: Classificazione dei punti vicino ai bordi.

Ci sono diversi indicatori che possono essere utili per la rilevazioni di bordi in una range image, come bruschi cambiamenti della normale o angoli di impatto acuti. In [48] è dimostrato che l'indicatore più importante, che è anche molto robusto al rumore e ai cambiamenti di risoluzione, è il cambiamento della distanza media dei vicini al punto. Per ogni punto allora si prendono in considerazione i suoi vicini locali e:

- si trova la tipica distanza 3D media dei vicini per punti che non sono bordi.
- si utilizza questa informazione per calcolare un valore che rappresenti la possibilità che un certo punto sia parte di un bordo,
- si identifica il tipo di punto tra quelli presentati sopra,
- si estrae la posizione esatta del bordo tramite una soppressione dei non massimi.

Estrazione dei keypoints

Dopo aver definito i bordi presenti nella nuvola di punti si utilizza tale informazione e le loro normali per rilevare i keypoints. Un keypoint stabile necessita di avere nel suo intorno cambiamenti significativi della superficie per essere rilevato nello stessa posizione anche se osservato in diverse prospettive. Questo significa che esistono direzioni dominanti di cambiamento di superficie in quell'area. Per definire i keypoints si usano questi passaggi:

- prendendo in considerazione i vicini locali di ogni punto si determina un valore che rappresenta di quanto cambia la superficie in quel punto e si calcolano le direzioni dominanti, inoltre si determinano i bordi;
- per ogni punto bordo, si calcola un valore d'interesse v che rappresenta $i)$ quanto le direzioni dominanti si discostano dalle altre, $ii)$ quando cambia la superficie in quel punto;
- si fa uno smoothing su tutta la nuvola di punti dei valori trovati per ridurre il rumore;
- tramite una soppressione dei non massimi, si trovano i keypoints finali.

In questa fase il parametro più importante è σ il diametro della sfera attorno al keypoints, che include tutti i punti le cui direzioni principali sono usate per calcolare il valore d'interesse v. Questo è lo stesso parametro che si utilizzerà per calcolare il descrittore.

Il descrittore NARF

Anche se molti descrittori 3D sono invarianti rispetto alla rotazione attorno alla normale, avere disponibile l'informazione di orientazione è utile in vari casi, per esempio quando si hanno per la maggior parte del tempo angoli di rotazione nulli, come nel caso di un robot con ruote. In altri casi, dove invece si utilizza un'unica orientazione, è possibile applicare filtri e supporti per avere un confronto diretto tra features definendo coordinate locali nell'intorno del keypoints. L'idea è simile a quello che accade nel SIFT 2D e nel SURF.

Come sottolinea il nome, il descrittore NARF permette di estrarre un'unica orientazione attorno alla normale. Tramite l'orientazione e la normale del keypoint si definisce una trasformazione a 6 gradi di libertà che lo riporti ad un riferimento. Per calcolare il descrittore NARF inizialmente è necessario un sistema di riferimento locale posizionato lungo la normale alla superficie nel punto, con il quale l'osservatore possa vedere un'area circoscritta attorno al punto. Si sovrappone a quest'area un pattern a stella, dove ogni trave corrisponde a un valore nella

descrizione finale che valuta quanto cambiano i pixels lungo di essa. Si estrae poi dal descrittore un'unica orientazione per rendere poi il confronto tra descrittori invariante alla rotazione.

3.5.5 Tassonomia di descrittori 3D

Prima di addentraci nel mondo dei descrittori 3D, si espone un metodo per la classificazione di questi. É possibile dividere i descrittori 3D in due categorie principali, chiamati *Signatures* e *Histograms*.
I metodi nella prima categoria descrivono la superficie 3D nell'intorno di un dato punto definendo un sistema di riferimento locale e una codifica, in accordo con le coordinate locali, di una o più misure geometriche calcolate individualmente per ogni punto nell'intorno. Le Signatures sono potenzialmente molto descrittive grazie all'uso delle informazioni di spazio, ma piccoli errori nella definizione del sistema di riferimento globale o piccole perturbazioni del tratto codificato possono modificare in maniera significativa il descrittore finale. I principali metodi che fanno parte di questa categoria sono Structural Indexing [49] e 3D Point Fingerprint [50].
Gli Histograms invece descrivono l'intorno del punto codificando in un istogramma contatori di entità precise che soddisfano alcune specifiche caratteristiche, come ad esempio coordinate del punto, curvatura o angoli delle normali. In questo modo non vi è bisogno di definire un sistema di riferimento locale, evitando tutti gli errori da esso derivati, ma viene perso il potere descrittivo comprimendo le informazioni in bins. In altre parole usare un descrittore Signatures o un descrittore Histograms dipende da un trade-off tra il potere descrittivo e la robustezza. I principali metodi che fanno parte di questa categoria sono Point Feature Histogram, Fast Point Feature Histogram e Intrinsic Shape Signatures. In figura 3.27 vi è una schema per comprendere meglio la differenza di classificazione utilizzata.

In questa seconda parte del capitolo descriveremo gli algoritmi sopracitati che sembrano essere più adatti alla descrizione di nuvole di punti 3D.

3.5.6 PFH e PFHRGB: Point Feature Histograms

Il Point Feature Histogram è un descrittore locale proposto nel 2008 da Rusu [51], creatore delle PCL. Ha lo scopo di codificare le proprietà geometriche di k vicini di un punto, generalizzando la curvatura media attorno al punto stesso utilizzando un istogramma multidimensionale di valori. É basato sulla relazione tra i k punti appartenenti all'intorno del punto descritto p_q e le relative normali di superficie. Cerca di catturare le più piccole variazioni di superficie attorno al punto considerato confrontando le direzioni delle normali stimate. L'algoritmo

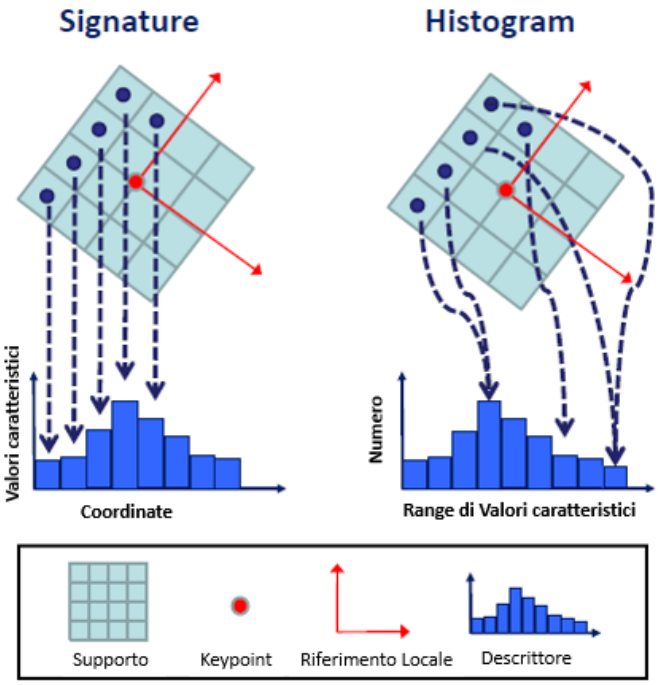


Figura 3.27: Differenza tra le due categorie di descrittori 3D: Signatures e Histograms.

dipende dunque dalla qualità della stima delle normali alla superficie calcolata in ogni punto.

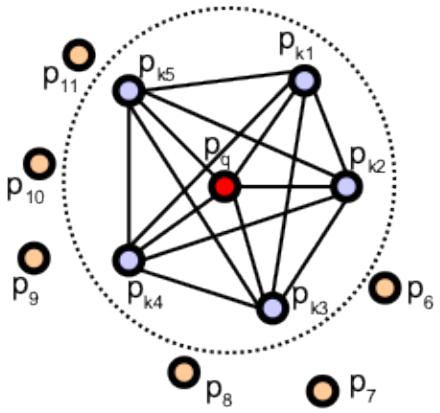


Figura 3.28: Schema delle connessioni per il calcolo dell'istogramma nel descrittore PFH.

Nella figura 3.28 è presentato un diagramma della regione di influenza del calcolo del PFH nel punto p_q, indicato in rosso e posizionato nel centro del cerchio (sfera in 3D) con un raggio r, e tutte le connessioni tra i suoi k vicini all'interno del cerchio, indicati in blu. Il descrittore PFH è quindi un istogramma di relazioni

tra tutte le coppie di punti nell'intorno, che avrà una complessità computazionale di $O(k^2)$. Questo significa che per una nuvola di n punti la complessità sarà $O(nk^2)$.

Per calcolare la differenza relativa tra due punti, source p_s e target p_t e le relative normali n_s e n_t, si definiscono delle coordinate fisse, visibili in figura 3.29, per ognuno dei punti:

$$\begin{aligned}\mathbf{u} &= \mathbf{n}_s \\ \mathbf{v} &= \frac{\mathbf{p}_t - \mathbf{p}_s}{\|\mathbf{p}_t - \mathbf{p}_s\|_2} \times \mathbf{u} \\ \mathbf{w} &= \mathbf{u} \times \mathbf{v}\end{aligned}$$

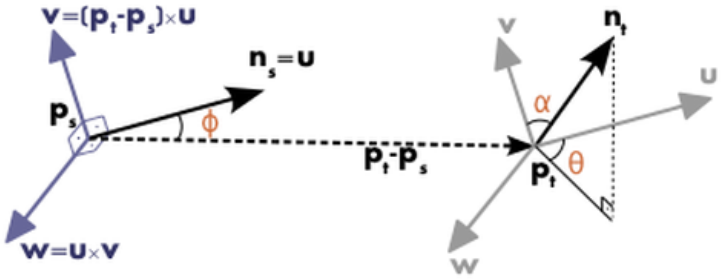


Figura 3.29: Definizione delle coordinate fisse per il calcolo della differenza tra due punti nell'algoritmo PFH.

Utilizzando questi parametri, la differenza tra due normali n_s e n_t può essere espressa come una terna di angoli così descritti:

$$\begin{aligned}\alpha &= \mathbf{v} \cdot \mathbf{n}_t \\ \phi &= \mathbf{u} \cdot \frac{\mathbf{p}_t - \mathbf{p}_s}{d} \\ \theta &= \arctan(\mathbf{w} \cdot \mathbf{n}_t, \mathbf{u} \cdot \mathbf{n}_t)\end{aligned}$$

dove d è la distanza Euclidea tra i punti $\mathbf{p}_s$ e $\mathbf{p}_t$, $d = \|\mathbf{p}_t - \mathbf{p}_s\|_2$. I quattro valori $(\alpha, \phi, \theta, d)$ sono calcolati per ogni coppia di punti all'interno della sfera, in modo da ridurre il numero di valori per l'informazione da 12, cioè 6 per la posizione e 6 per le normali per ogni punto della coppia, in 4. Per creare la rappresentazione finale PFH, le quadruple vengono raggruppate in un istogramma.

Il raggruppamento consiste nel dividere ognuno dei 4 valori presentati in b sottodivisioni e contare il numero di valori che appartengono ad una certa

sottodivisione. Siccome α, ϕ e θ rappresentano gli angoli tra le normali, i loro valori possono essere facilmente normalizzati negli stessi intervalli trigonometrici circolari. L'implementazione PFH utilizza di default 5 suddivisioni e non tiene conto della distanza. Il risultato sarà un istogramma di 125 (5^3) valori di tipo float.

Inizialmente anche questo algoritmo è nato tenendo in considerazioni solamente le normali. Un'informazione importante che il Kinect è in grado di fornire è il colore. Aggiungendo 125 intervalli nell'istogramma, è possibile ottenere anche un confronto tra i colori dei punti appartenenti all'intorno del punto in esame, dando vita così all'algoritmo chiamato PFHRGB. Questa caratteristica può portare benefici per la registrazione di nuvole di punti colorate.

3.5.7 FPFH: Fast Point Feature Histograms

Un anno dopo, nel 2009, Rusu rendendosi conto delle potenzialità del PFH cerca di ottimizzare l'algoritmo mantenendo pressoché inalterato il potere discriminante. Il problema principale è la complessità computazionale troppa elevato, che per una nuvola di n punti e k vicini è nell'ordine di $O(n\dot{k}^2)$.

Questo rappresenta il maggior collo di bottiglia per la registrazione. Nasce così il Fast Point Feature Histogram descritto in [52] che semplifica la complessità computazionale a $O(n\dot{k})$ senza ridurre il potere discriminante del PFH.

L'algoritmo può essere semplificato in due passi:

- Si calcola per ogni punto p_q da caratterizzare, solamente le relazioni tra lo stesso e i suoi vicini, questo passo è chiamato Simplified Point Feature Histogram (SPFH).
- si calcola il valore SPFH per tutti i k vicini e lo si utilizza per pesare l'istogramma finale di p_q:

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k}\cdot SPFH(p_k)$$

dove ω_k rappresenta la distanza Euclidea tra il punto p_q e i suoi vicini p_k. Per capire l'importanza di questo schema di attribuzione di peso, la figura 3.30 presenta la regione di influenza dell'algoritmo applicato su p_q.

Quindi l'algoritmo stima i valori di SPFH del punto p_q creando coppie tra il punto stesso e i suoi vicini, illustrati dalle linee rosse. I valori SPFH trovati di p_q si pesano usando i valori SPFH dei suoi vicini p_k, questo crea il FPFH di p_q. Come mostra il diagramma tramite linee più spesse, alcuni valori di coppia sono calcolati due volte. L'implementazione di default FPFH usa 11 suddivisioni per i parametri α, ϕ, θ, come il PFH infatti, non tiene conto della distanza, gli istogrammi sono

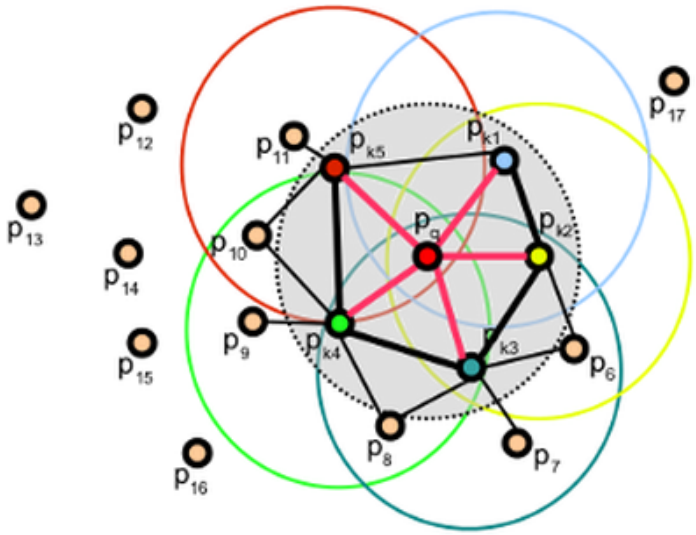


Figura 3.30: Schema delle connessioni per il calcolo dell'istogramma nel descrittore FPFH.

calcolati separatamente e vengono poi concatenati. Il risultato sarà un istogramma di 33 $(3 * 11)$ valori di tipo float.

Le differenze principali tra il PFH e il FPFH sono:

1. il FPFH non interconnette tra loro tutti i vicini di p_q come si può vedere in figura 3.30;
2. il PFH modellizza una precisa e determinata superficie attorno al punto caratterizzato, mentre il FPFH include dei punti addizionali fuori dalla sfera di raggio r, fino ad una distanza massima di $2r$;
3. il FPFH combina i valori SPFH e calcola due volte il valore di coppia tra alcuni vicini.
4. la complessità di FPFH è ridotta, rendendo più vicino la possibilità di utilizzare l'algoritmo in applicazioni real time.
5. l'istogramma risultante è semplificato poiché composto da istogrammi separati, uno per ogni dimensione caratteristica, concatenati insieme.

3.5.8 SHOT e C-SHOT: Signature of Histograms of Orientations

Nella sezione 3.5.5, abbiamo classificato i descrittori 3D in due macrocategorie, Signatures e Histograms. L'algoritmo SHOT, Signature of Histograms of Orientation creato nel 2010 [53], si propone di mantenere sia il potere descrittivo dei Signatures sia la robustezza degli Histograms.

I suoi creatori hanno preso ispirazione dall'algoritmo SIFT 2D , presentato in 3.4.2. Analizzando l'algoritmo SIFT si possono intuire le ragioni del suo successo. In esso vengono utilizzati istogrammi, dalla definizione dell'orientazione locale al descrittore finale, aumentando la robustezza del metodo. Siccome un singolo istogramma calcolato sull'intera nuvola non può essere abbastanza descrittivo, il SIFT conta su insieme di istogrammi locali calcolati su uno specifico set di pixel definito da una griglia, visibile in figura 3.16. L'utilizzo di questa griglia dona al SIFT una struttura simile ai descrittori Signatures. Al contrario, gli istogrammi locali, in cui sono allocati i valori della derivata prima dei gradienti di intensità luminosa, lo avvicinano ai descrittori Histograms.

Basandosi su questa considerazione è nato SHOT. Pur descrivendo il keypoints tramite istogrammi, viene migliorato il potere discriminativo mediante l'introduzione dell'informazione sulla posizione dei punti dell'intorno grazie all'utilizzo di una griglia di supporto sferica 3D. La griglia è allineata agli assi del riferimento locale calcolati ricavando gli autovettori dalla matrice di covarianza M dei k vicini p_i del punto:

$$M = \frac{1}{\sum_{i:d_i \leq R}(R - d_i)} \sum_{i:d_i \leq R} (R - d_i)(\mathbf{p}_i - \mathbf{p})(\mathbf{p}_i - \mathbf{p})^T$$

dove $d_i = \|\mathbf{p}_i - \mathbf{p}\|_2$. Si calcolano allora una serie di istogrammi locali nei volumi 3D definiti dalla griglia sferica di supporto, che rendono discriminante e robusto il descrittore e di difficile collocazione in una delle due macrocategorie sopracitate.

Per ogni istogramma locale, si raccolgono i punti in base al coseno dell'angolo θ_q tra la normale al punto, n_q, e l'asse z del sistema di riferimento locale proprio del punto, z_k. Le ragioni per cui si utilizza il coseno sono due: può essere calcolato velocemente, siccome $\cos(\theta_q) = z_k \cdot n_q$, e un cambiamento del valore $\cos(\theta_q)$ è equivalente a una variazione spaziale di θ_q. Per cui si hanno intervalli più grezzi per le direzioni vicine alla normale di riferimento e intervalli più fini nelle direzioni ortogonali. In questo modo punti con piccole variazioni nelle direzioni ortogonali alla normale, presumibilmente le più importanti, vengono collocati in bins diversi.

Come supporto, si è utilizzata una griglia sferica isotropa divisa in 32 settori, formate da due sfere concentriche, risultanti da 8 divisioni azimuth, 2 in altezza e 2 in distanza Euclidea, come si può vedere in figura 3.31. Gli istogrammi sono composti da 11 bin, per questo il descrittore avrà una lunghezza totale di 352 valori, 11×32.

Anche in questo caso si è tenuta in considerazione la possibilità di sfruttare al meglio l'informazione dei sensori RGB-D. Come proposto in figura 3.32, due istogrammi vengono calcolati per ognuno dei volumi ricavati tramite il supporto sferico, uno che tiene conto delle informazioni di forma e l'altro della texture; i

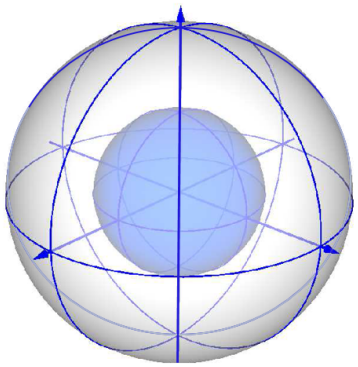


Figura 3.31: Griglia sferica di supporto all'algoritmo SHOT.

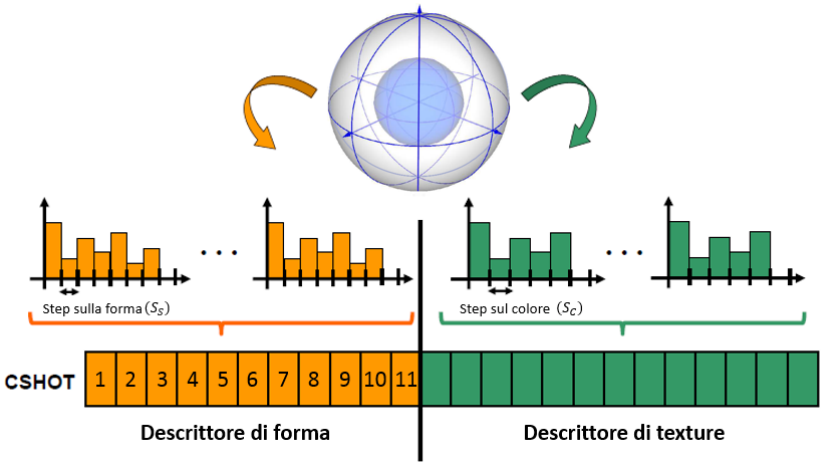


Figura 3.32: Composizione del descrittore SHOT.

due istogrammi vengono poi concatenati insieme per formare un unico descrittore $D(k)$ del punto k:

$$D(k) = D_{shape}(k) \cup D_{texture}(k)$$

Il descrittore di forma viene ricavato come in precedenza. Per quanto riguarda il descrittore di texture, le informazione colore dei punti non è caratterizzata tramite RGB, ma tramite lo spazio colore CIELab poiché risulta indipendente dal dispositivo utilizzato. Il descrittore finale avrà lunghezza di 1344 valori, 352 per la forma e 992 per il colore.

Capitolo 4

Prove e analisi degli algoritmi

Questa sezione illustra il setup, lo svolgimento ed i risultati delle prove sperimentali svolte al fine di eseguire una analisi delle prestazioni dei diversi argomenti precedentemente descritti. Vengono analizzate e comparate le varie tipologie di ICP (Iterative Closest Point), sia come metodo di allineamento che come metodo di rifinitura dell'allineamento di nuvole di punti. Successivamente vengono confrontati rilevatori di keypoint 2D e 3D presentati, per analizzarne la robustezza e la velocità di computazione. Infine si confronteranno i carichi computazionali per ricavare le differenti features 3D.

Dopo aver analizzato e scomposto i diversi metodi di registrazione è necessaria una valutazione degli stessi. Poiché la registrazione manuale è affetta dall'errore umano, concentriamo maggiormente l'attenzione sui metodi automatici: registrazione con ICP e features based.

Tutte le prove di valutazione sono state effettuate utilizzando un computer con Intel Core i7-5500U dual core da $2,4$ GHz, 6 GB di RAM e una scheda grafica AMD Radeon R5 M330.

4.1 ICP

L'algoritmo ICP è ampiamente utilizzato per l'allineamento geometrico dei modelli tridimensionali. Nel capitolo precedente questo algoritmo è stato scomposto in quattro fasi. Per ogni fase si sono presentate diverse scelte che caratterizzano e danno origini a varianti di ICP con caratteristiche qualitative differenti.

Tuttavia, per comprendere l'algoritmo, uno studio esclusivamente teorico non basta. É stato implementato un software per la comparazione delle diverse tipologie di ICP che le librerie PCL consentono di utilizzare. Questo allo scopo di valutare

in dettaglio le prestazioni degli algoritmi in termini di carico computazionale e di accuratezza di allineamento.

4.1.1 Descrizione software implementato

In questo paragrafo, viene presentata la parte di software dedicata alla valutazione delle prestazioni dell'ICP. Nell'interfaccia utente sono presenti diverse opzioni che permettono il completo controllo dei parametri dell'ICP. L'interfaccia è composta da:

1. Selezione delle nuvole: due finestre in cui è possibile scegliere la nuvola source e la nuvola target che vengono trattate dall' ICP.

2. Selezione e caratterizzazione dell'ICP: viene scelto il tipo di ICP da utilizzare e i suoi parametri interni. In questa finestra sono selezionabili gli ICP già implementati nelle PCL oppure, scegliendo la tipologia "Custom", è possibile aprire la finestra di personalizzazione.

3. Personalizzazione: permette di cambiare le singole fasi interne dell'ICP già descritte, se nella finestra precedente l'utente decide di selezionare la tipologia "Custom".

4. Salvataggio: è possibile salvare su file di testo, per ogni iterazione, l'errore di allineamento point-to-point tra le nuvole e i tempi di computazione, entrambi fondamentali per il confronto.

5. Calcolo: applica l'algoritmo scelto o sulle due nuvole impostate al punto uno o su una serie di nuvole di punti caricate in precedenza nel programma.

Nella finestra 2 della figura 4.1 sono presenti dei parametri, utilizzati dalle PCL, che permettono di specificare criteri di termine per l'ICP. In particolare è possibile impostare nella casella "How to Exit?" due opzioni.

Tramite comando "*All Termination Criteria*" l'ICP termina se è verificato almeno uno dei tre criteri, discussi nel capitolo precedente, con i parametri visualizzati. Tramite "*Iteration*", non visibile nell'immagine, l'ICP termina raggiunto il numero di iterazioni impostate.

Nella tabella 4.1 sono presenti tutte le possibili combinazioni di ICP che possono essere impostati per personalizzare ogni singola fase.

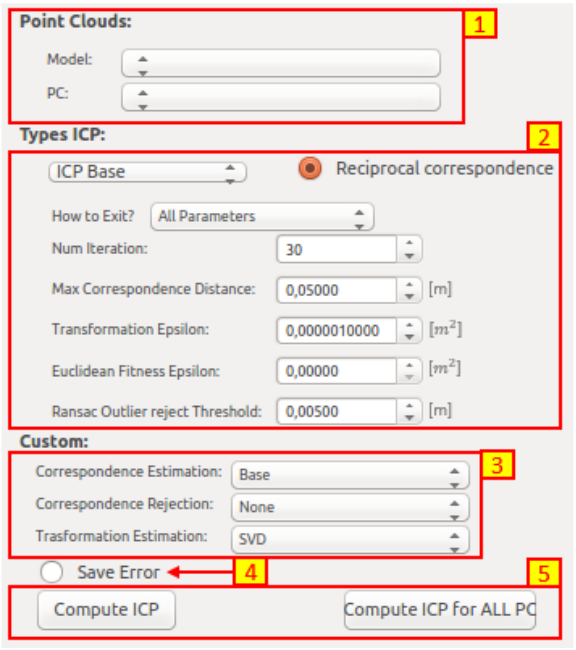


Figura 4.1: Interfaccia utente per la gestione dell'ICP.

Personalizzazioni dell'ICP.

Correspondence Estimation	Base
	Back Projection
	Normal Shooting
	Organized Projection
Correspondence Rejection	Distance
	Median Distance
	One To One
	Sample Consensus
	Organized Boundary
Transformation Estimation	SVD
	LM
	Point To Plan LLS
	Generalized

Tabella 4.1: Elenco degli algoritmi utilizzabili per la personalizzazione dell'ICP.

4.1.2 Descrizione prove sperimentali dell'ICP

Durante la sperimentazione sono state provate tutte le possibili combinazioni verificando come ciascuna riesca a raggiungere la convergenza. La sostanziale differenza sta nel tempo impiegato per l'operazione.

Tale differenza è attribuibile alla tipologia di *Transformation Estimation* utilizzata. Per questo motivo, la sperimentazione è stata concentrata sullo studio di ICP con struttura simile, variando unicamente l'algoritmo nella sua ultima fase.

Struttura degli algoritmi

Quanto segue è stato pensato con l'idea che:

- Nessuna rimozione dei punti nella parte di selezione: l'intenzione è quella di trattare nuvole di punti in uscita diretta dal sistema di acquisizione, senza un preprocessing dei dati entranti nell'ICP.
- Correspondence Estimation base: viene utilizzato l'algoritmo K-d tree e vengono applicate le corrispondenze tra ogni punto della nuvola source e il punto appartenente alla nuvola target che ha distanza euclidea minima da esso.
- Correspondence Rejection con RANSAC: si eliminano alcune corrispondenze se la distanza euclidea tra i punti dopo l'applicazione della trasformazione ottima sulla nuvola di punti source è maggiore di una certa soglia, che fissiamo conservativamente di un ordine di grandezza rispetto alla densità spaziale della nuvola (1 cm).

Come accennato, la diversificazione tra gli ICP si avrà nella fase di Transformation Estimation. Verranno proposti i metodi più utilizzati nella letteratura:

- Transformation Estimation SVD: minimizza l'errore metrico point-to-point utilizzando un metodo basato sulla decomposizione ai valori singolari (SVD).
- Transformation Estimation LM: minimizza l'errore metrico point-to-point basandosi sulla stima attraverso i metodi di Levenberg-Marquardt.
- Transformation Estimation PointToPlaneLLS: un metodo iterativo che accumula gli errori point-to-plane di tutte le corrispondenze in una matrice A e stima la rotazione e la traslazione tramite i minimi quadrati risolvendo un sistema lineare della forma $A^T\ Av = A^T b$, con v un vettore di sei elementi che rappresentano i parametri di trasformazione ottimali da trovare.

Oltre a questi ICP, aventi struttura simile, si utilizzerà un algoritmo implementato nelle PCL, proposto da Segal [36], che esegue il confronto tra le covarianze dell'intorno dei punti per poter allineare le superfici piuttosto che i punti stessi. Questo algoritmo è chiamato *Generalized ICP*.

Dataset

In linea con la volontà di realizzare un sistema di scansione di ampie scene, il dataset di prova è composto da acquisizioni con il Kinect V2 di due scene indoor. Il primo dataset mostra tre piani ortogonali tra loro e degli oggetti che possono essere trovati in un ambito indoor. Il secondo, invece, è formata principalmente da piani paralleli ed ortogonali all'asse z (in blu) uscente dallo strumento di scansione, telecamera 3D. Entrambe sono visibili in figura 4.2 e verranno chiamate rispettivamente scena A e scena B. Per ogni scena sono state effettuate due acquisizione nella stessa posa che saranno indicate con il pedice 1, la prima, e con il pedice 2, la seconda.

Le acquisizioni effettuate sulla stessa scena e nella stessa posa sono difficilmente distinguibili ad occhio nudo, ma non identiche. Pur inquadrando la stessa scena ogni singolo punto della nuvola avrà valori leggermente diversi a causa della incertezza di misura che lo caratterizza e rende unica l'acquisizione, la quale nel Kinect V2 è composta da 217088 punti (512×424).

Figura 4.2: Scene utilizzate per le prove ICP: la scena A, a sinistra e la scena B, a destra.

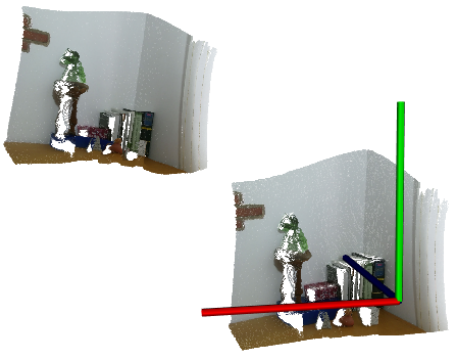
Struttura delle prove

In totale sono state effettuate quattro prove con nuvole di ingresso diverse; si ricorda che si definisce come "source" la nuvola di punti da allineare e come "target" la nuvola di riferimento. In tutte le prove viene rototraslata la nuvola source di

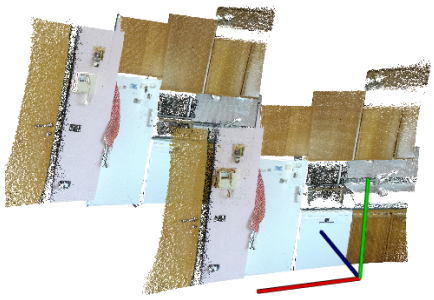
una matrice di R, derivata prima dalla rotazione di un angolo di 10^o rispettivamente attorno all'asse x, y e z del sistema di riferimento comune ad entrambe, poi traslando la nuvola di un metro lungo gli stessi assi. Sono state applicate tali trasformazioni in modo che le nuvole in ingresso all'algoritmo abbiano pose differenti sia per quanto riguarda la rotazione che la traslazione. Uno schema delle prove eseguite è visibile in tabella 4.2, mentre nella figura 4.3 è possibile osservare le posizioni iniziali delle nuvole. Si identifica con la notazione A_1 la prima nuvola acquisita nella scena A, A_2 la seconda nuvola acquisita nella scena A e in maniera simile si identificano B_1 e B_2.

		Source	**Target**
Scena a angolo	Prova 1	A_1	A_1
	Prova 2	A_2	A_1
Scena a piani paralleli	Prova 3	B_1	B_1
	Prova 4	B_2	B_1

Tabella 4.2: Tabella riassuntiva delle prove per la valutazione di vari algoritmi ICP.



(a) Posizione iniziale delle nuvole in ingresso alle prove 1 e 2.



(b) Posizione iniziale delle nuvole in ingresso alle prove 3 e 4.

Figura 4.3: Setting iniziale delle prove.

Date due nuvole di punti P e Q, sia q_k il punto appartenente a Q che ha distanza minima dal punto p_k appartenente a P. L'errore di allineamento E_{all} è calcolato come:

$$E_{all} = \sqrt{\frac{\sum_{k=1}^{N}(\| p_k - q_k \|)^2}{N}}$$

dove (p_k, q_k) è la k-esima delle N coppie di corrispondenza tra la nuvola P e la Q. Tale errore rappresenta la radice dell'errore quadratico medio delle distanze tra tutte le coppie di corrispondenza.

Ad ogni iterazione di ICP è stato calcolato e salvato l'errore di allineamento raggiunto tra le due nuvole e il tempo impiegato per raggiungerlo. Confrontando gli errori, i tempi e il numero di iterazioni dei diversi ICP è stato possibile ricavare due grafici per ogni prova. Si è messo successivamente in relazione l'errore di allineamento dei diversi ICP con il tempo impiegato per raggiungerlo nel primo grafico, e con il numero di iterazione nel secondo.

4.1.3 Risultati

In ogni grafico ottenuto, visibili nelle figure 4.4 e 4.5, sono rappresentate quattro curve che si riferiscono ai quattro ICP analizzati. Per poter visualizzare meglio le variazione d'errore, i grafici sono in scala logaritmici sull'asse y. Inizialmente si concentri l'attenzione sui grafici che rappresentano l'errore in funzione del tempo. L'efficienza di questi algoritmi può essere discussa attraverso alcune considerazioni effettuate confrontando tre valori caratteristici: valore di convergenza, tempo di convergenza e pendenza della curva.

Valore di convergenza dell'errore

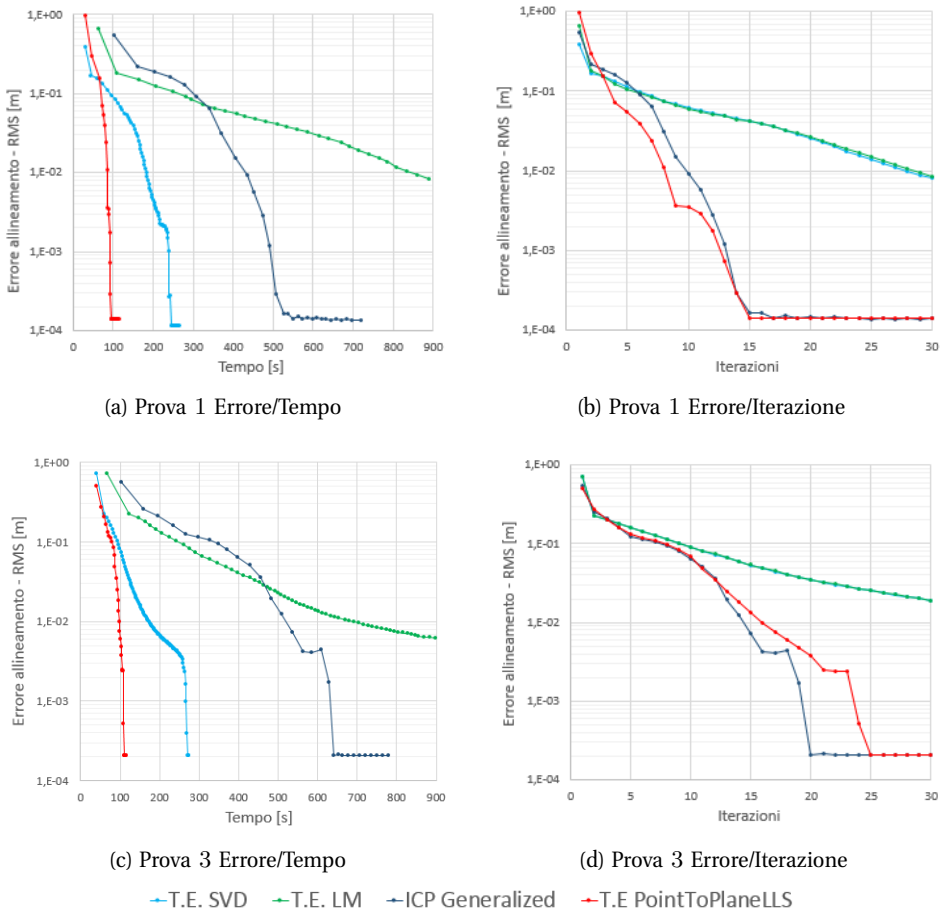
Innanzitutto è possibile notare come i valori di convergenza degli errori di allineamento dei diversi algoritmi sia tra loro comparabile. Da questo si deduce che nessun metodo è più accurato di un altro. In verità, nei grafici proposti non risulta possibile osservare il comportamento a convergenza dell'ICP con il metodo Transformation Estimation LM poiché il tempo di convergenza risultava troppo elevato da graficare. Tuttavia anche quest'ultimo risulta avere un valore di convergenza simile agli altri.

Il valore di convergenza dipende unicamente dalle nuvole di punti in ingresso. Le prove effettuate aventi in ingresso esattamente le stesse nuvole, raggiungono infatti un valore di convergenza minore di quelle effettuate con nuvole diverse.

Tempo di convergenza

Mentre tra metodi diversi i valori di convergenza dell'errore risultano comparabili, i tempi di convergenza dipendono dall'efficacia dell'algoritmo. In tutte le prove il tempo di convergenza dell'ICP che utilizza il metodo di transformation estimation *PointToPlaneLLS*.

Questo algoritmo si basa sull'utilizzo delle normali, perciò è necessario introdurre nella stima del tempo impiegato dal processo anche il tempo per il calcolo delle stesse. Utilizzando l'algoritmo *IntegralImageNormalEstimation*, implementato nelle PCL e ottimizzato per le nuvole di punti proiettabili, il tempo impiegato per il calcolo delle normali è di circa 40 millisecondi, valore complessivamente trascurabile.



(a) Prova 1 Errore/Tempo

(b) Prova 1 Errore/Iterazione

(c) Prova 3 Errore/Tempo

(d) Prova 3 Errore/Iterazione

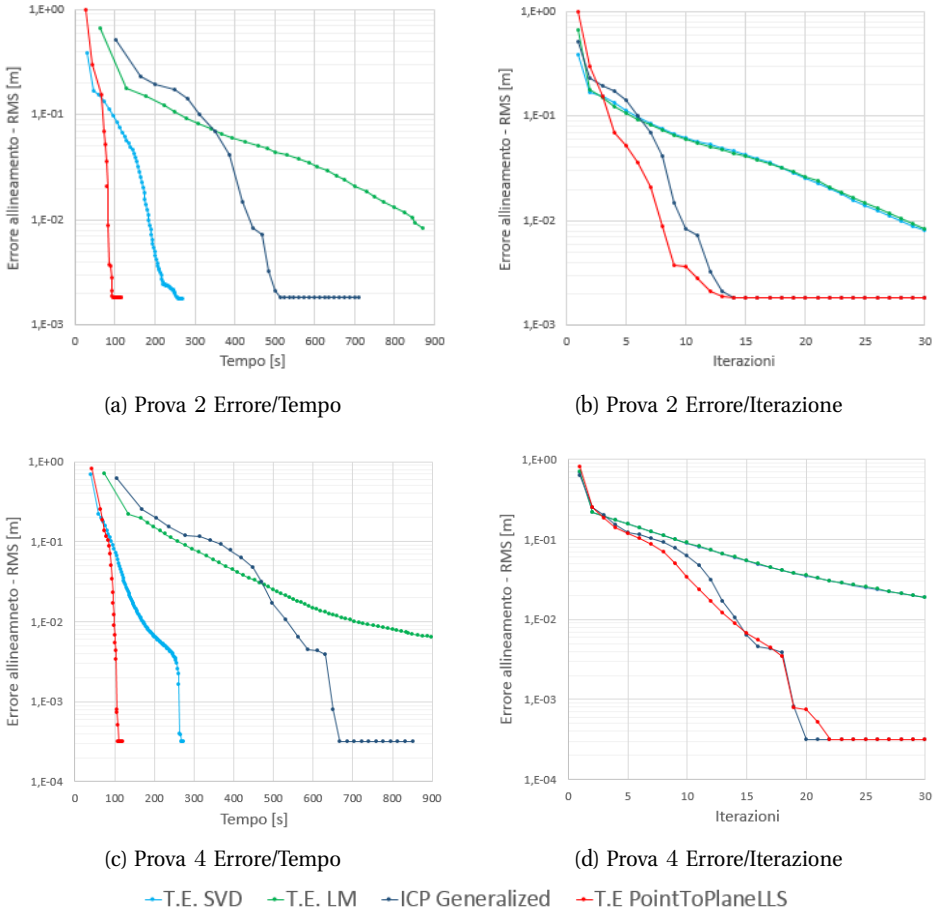
Figura 4.4: Risultati prove ICP con nuvole in ingresso identiche ma rototraslate.

Pendenza della curva

La pendenza di convergenza della curva nei grafici a sinistra definisce di quanto rapidamente l'errore migliora nel tempo. Si può notare come l'unico algoritmo ad avere una pendenza minore sia lo stesso che va a convergenza più lentamente.

Viene rilevata inoltre la massima pendenza di ogni curva, tra le iterazioni prima che l'algoritmo arrivi a convergenza. Ciò significa che la rapidità dell'algoritmo dipende da quanto risultino vicine le nuvole. Se dunque le nuvole risultano parzialmente allineate, l'ICP raggiunge la convergenza in poco tempo. Per questo l'ICP verrà utilizzato principalmente come algoritmo per la rifinitura dell'allineamento.

Analizzando invece i grafici errore/iterazioni si nota come gli algoritmi ICP che utilizzano come stima della trasformazione i metodi SVD e Levemberg Marquardt risultino sovrapponibili, poiché entrambi stimano la trasformazione minimizzando



(a) Prova 2 Errore/Tempo

(b) Prova 2 Errore/Iterazione

(c) Prova 4 Errore/Tempo

(d) Prova 4 Errore/Iterazione

Figura 4.5: Risultati prove ICP con nuvole in ingresso diverse ma acquisite dalla stessa posa.

lo stesso errore metrico point to point, anche se con due metodi differenti.

Viene introdotta ora una considerazione sull'errore introdotto da Kinect V2. Per maggior chiarezza, definiamo e_{Pn} il valore di convergenza dell'errore di allineamento della prova n. Una stima dell'errore di acquisizione introdotto dal Kinect V2 e_k^A tra le due prove della scena A può essere calcolata come:

$$e_k^A = |e_{P2} - e_{P1}|.$$

e e_k^B in modo analogo. É normale aspettarsi che risulti $e_k^A \simeq e_k^B$. Dalle prove però si nota come l'errore introdotto dal Kinect V2 nella scena A sia molto maggiore rispetto a quello introdotto dalla scena B, $e_k^A >> e_k^B$.

(a) (b)

Figura 4.6: Schema rappresentante il percorso compiuto dai raggi luminosi in presenza di una scena in cui si verificano riflessioni multiple (a) e acquisizione distorta relativa ad una geometria concava (b).

La differenza tra questi valori è da ricercare in un noto problema di misura relativo alle telecamere a tempo di volo: le riflessioni multiple. Questo fenomeno ha luogo in presenza di geometrie concave schematizzabili attraverso un diedro costituito da due piani secanti, come possono essere semplificati i due muri presenti nella scena A. L'errore di misura si verifica poiché parte della luce IR, incidente su ciascuna delle due superfici, ritorna alla telecamera solo dopo aver compiuto una seconda riflessione sul piano adiacente, percorrendo così, come mostrato in figura 4.6a, un tragitto più lungo di quello ideale e generando distorsioni (4.6b) nella nuvola di punti acquisita.

4.2 Keypoint Detectors

Il costo computazionale dei descrittori è generalmente elevato. Ha poco senso quindi estrarre i descrittori da tutti i punti della nuvola. I rilevatori di keypoints sono utilizzati per selezionare dei punti interessanti della scena, che vengono poi descritti, in modo da ridurre i tempi per il calcolo delle corrispondenze.

Lo scopo di queste prove è valutare e comparare la robustezza dei principali, e più promettenti, keypoints detector 2D e 3D utilizzando un dataset acquisito con Kinect V2.

4.2.1 Descrizione prove sui Keypoint Detector

É importante sottolineare che lo scopo di queste prove non è valutare la robustezza degli algoritmi di rilevazione di punti in sè ma la robustezza degli algoritmi eseguiti su acquisizioni fornite dal sensore Kinect V2. Per questo motivo, dopo

aver presentato i keypoint detectors confrontati e il dataset utilizzato, introdurremo il concetto di ripetibilità del keypoint.

Keypoint Detectors confrontati

Nel capitolo 3 si sono introdotti, per completezza, i keypoint detector più utilizzati che hanno segnato il percorso evolutivo della visione artificiale. Per queste prove, invece, sono stati presi in considerazione i keypoint detectors 2D e 3D più promettenti e utilizzati in questi ultimi anni.

Per quanto riguarda l'ambito 2D sono stati confrontati il *BRISK* e l'*AGAST*, mentre in ambito 3D gli algoritmi *Harris 3D*, *Harris 6D*, *SIFT 3D* e *ISS 3D*.

Dataset

Il dataset è composto da acquisizioni effettuate con il Kinect V2 di due scene indoor. La prima scena è molto simile a quella utilizzata durante le prove per il confronto tra ICP; in essa sono presenti tre piani ortogonali tra loro e degli oggetti che possono essere trovati in un ambito indoor. La seconda scena raffigura nella parte superiore dei piani e degli spigoli, mentre in quella inferiore una "free form", caratterizzata dalla assenza di spigoli ma descrivibili da una superficie continua (un divano). La scelta di questa scena è motivata dal cercare di mettere in difficoltà i detector 3D, poiché essi si basano sulla ricerca di corner. Entrambe le scene sono visualizzate in figura 4.7. Come durante le prove dell'ICP, per ogni scena sono state effettuate due acquisizione nella stessa posa.

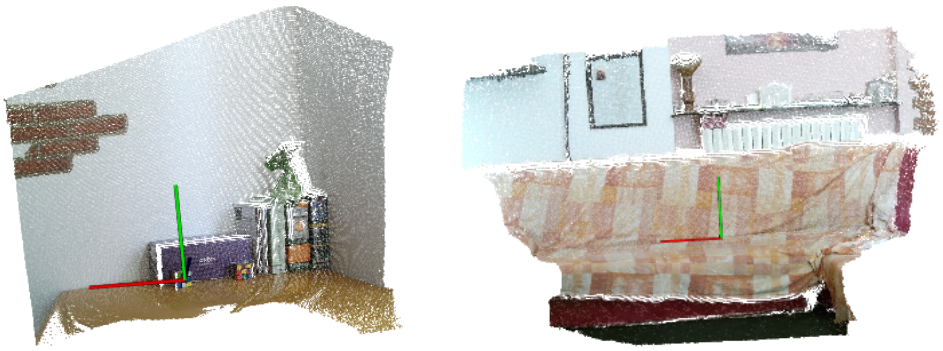


Figura 4.7: Scene utilizzate per le prove per la valutazione della ripetibilità dei keypoint: la scena 1, a sinistra e la scena 2, a destra.

Struttura delle prove

Per il confronto di keypoint detector utilizziamo lo schema rappresentato in figura 4.8. Per ogni scena, dopo aver effettuato con il Kinect V2 due acquisizione con la stessa posa, ne vengono calcolati i keypoint tramite lo stesso tipo di keypoint detector e con gli stessi parametri. Ragionevolmente, più le nuvole di punti contenenti i keypoint risultano simili tra loro, maggiore sarà la ripetibilità del keypoint detector.

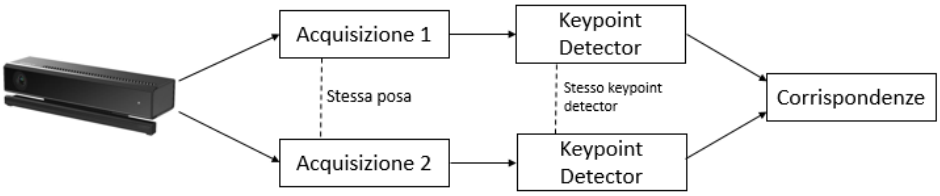


Figura 4.8: Schema per il confronto di keypoint.

Sono state calcolate le corrispondenze tra i keypoints estratti dalla prima acquisizione e quelli estratti dalla seconda acquisizione, tramite una distanza euclidea tra i punti. Per ogni keypoint k_i^1 appartenente alla prima nuvola, viene trovato, tramite un K-d tree, il punto spazialmente più vicino appartenente alla seconda k_j^2 e impostata una corrispondenza $c(i, j)$. A questo punto è necessario descrivere il concetto di ripetibilità.

La capacità di ottenere lo stesso set di keypoints in istanze differenti di un particolare modello è definita come ripetibilità. Essa è la caratteristica più importante per una rilevazione robusta di keypoints. Una scarsa ripetibilità può essere determinata dal rumore di misura, dal cambio di punto di vista o dalla presenza di occlusioni tra due diverse acquisizioni. Nelle prove effettuate non sono presenti ne occlusioni ne cambi di posa, per questo la ripetibilità di ogni algoritmo studiato è influenzato unicamente dall'incertezza di misura.

Il keypoint i-esimo estratto dalla prima acquisizione k_i^1 è detto ripetibile se la distanza $d(k_1^1, k_j^2)$ dal keypoint j-esimo più vicino estratto dalla seconda acquisizione, k_j^2, risulta minore di una certa distanza soglia ϵ imposta, $d(k_1^1, k_j^2) < \epsilon$.

É stata definita la ripetibilità assoluta e relativa dell'intero algoritmo in questo modo. Sia C una matrice $n \times m$, dove n è il numero di keypoints estratti dalla prima acquisizione e m il numero di keypoints estratti dalla seconda. L'elemento $C(i, j)$ della matrice ha valore 1 se tra k_i^1 e k_j^2 vi è una corrispondenza $c(i, j)$ e se i risulta ripetibile, 0 altrimenti. La ripetibilità assoluta di un keypoint detector è definita tramite la formula:

$$r_{abs} = \sum_{\substack{0<i<n-1 \\ 0<j<m-1}} C(i,j)$$

mentre la ripetibilità relativa è data da:

$$r_{rel} = \sum_{\substack{0<i<n-1 \\ 0<j<m-1}} C(i,j) \cdot \frac{1}{n}$$

In altre parole, la ripetibilità assoluta di un algoritmo è il numero di keypoints tramite lo stesso che risultano ripetibili; mentre la ripetibilità relativa ha un valore compreso tra 0 e 1 ed è un rapporto tra il numero di keypoints che risulta ripetibile e il numero di keypoint della prima acquisizione.

Dalle formule di ripetibilità è intuibile come i valori sono dipendenti dall'ordine delle nuvole prese in considerazione. Per descrivere meglio il problema esageriamo il concetto. Se nella prima nuvola di punti viene estratto un solo keypoint e nella seconda mille, si avrà $r_{rel} = 1$, mentre scambiando l'ordine delle nuvole si avrà $r_{rel} = 0,001$. Per questo motivo, le prove sono sempre state svolte in modo da avere in prima posizione l'acquisizione che risulta avere meno keypoints.

4.2.2 Descrizione software

In questo paragrafo, descriviamo il software dedicato al confronto dei keypoint detectors 2D e 3D. Dividiamo l'implementazione in due passaggi: la computazione dei keypoints e la valutazione dei keypoints.

Nell'interfaccia per la computazione di keypoints, in figura 4.9, sono distinguibili due aree. La prima area è dedicata alla scelta della tipologia di keypoint da utilizzare, mentre la seconda è dedicata alla scelta dei parametri da impostare per quella stessa tipologia. Per un corretto funzionamento è necessario scegliere accuratamente i parametri, poiché la variazione di questi influenza notevolmente le prestazioni di ogni tipologia di keypoint.

Un confronto tra diversi keypoint detectors risulta difficile per due motivi. Il primo deriva dal fatto che risulta impossibile trovare delle relazioni dirette tra i parametri di ognuno di essi. Questi, infatti, agiscono su strutture algoritmiche differenti. Per porre rimedio a questo problema è stato ricercato per ogni keypoint detector un setting ottimo di parametri svolgendo numerose prove su diverse nuvole di punti, i quali sono stati inseriti nel software come valori di default. In questo modo l'utente non ha necessariamente bisogno di comprendere a pieno tutti i diversi parametri (18 per questa fase) rendendo di fatto più semplice l'utilizzo del software.

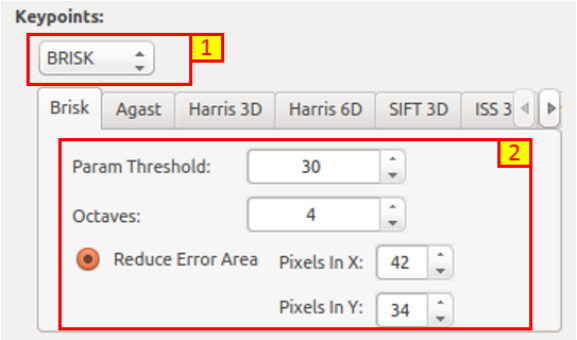


Figura 4.9: Interfaccia utente per la computazione dei keypoints.

Il secondo motivo è dovuto dalla presenza di keypoints detector 2D e 3D. I primi infatti necessitano in ingresso un'immagine bidimensionale e restituiscono una serie di punti bidimensionali, i secondi invece utilizzano nuvole di punti. Per questo motivo, in letteratura [54] [55] [55] si è sempre preferito dissociare il confronto tra keypoint detector 2D e 3D.

In questa tesi i ketpoints 2D e 3D vengono comparati tra loro servendosi della possibilità del Kinect di restituire nuvole di punti organizzate e proiettabili. Una nuvola di punti organizzata ha una struttura simile ad un'immagine (o matrice), in cui i dati sono divisi in righe e colonne, mentre una nuvola di punti proiettabile ha una relazione diretta in base al modello della telecamera pinhole tra gli indici (u, v) di un punto della nuvola organizzata e i propri valori 3D x, y e z che lo definiscono nello spazio rispetto ad un sistema di riferimento.

Sono state implementate nel software delle funzioni che permettono, sapendo i parametri interni della telecamera, di trovare per ogni punto della nuvola il corrispettivo punto nella matrice bidimensionale dell'immagine, e viceversa. Dopo aver ricavato dalla nuvola di punti acquisita un'immagine bidimensionale, avente necessariamente la stessa risoluzione, essa viene passata in ingresso ad un keypoint detector 2D. Questo restituirà una serie keypoint bidimensionali indicizzati, i quali verranno poi proiettati nel 3D, ricavando così una nuvola di keypoint 3D.

Per la fase di valutazione della ripetibilità sono state create tre funzioni visibili in figura 4.10:

1. Evaluate: dopo aver calcolato le corrispondenze, restituisce a schermo la ripetibilità assoluta e relativa del algoritmo per una certa distanza soglia ϵ impostabile, facendo scorrere tutti i keypoint ricavati dalla prima acquisizione e contando quante distanze tra le rispettive corrispondenze risultano minori di ϵ.

2. Histogram: recupera dai parametri impostati il valore "bin" e la distanza

massima "max range" espressa in metri e restituisce un istogramma con l'asse orizzontale compreso tra zero e "max range" e diviso con il numero di bin impostati. Ogni singolo bin avrà un'ampiezza orizzontale X calcolabile tramite la formula e $X = (max\ range)/bin$ e verticale Y in base al numero di distanze, tra i punti di corrispondenza, che cadono nel dato bin.

3. Graph: dopo aver calcolato le corrispondenze, restituisce un grafico di ripetibilità assoluta e relativa dividendo l'asse x delle distanze, in intervalli di grandezza impostabile.

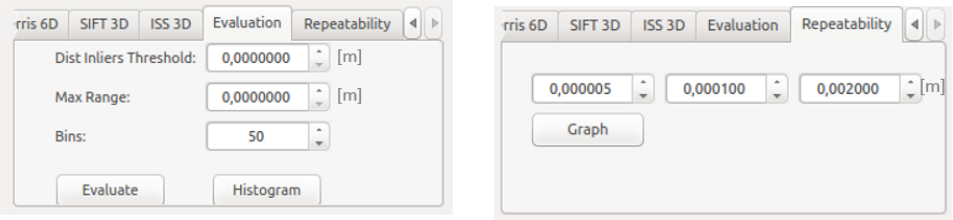


Figura 4.10: Interfaccia utente per la valutazione dei keypoints.

4.2.3 Risultati

In primo luogo analizziamo le tabelle 4.3 e 4.4 in cui sono riportati il numero di keypoint estratti e la somma dei tempi impiegati dal processo di estrazione, per entrambe le scene e per tutti i keypoint implementati.

	Numero K1	**Numero K2**	**Tempo [ms]**
BRISK	934	965	42
AGAST	638	660	22
Harris 3D	1181	1216	6196
Harris 6D	454	432	50596
SIFT 3D	710	675	13219
ISS 3D	253	241	19064

Tabella 4.3: Numero di keypoint e tempo impiegato per il loro calcolo per le nuvole della scena 1.

Dal numero di keypoint trovati non si possono fare importanti considerazioni. Questo valore, oltre ad essere largamente influenzato dai parametri impostati, non è un fattore importante per un buon keypoint detector, poiché non fornisce un'idea di quanto ogni singolo keypoint sia ripetibile. É preferibile, infatti, avere

	Numero K1	Numero K2	Tempo [ms]
BRISK	1197	1179	47
AGAST	723	711	17
Harris 3D	700	747	8606
Harris 6D	238	235	59136
SIFT 3D	863	888	19974
ISS 3D	825	855	22839

Tabella 4.4: Numero di keypoint e tempo impiegato per il loro calcolo per le nuvole della scena 2.

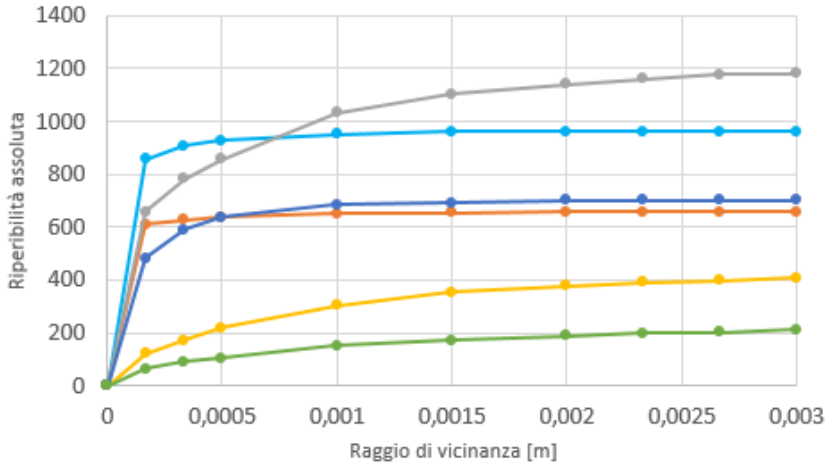
pochi keypoints ripetibili, che viceversa, in quanto, come si vedrà nelle prove della prossima sezione, il tempo computazionale di qualunque keypoint descriptor e della ricerca di corrispondenze tra features, dipende direttamente dal numero di punti descritti.

Analizzando, invece, i tempi di computazione dei diversi algoritmi si nota come ci sia una disparità enorme tra i keypoint detectors 2D e 3D. Come già detto, il valore riportato consiste nella somma dei tempi del processo di estrazione di entrambe le nuvole espressa in millisecondi. L'estrazione di keypoints tramite algoritmi 3D impiega un tempo nell'ordine dei secondi, mentre con algoritmi 2D sono sufficienti alcuni millisecondi. Pur aggiungendo il tempo computazionale per la proiezione, alcune decine di millisecondi, il tempo per l'estrazione di keypoint 2D risulta quindi notevolmente inferiore.

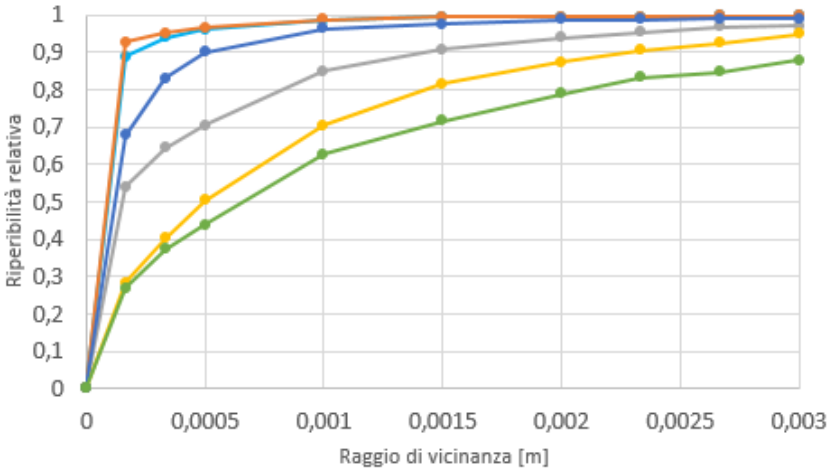
L'analisi principale sui keypoint detectors è stata effettuata valutandone la ripetibilità. Vengono ricavati, tramite la funzione implementata "Graph", i grafici di ripetibilità, assoluta e relativa, in funzione della distanza di soglia ϵ, per entrambe le scene prese in considerazione. I grafici sono riportati nelle figure 4.11 per la scena 1 e 4.12 per la scena 2.

Si concentri l'attenzione sui grafici che rappresentano la ripetibilità relativa in quanto, come già spiegato, il numero di keypoint estratti non è una caratteristica fondamentale ed è mutabile a seconda dei parametri. É evidente come l'errore di misura introdotto dal Kinect V2 nelle due diverse acquisizioni della stessa posa, riduca drasticamente la ripetibilità dei keypoint detectors 3D. Prendendo per esempio una distanza di soglia $\epsilon = 0,001[m]$, la riproducibilità relativa dei detectors 3D, ad eccezione del SIFT 3D, ha valori compresi tra $0,45$ e $0,7$. Per questo motivo si scarta l'ipotesi di utilizzare Harris 3D, Harris 6D e ISS3D come keypoints detector, per acquisizione effettuate con il sensore Kinect V2.

Al contrario, SIFT 3D risulta avere un'ottima ripetibilità relativa, che, per una distanza soglia $\epsilon = 0,001[m]$, è di circa $0,98$. Tale algoritmo allora risulta essere il migliore tra i keypoints detector 3D studiati anche se il suo tempo di computa-



(a) Scena 1: Ripetibilità assoluta.

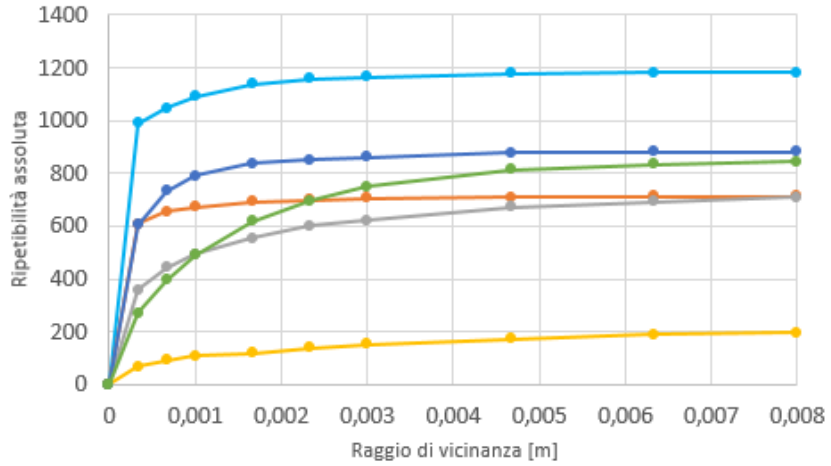


(b) Scena 1: Ripetibilità relativa.

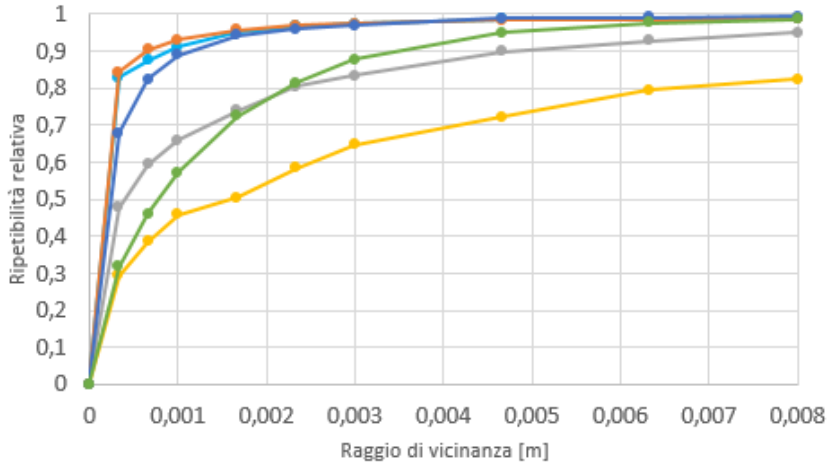
Figura 4.11: Misura della ripetibilità assoluta, in (a), e relativa, in (b), per la prima scena.

zione, di circa 10-15 secondi, non è adatto per applicazioni real time, come invece risultano essere i keypoint detectors 2D.

Analizzando le curve che caratterizzano i keypoints 2D si può notare come la ripetibilità relativa sia molto vicina al valore massimo. Tolta l'incertezza di misura strumentale, che garantisce una distanza minima tra due corrispondenze, i keypoints ricavati dalle due acquisizioni risultano essere gli stessi punti fisici di interesse nella scena.



(a) Scena 2: Ripetibilità assoluta.



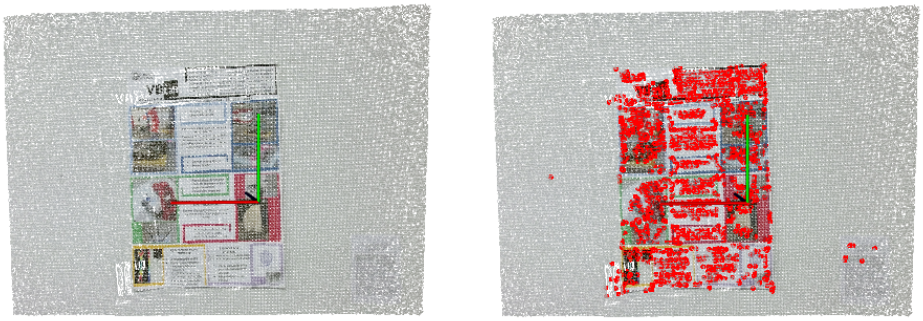
(b) Scena 2: Ripetibilità relativa.

Figura 4.12: Misura della ripetibilità assoluta, in (a), e relativa, in (b), per la seconda scena.

Dalle prove effettuate la ripetibilità e il tempo computazionale rendono l'uso dei Keypoints 2D la scelta ottimale per lo scopo di questa tesi. Per completare la trattazione e la valutazione dei keypoint detectors introduciamo due casi particolari che si sono riscontrati durante il cammino percorso in questa tesi.

Analisi di un piano colorato

Si prenda in considerazione la nuvola di punti in figura 4.13a che rappresenta un muro su cui è attaccato un cartellone, o più generalmente un piano con delle porzioni con colore diverso dallo sfondo. Per le nuvole appartenenti a questa tipologia non risulta possibile ricavare dei keypoints interessanti con l'utilizzo di detector 3D poiché, come già discusso nel capitolo precedente, essi si basano sulla forma tridimensionale della nuvola di punti in ingresso. L'unico metodo possibile è l'utilizzo dei keypoint detectors 2D, i quali, invece, agiscono sul gradiente dell'intensità luminosa riuscendo a cogliere keypoints descrittivi della nuvola. L'unico keypoint detectors 3D studiato che tiene anche conto della differenza di luminosità è Harris 6D, il quale però risulta dalle prove poco ripetibile. Per questo motivo nel software implementato si è inserita la possibilità di utilizzare anche algoritmi di keypoint detector 2D. Le immagini dei keypoints estratti dalla nuvola sono visibili in figura 4.13.



(a) Nessun keypoints estratto con il SIFT 3D.

(b) Keypoints estratti con BRISK.

Figura 4.13: Nuvola piana colorata e, in rosso, i keypoints estratti da essa tramite i diversi metodi descritti.

Analisi di scene a bassa luminosità

Si prenda in considerazione una nuvola di punti poco illuminata, come per esempio quella in figura 4.14. I detector 2D non riescono a ricavare alcun keypoint poiché nell'immagine nera, derivante dalla proiezione della nuvola, non sono presenti differenze rilevanti nella luminosità. In queste condizioni, l'unica opzione consiste nell'utilizzo di keypoint detector 3D. Nel paragrafo 5.5.4 attraverso il software implementato viene ricostruita una scena buia poichè esso è in grado di rilevare e descrivere keypoints 3D.

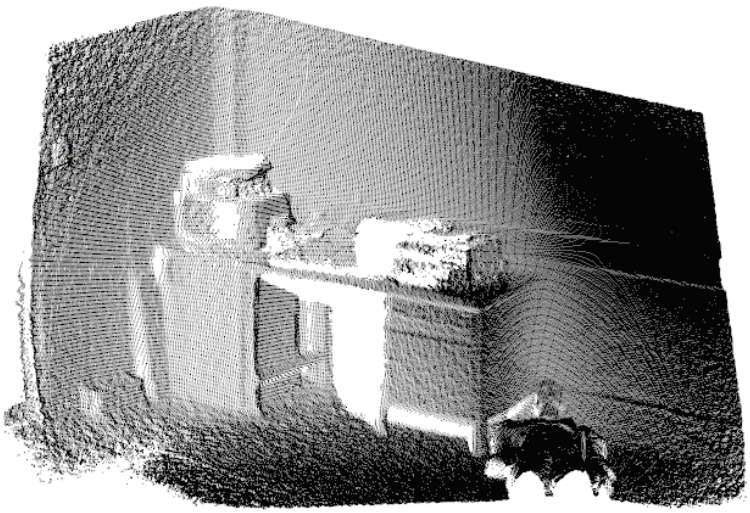


Figura 4.14: Nuvola di punti a bassa luminosità.

4.3 Keypoint Descriptors

Dopo aver estrapolato keypoints ripetibili, per un corretto allineamento tra nuvole di punti, è necessario descriverli, ottenendo delle features. Tali features devono poi essere accoppiate tra loro in modo da ricavare una trasformazione ottima tramite la minimizzazione delle distanze tra le corrispondenze. La descrizione dei keypoints risulta fondamentale per avere una registrazione consistente di due, o più, nuvole di punti.

Le caratteristiche principali di un keypoint descriptor sono la robustezza del feature ricavato e la complessità computazionale necessaria per descriverlo. Una feature è robusta se risulta poco sensibile ai disturbi come la presenza di rumore piuttosto che una variazione della risoluzione della nuvola. In letteratura [56], è stato effettuato uno studio dettagliato sulla robustezza delle features derivate da diversi keypoint descriptors 3D. Per questo motivo le prove sperimentali effettuate, descritte in questa sezione, si concentreranno principalmente sul confronto tra i tempi computazionale di algoritmi utilizzati per la descrizione di punti.

Nel capitolo 3 si sono introdotti, per completezza, keypoint descriptor 2D e 3D più utilizzati che hanno segnato il percorso evolutivo della computer vision. Si ricorda, però, che per lo scopo di questa tesi verranno presi in considerazione unicamente i keypoints descriptor 3D.

In questa sezione presentiamo le prove effettuate per avere un'idea sul carico, e quindi sul tempo, computazionale necessario per la descrizione di keypoints.

4.3.1 Descrizione software

In questo paragrafo, descriviamo il software dedicato alla valutazione dei keypoint descriptor 3D. Nell'interfaccia, in figura 4.15, sono distinguibili due diverse sezioni. La prima sezione è dedicata alla scelta del descrittore di keypoint, mentre la seconda è dedicata alla scelta dei parametri utilizzati nella descrizione. A differenza dei keypoint detectors, i parametri da impostare nei diversi descriptors sono gli stessi. Per questo motivo il confronto tra di essi risulta più diretto.

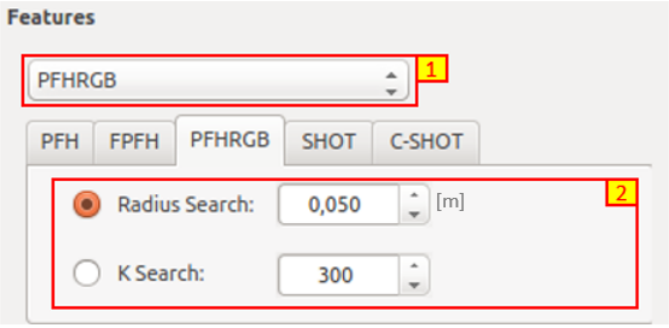


Figura 4.15: Interfaccia utente per la gestione dei descrittori di keypoints.

Ricordiamo che la feature del punto raccoglie non solo le informazioni del punto stesso, ma anche di quelli vicini appartenenti all'intorno. Risulta necessario quindi descrivere i due metodi utilizzati per identificare l'intorno del punto descritto.

Il primo metodo è chiamato *Radius Search*: sia m il keypoint considerato, p un punto generico della nuvola, la cui distanza con m è $d(m, p)$, e ϵ una distanza soglia, impostabile dall'utente. Il punto p appartiene all'intorno di m se si ha $d(m, p) < \epsilon$. Il secondo è chiamato *K-Search*: l'intorno del keypoint è definito dai k punti più vicini ad esso. Il parametro k è impostato dall'utente.

Non sempre, però, è possibile utilizzare entrambi i metodi. Si ricorda infatti che SHOT e C-SHOT descrivono il punto tramite la suddivisione di una sfera attorno ad esso. L'unico metodo utilizzabile per questi due descrittori 3D è dunque il primo. Questo influenza notevolmente la struttura delle nostre prove. Vengono eseguite due prove, nella prima si utilizzerà il metodo K-Search e nella seconda il metodo Radius Search.

4.3.2 Descrizione e risultati della prima sperimentazione

Lo scopo di questa sperimentazione è quello di analizzare il carico computazionale di vari descrittori, in diverse scene e di keypoint differenti, in funzione del numero di punti compresi nell'intorno di ciascun keypoints. Il carico computazione viene

verificato misurando il tempo di elaborazione sull'hardware standard utilizzato per la tesi.

Dataset

Il dataset è composto da quattro nuvole di keypoints derivanti da altrettante acquisizioni con il Kinect V2 e ricavate con metodi di keypoint detection differenti. In particolari sono state riprese sia le acquisizioni utilizzate per il confronto tra ICP che quelle utilizzate per lo studio effettuato sui keypoint detectors e, da queste, ricavate le nuvole di keypoints.

Sono qui riportate le acquisizioni in figura 4.16, la tabella 4.5 con il numero di keypoints estratto e con l'algoritmo di detection utilizzato.

(a) Scena A.

(b) Scena B.

(c) Scena C.

(d) Scena D.

Figura 4.16: Acquisizioni utilizzate per la prima prova.

Si è scelto di utilizzare diverse acquisizioni e diversi detectors per verificare che il tempo computazionale non dipenda ne dalla scena, ne dalla tipologia di algoritmo utilizzato per l'estrazione dei punti. É prevedibile infatti che il tempo per la descrizione di keypoints dipenda unicamente dal numero di essi e dal numero di punti nei loro intorni di descrizione.

	Keypoint Detector	Numero Keypoint
Scena A	SIFT 3D	2161
Scena B	Harris 3D	1920
Scena C	ISS 3D	1158
Scena D	BRISK	1672

Tabella 4.5: Setting per la prima prova sui descrittori.

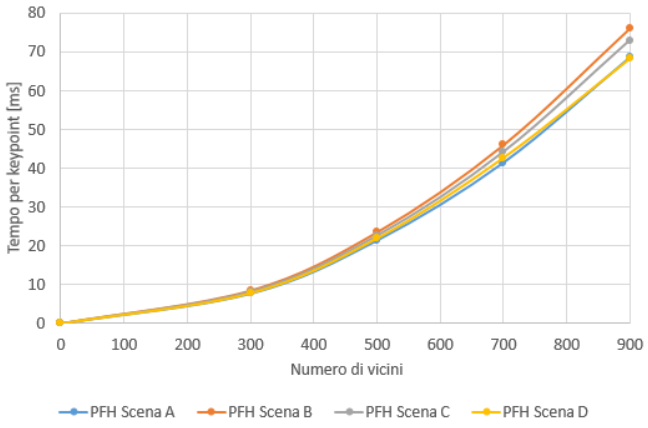
Noto il numero di keypoints descritti n, sono state eseguite delle prove per ogni scena, facendo variare il numero di punti appartenente all'intorno di descrizione k. Da queste, si è misurato il tempo totale T e calcolato il tempo medio di descrizione per punto t tramite la formula $t = T/n$. Per la struttura di questa prova è necessario vincolare il numero di punti appartenenti all'intorno di un keypoint; si utilizza dunque il metodo di identificazione dell'intorno K-Search. Lo studio quindi è limitato agli algoritmi PFH, FPFH e PFHRGB.

Sono stati ricavati tre grafici cartesiani, uno per ogni descrittore, aventi i valori t nelle ordinate, e i rispettivi valori k nelle ascisse. In essi vengono rappresentate le curve calcolate interpolando i tempi t ricavati dalle prove.

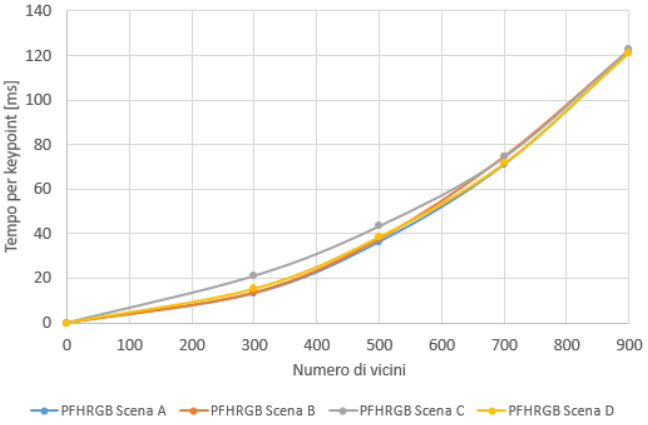
Risultati

I grafici elaborati sono visibili nelle figure 4.17 e 4.18a. Analizzandoli è possibile notare come le curve ricavate tramite l'utilizzo dello stesso algoritmo siano sovrapponibili.

Dalla somiglianza tra le curve ottenute con i medesimi algoritmi di descrizione dei punti e scene differenti è possibile dedurre che il tempo computazionale non dipende ne dalla scena, ne dalla tipologia di algoritmo utilizzato per l'estrazione dei punti.



(a)

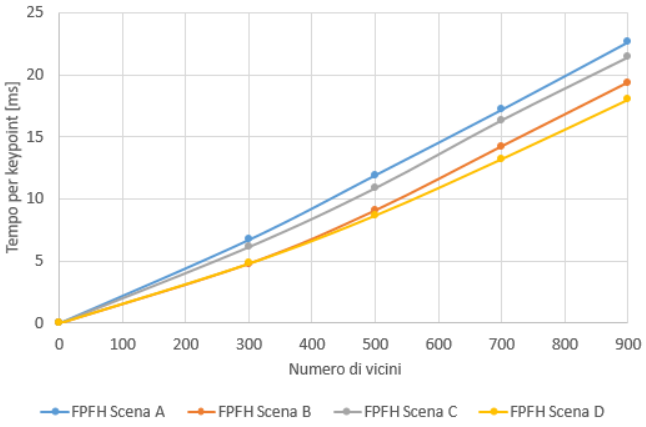


(b)

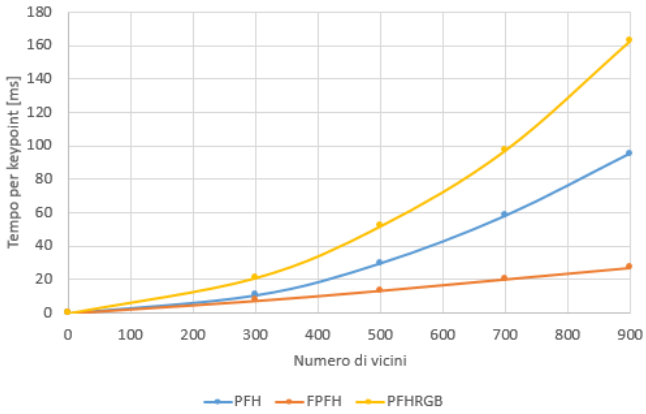
Figura 4.17: Tempo medio per il calcolo delle features tramite gli algoritmi PFH (a) e FPFH (b) applicati alle diverse scene, in funzione del numero di punti utilizzati per descrivere i keypoints.

Una volta verificato che il tempo computazionale dei vari algoritmi dipende sostanzialmente solo dal numero di keypoints descritti e dal numero di punti nei loro intorni di descrizione, si è valutato la complessità computazionale di questi descrittori. Per la comparazione dei tempi impiegati dai tre diversi descrittori, è stato ricavato un ulteriore grafico, rappresentato in 4.18b in cui tre diverse curve rappresentano gli andamenti medi delle curve nei tre grafici di figura 4.17a, 4.17b e 4.18a.

A questo punto sono state effettuate delle considerazioni sulla complessità computazionale dei vari algoritmi considerando l'andamento delle curve di que-



(a)



(b)

Figura 4.18: Nella figura (a) ha è rappresentato il tempo medio per il calcolo delle features tramite gli algoritmi PFH applicato alle diverse scene, in funzione del numero di punti utilizzati per descrivere i keypoints, in (b), invece, sono presenti gli andamenti medi.

st'ultimo grafico. É possibile notare, infatti, come l'andamento delle curve PFH e PFHRGB sia esponenziale, mentre quello relativo a FPFH sia lineare. Questi risultati sono coerenti alla complessità di calcolo che ci si aspetta analizzando la struttura dei metodi descrittivi usati.

Gli algoritmi PFH e PFHRGB infatti confrontano tutti i punti nell'intorno del keypoint, calcolando una rete di relazioni che ha una complessità computazionale di $O(k^2)$. Questo significa che, per una nuvola di n punti, la complessità è di $O(nk^2)$. Notiamo inoltre che la curva PFHRGB ha un guadagno proporzionale maggiore di uno rispetto a PFH. Questo è dovuto dal maggior tempo impiegato per confrontare ogni singolo legame della rete di relazioni, in quanto vengono

confrontati anche il colore dei punti.

L'algoritmo FPFH invece confronta solamente il keypoint e i suoi k vicini, creando una rete di relazioni che avrà una complessità computazionale di $O(k)$. Dunque, per una nuvola di n punti, si avrà una complessità computazionale di $O(nk)$, in linea con l'andamento di FPFH nel grafico 4.18b. La minor complessità computazionale rappresenta il vantaggio di FPFH rispetto agli altri algoritmi.

4.3.3 Descrizione e risultati della seconda sperimentazione

Per confrontare tutti i keypoint descriptors implementati nel software ed avere quindi un'indicazione sui tempi richiesti, sono state fatte delle prove definendo l'intorno dei punti tramite il metodo *Radius Search*. Come già detto, SHOT e C-SHOT, per struttura, necessitano di questo metodo.

Il dataset preso in considerazione è quello utilizzato nella prima prova. L'unica differenza sta nel fatto che sono state eseguite due acquisizioni nella stessa posa per ogni scena.

In una registrazione basata su features, la scelta del descrittore non influisce solamente sul tempo effettivo di descrizione. Tale scelta determina il tipo di features, e quindi la sua relativa grandezza, che verranno confrontate per ricavare delle corrispondenze. Il calcolo computazionale impiegato per confrontare due features dipende direttamente dal numero di bins presenti negli istogrammi delle stesse. Se l'istogramma di un punto contiene molti bins, il calcolo computazionale per trovare il punto più simile nell'altra nuvola sarà più dispendioso. Il tempo per la ricerca delle corrispondenze è dunque attribuibile al tipo di features e proprio del keypoint descriptor scelto.

Innanzitutto sono stati calcolati, per ogni algoritmo, i tempi di descrizioni dei keypoints estratti dalla prima acquisizione di ogni scena, variando il raggio che identifica l'intorno e, da questi, si è ricavato l'andamento dei tempi di descrizione.

In seguito, per ogni scena e utilizzando i vari algoritmi, si sono ricavate le descrizioni di entrambe le nuvole di keypoints estratti dalle due diverse acquisizioni, sempre in funzione del raggio utilizzato. Vengono poi ricercate le corrispondenze e misurato il tempo impiegato in quest'ultima fase.

Il tempo totale aggiunto alla registrazione basata su features dalla scelta di utilizzo di un keypoint descriptor, verrà calcolata, raddoppiando il tempo di descrizione, poiché per effettuare la registrazione vengono calcolate le features per due nuvole, e sommando ad esso il tempo per la ricerca di corrispondenze.

Risultati

Sono state eseguite le prove su tutte le scene a disposizione e sono stati analizzati i grafici relativi. Nell'intento di essere sintetici si riportano solamente i grafici della scena "A" in quanto rappresentativi di tutte le prove svolte. Anche le considerazioni che verranno sviluppate valgono indistintamente per ogni grafico.

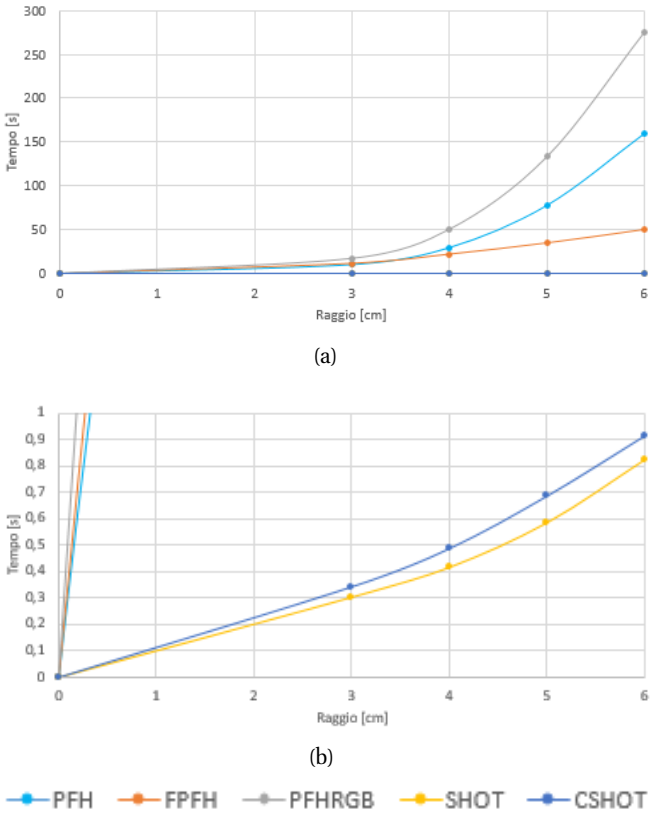


Figura 4.19: In (a) gli andamenti dei tempi di descrizione in funzione del raggio dell'intorno utilizzato per la descrizione stessa e in (b) un relativo dettaglio.

In figura 4.19a sono illustrati gli andamenti dei tempi di descrizione di ogni algoritmo, in funzione del raggio che identifica l'intorno del punto descritto. Si nota come le curve relativa a PFH, FPFH e PFHRGB siano paragonabili ai risultati delle prove precedenti. Il dato che salta all'occhio in questo grafico, è il tempo impiegato per la descrizione tramite SHOT e C-SHOT. Per poter apprezzare l'andamento del tempo di questi due algoritmi è necessario un cambio di scala. In figura 4.19b è presente un dettaglio degli andamenti dei tempi di SHOT e C-SHOT. Entrambi gli algoritmi impiegano, per descrivere la nuvola di keypoints, un tempo nell'ordine dei decimi di secondo, nettamente inferiore paragonato agli altri algoritmi. Questa velocità di esecuzione è il punto di forza dei descrittori SHOT e C-SHOT.

Dopo aver analizzato il tempo di descrizione dei vari algoritmi, si sposti l'attenzione sul tempo di ricerca delle corrispondenze. Intuendo come questo tempo sia derivato soprattutto dal numero dei bins da cui è composto l'istogramma, viene riportata la tabella riassuntiva 4.6.

	PFH	**PFHRGB**	**FPFH**	**SHOT**	**C-SHOT**
Numero di Bins	125	250	33	352	1344

Tabella 4.6: Numero di bins che definiscono le diverse tipologie di features.

Dalla figura 4.20 è possibile vedere come il tempo impiegato per la ricerca di corrispondenze degli algoritmi PFH, FPFH e PFHRGB sia trascurabile. Con i descrittori SHOT e C-SHOT, invece, viene impiegato più tempo nel calcolo delle corrispondenze che nel descrivere i keypoints.

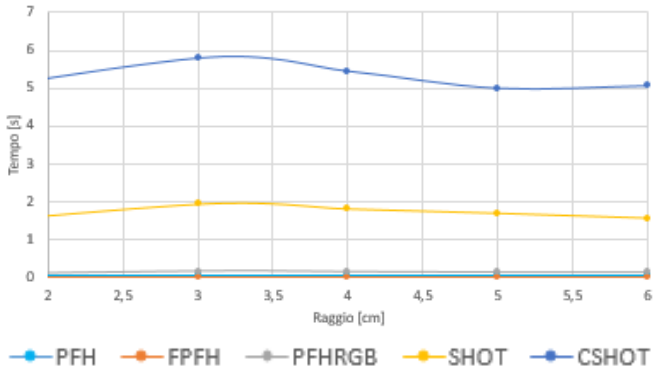


Figura 4.20: Andamento dei tempi per il calcolo delle corrispondenze in funzione del raggio dell'intorno.

Infine si è ricavato un grafico degli andamenti del tempo totale aggiunto alla registrazione basata su features, in base alla scelta di utilizzo di un keypoint descriptor, visibile in figura 4.21. Questo tempo, come già accennato, è stato calcolato raddoppiando il tempo di descrizione e sommando il tempo per la ricerca delle corrispondenze tra feature. Dal grafico è chiaro come gli algoritmi SHOT e C-SHOT descrivono e ricavano un set di corrispondenze in un tempo minore rispetto agli altri algoritmi.

Analizzando quest'ultimo grafico inoltre può essere effettuata un'ulteriore considerazione. Nei primi tre algoritmi, (PFH, PFHRGB e FPFH) una piccola variazione del raggio di intorno e quindi del numero dei vicini utilizzati per la descrizione, comporta un aumento consistente del tempo computazionale. Il tempo di questi algoritmi quindi è molto sensibile alla variazione dei parametri. Questo per un utente non esperto può risultare un problema. Lo stesso comportamento non è

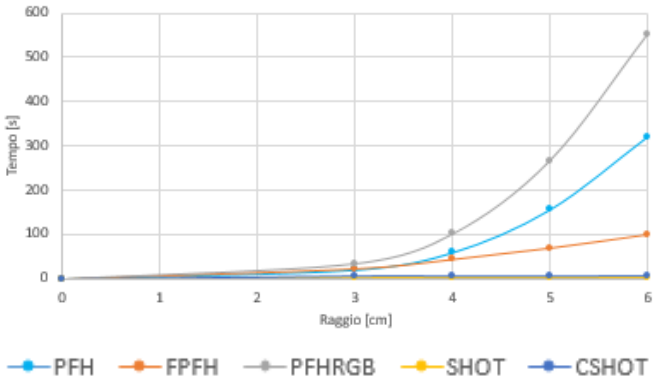


Figura 4.21: Andamento dei tempi totali stimati

riscontrabile negli algoritmi SHOT e C-SHOT, i cui tempi risultano poco sensibili ad un cambio di parametri.

É necessaria inoltre un'ultima riflessione sull'utilizzo dell'informazione del colore. Se durante la registrazione viene scelto un metodo di rilevazione dei keypoints 2D (BRISK o AGAST), il punto di interesse trovato sarà caratteristico in base alla variazione dell'intensità luminosa intorno ad esso. Utilizzare in questo caso un descrittore che considera solamente l'informazione di forma risulta controproducente. Questa affermazione è supportata da prove effettuate sulla scena, chiamata anche piano colorato, in figura 4.16d. Si è verificato infatti come le corrispondenze, tra nuvole di keypoint rilevati con algoritmi 2D e descritti con descrittori di forma, risultino errate. In questo caso la scelta è limitata agli algoritmi PFHRGB e C-SHOT.

4.4 Conclusioni

In questo capitolo sono stati analizzati e confrontati gli algoritmi principali utilizzati per la registrazione di nuvole di punti. I risultati di questa sperimentazione permettono di definire le migliori pipeline di algoritmi da utilizzare durante la registrazione di scene. Le linee guida ricavate sono quelle sotto riportate.

In presenza di un ambiente colorato si prediligono keypoint detectors 2D poiché risultano essere computazionalmente molto più rapidi e con una maggior ripetitività, non essendo affetti da riflessioni multiple, rispetto a quelli 3D. In particolare si è scelto di utilizzare l'algoritmo BRISK, poiché risulta essere, a differenza dell'AGAST, invariante alla scala. Quando però la scena non risulta essere sufficientemente illuminata, o con poche variazioni di colore, è necessario utilizzare un keypoint detector 3D. Si predilige l'algoritmo SIFT 3D perché risulta essere quello più ripetibile.

Per quanto riguarda i descrittori, SHOT e C-SHOT risultano essere algoritmi molto più rapidi degli altri keypoint descriptors 3D. Dopo diverse prove qualitative di registrazione si sono rivelati anche robusti. Per questo motivo verrà utilizzato l'algoritmo C-SHOT se i keypoints sono stati estratti tenendo in considerazione la variazione di colore, l'algoritmo SHOT viceversa.

Se dopo questa fase di registrazione basata su features, le nuvole risulteranno esser poco allineate, cioè sarà visibile un errore di allineamento, verrà effettuata una fase di rifinitura dell'allineamento utilizzando un ICP con il metodo *TransformationEstimationPointToPlainLSS*, poiché dalle prove risulta essere estremamente più veloce e ugualmente accurato rispetto alle altre tipologie di ICP.

Capitolo 5

Test Applicativi

Il capitolo descrive i vantaggi ed i limiti dei software per la ricostruzione di scene, mettendo in evidenza i motivi che hanno portato alla implementazione di una nuova soluzione. La soluzione sviluppata nel lavoro di tesi si basa su due moduli. Il primo gestisce l'acquisizione tramite Kinect V2 ed il secondo la ricostruzione delle scene. Da ultimo vengono poi presentati i due casi applicativi in cui i software e il sensore Kinect V2 sono messi a confronto con sistemi già utilizzati in campo professionale: la ricostruzione di un sito archeologico e di una scena del crimine.

5.1 Programmi esistenti

Prima di passare alla descrizione dei programmi implementati si presentano i vantaggi e i limiti dei software per la ricostruzione di scene presenti in letteratura, mettendo in risalto i motivi per cui è stato necessario l'implementazione di moduli specifici per le finalità di questa tesi. Saranno trattati solo i software di ricostruzione che utilizzano come strumento di acquisizione preferibilmente il Kinect V2.

5.1.1 Kinect Fusion

Un software robusto che Windows mette a disposizione per la ricostruzioni di scene indoor è Kinect Fusion, implementato in una versione open source in KinFu [57]. In questo programma le nuvole acquisite dal sensore vengono fuse in un unico modello, che rappresenta la scena globale, formato non da nuvole di punti ma da superfici. L'output quindi non è una nuvola di punti, bensì una mesh. Le mesh 3D sono delle griglie 3D composte da una serie di vertici, edge, e triangoli/poligoni.

Questo programma permette di ricostruire in tempo reale una scena relativamente piccola. La ricostruzione, infatti, è limitata ad un cubo di tre metri per lato per motivi di saturazione della memoria dell'elaboratore. In questo caso non è possibile modificare il codice e la limitazione sul campo di acquisizione non permette una facile ricostruzione di un intera scena a 360^o. Tale ricostruzione viene infatti eseguita registrando i cubi acquisiti dapprima a mano e poi rifinendo l'allineamento tramite un ICP.

Il KinFu Large Scale è un software che funziona esattamente come il precedente, ma non è limitato nel volume di scansione. In pratica crea ed allinea una successione di cubi $3 \times 3 \times 3$ m. Richiede però una GPU ad elevate prestazioni per l'esecuzione della registrazione delle nuvole.

5.1.2 RTAB-Map

Il programma che risulta essere più promettente nella ricostruzione di scene è RTAB-Map (Real-Time Appearance-Based Mapping). Questo software la cui implementazione è durata circa 4 anni (e che tuttora continua), è il vincitore del primo premio all' *IROS 2014 Microsoft Kinect Challenge*, per la sua capacità di ricostruire ambienti indoor in applicazioni real time.

Con questo programma è possibile utilizzare i più diffusi sensori RGB-D sul mercato, tra cui il Kinect e il Kinect V2.

La pipeline che utilizza per la registrazione di nuvole di punti si basa su keypoint detectors e descriptors 2D per trovare corrispondenze tra immagini. Tramite la matrice di proiezione della telecamera, descritta nella sezione 1.3.1, è possibile ricavare da un qualsiasi pixel dell'immagine il rispettivo punto 3D. Per ogni corrispondenza trovata tra due punti dell'immagine, vengono ricavati tramite la matrice di proiezione i rispettivi punti in 3D e tra di essi viene impostata una corrispondenza.

La ricerca delle corrispondenze di punti su immagini risulta essere molto più rapida, per questo il software riesce ad allineare le nuvole in tempo reale.

Un problema che sussiste nella elaborazioni di nuvole in larga scala è che il tempo di processo richiesto per la nuova osservazione dipende dalla grandezza della mappa interna ricostruita, questo può influenzare la natura real time del processo. Il software utilizza una complessa gestione della memoria, descritta in [58] e in [59] , che permette di mantenere il tempo computazionale sotto un certo limite fissato e di determinare i loop closure.

RTAB-Map risulta molto efficace per i problemi di tipo SLAM, ma utilizza solamente algoritmi 2D per l'estrazione e la descrizione di keypoints. Il programma implementato invece ricostruisce la scena utilizzando algoritmi che si basano su informazioni 3D, come distanza euclidea, normali e curvatura. In particolare si

ha la possibilità di utilizzare keypoint detector 2D o 3D e unicamente descriptor 3D. In figura 5.1 è possibile vedere le pipeline di processo utilizzate. Le linee verdi identificano il momento nel processo in cui avviene la proiezione necessaria per passare da pixel a punti 3D. É chiaro come rilevare i keypoints in 3D e descriverli in 2D non risulta avere molto senso.

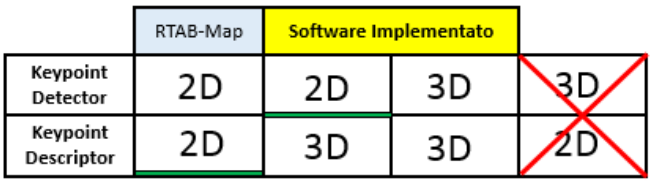


Figura 5.1: Schema di funzionamento per i programmi che utilizzano registrazioni features based.

Il carico computazionale del software implementato sarà, per l'utilizzo di algoritmi 3D, necessariamente più elevato e quindi l'intero processo più lento. D'altra parte però, utilizzare le informazioni di forma può migliorare l'accuratezza della ricostruzione di una scena come sopra spiegato.

5.2 Moduli software sviluppati

Il primo modulo permette l'acquisizione dei dati tramite Kinect V2, mentre il secondo si occupa della gestione delle nuvole di punti ed in particolare del loro allineamento tramite tecniche SLAM, con lo scopo di ricostruire un'ampia scena.

Si è scelto di non implementare entrambe le funzionalità in un software unico in modo da disaccoppiare l'acquisizione dalla ricostruzione. Questa soluzione risulta interessante in quanto il modulo di ricostruzione è utilizzabile non solamente con le nuvole di punti acquisite con il Kinect V2, ma con tutte quelle che hanno una struttura organizzata e risultano proiettabili, semplicemente cambiando i parametri della matrice di proiezione.

5.2.1 Modulo di acquisizione

Esistono già programmi open source disponibili per l'acquisizione con il sensore Kinect V2. L'implementazione di uno specifico modulo è motivata dalla necessità di introdurre caratteristiche specifiche per le due applicazioni di ricostruzione sviluppate durante la tesi: minimizzare i tempi di acquisizione e possibilità di avere un riscontro visivo della regione di scena in acquisizione.

Nell'ambito della ricostruzione di una scena del crimine elementi fondamentali sono infatti il tempo impiegato per l'acquisizione delle nuvole e la semplicità di

utilizzo del sistema. É necessario infatti che questo tipo di operazione sia fatta nel minor tempo possibile, in modo da non intralciare gli altri operatori coinvolti nel rilievo. Nello specifico, la maggior parte dei software disponibili non ha la possibilità di acquisire in modo continuo le nuvole di punti rilevate dal Kinect, ma viene eseguita una singola acquisizione cliccando con il mouse un particolare pulsante. In questo modo, l'acquisizione di un dataset di nuvole che comprende l'intera stanza risulta dispendioso in termini di tempo e oltre che di non semplice esecuzione.

Il secondo requisito necessario per un buon programma di acquisizione è la possibilità di avere un riscontro sull'area di scena che si sta acquisendo in quel momento. Questo offre la possibilità di verificare immediatamente che tutta la scena che si intende acquisire venga effettivamente e correttamente scandita.

Il software di acquisizione, dunque, è stato sviluppato con lo scopo di minimizzare i tempi dovuti all'acquisizione, potendo avere un riscontro visivo della regione di scena che si sta acquisendo.

Come già discusso il software è sviluppato per un sistema operativo Ubuntu. Mentre in Windows esistono programmi che restituiscono una nuvola di punti colorata, in Ubuntu è unicamente possibile ricavare dal Kinect V2 singolarmente le immagini a colori, le immagini IR e le immagini di profondità. Per ricavare queste informazioni sono state utilizzate delle librerie open source chiamate "*libfreenect2*".

Per lo sviluppo del software è stato necessario convertire l'immagine di profondità in nuvola di punti e proiettare su di essa il colore acquisito dalla telecamera RGB, ottenendo così una nuvola di punti colorata. La proiezione è stata effettuata basandosi sulla matrice di proiezione del modello di camera pinhole definita nel paragrafo 1.3.1.

Dopo aver giustificato la necessità di implementare un nuovo software, si passa alla descrizione dell'interfaccia e delle funzionalità.

L'interfaccia utente è strutturata in due finestre. La prima consiste in un visualizzatore di nuvole di punti disponibile nelle librerie PCL che ha lo scopo di permettere all'utente di rendersi conto di cosa sta acquisendo. Nella parte bassa del visualizzatore, in modo da non interferire con la visualizzazione della scena, sono stati inseriti il numero di nuvole acquisite nella sessione corrente ed il numero di frame al secondo a cui sta funzionando il sistema. Questi due informazioni sono utili all'utente in quanto è possibile stimare la memoria utilizzata per la sessione, sapendo che ogni nuvola di punti acquisita con il Kinect V2 e in formato Point Cloud Data (PCD), utilizzato dalle librerie PCL, pesa circa 7 MB.

La seconda finestra, visibile in figura 5.2 consiste in una serie di pulsanti utilizzati per gestire il visualizzatore ed il salvataggio delle nuvole di punti.

Nella schermata sono identificabili tre aree. La prima area è composta da quattro pulsanti. Inizialmente il visualizzatore è spento. Con il tasto "play" il software

Figura 5.2: Schermata principale del software di acquisizione.

cerca la presenza di un Kinect V2 collegato al computer, ricava i parametri della matrice di proiezione e incomincia una visualizzazione continuamente aggiornata delle nuvole di punti acquisite. A questo punto premendo il pulsante "Rec", il programma crea, nel percorso identificato dalla seconda area, una sottocartella, il cui nome è composto dalla data e ora corrente, e salva in essa le nuvole che il sensore acquisisce, alla velocità impostata. Il tasto "pause" permette di fermare unicamente il salvataggio mentre il tasto "stop" ferma anche la visualizzazione. Cliccando il tasto "rec" dopo il tasto "stop" si crea una nuova sottocartella, iniziando così una nuova sessione di acquisizione. Pur avendo una interfaccia semplice, l'implementazione di questo codice ha richiesto un importante lavoro.

Innanzitutto è necessario rilevare il Kinect V2, creare due canali listener che ricevono i dati di colore e profondità e ricavare i parametri di proiezione che identificano la telecamera, immagazzinati nella telecamera stessa. Una volta recuperati i parametri, tramite le formule descritte nel paragrafo 1.3.1, vengono proiettate le informazioni di colore sull'informazione profondità ricavando così una nuvola di punti colorata.

Il problema che ha richiesto un maggiore studio però non è risultato l'estrazione di nuvole di punti dal sensore. Analizzando la struttura di un visualizzatore che si aggiorna continuamente all'arrivo di una nuova nuvola, è possibile intuire come questo sia necessariamente una funzione iterativa che aspetta l'acquisizione, la riceve, la salva e di conseguenza si aggiorna. Una volta entrato in questo loop è necessario controllare, tramite i comandi descritti, l'uscita da questa funzione. Questi, però, non risultano essere più selezionabile poiché il processo è impegnato.

Il problema è stato risolto utilizzando quello che in informatica è definito come *multithreading*. Un thread è una sequenza di istruzioni di un programma in corso di esecuzione. Nelle architetture a processore singolo, quando la CPU esegue alternativamente istruzioni di thread differenti, si parla di multithreading a divisione di tempo: la commutazione fra i thread avviene di solito tanto frequentemente da dare all'utente l'impressione che tutti i task siano eseguiti contemporaneamente. Nelle architetture multi-processore i thread vengono invece realmente eseguiti

contemporaneamente cioè in parallelo, ciascuno su un distinto core. Si è spostata tutta la parte di codice che riguarda la visualizzazione e il salvataggio di nuvole in un thread appositamente creato con alta priorità. In questo modo le funzioni chiamate dai pulsanti di comando si trovano in un diverso sottoprocesso rispetto al visualizzatore, e risultano quindi selezionabili. Le funzioni "play", "stop" e "pause" cambiano delle variabili globali del software utilizzate da entrambi i sottoprocessi e grazie ad esse è possibile determinare la visualizzazione e il salvataggio delle nuvole.

5.2.2 Modulo per la ricostruzione di scene

Una volta acquisite le nuvole di punti è necessario un programma che consenta la ricostruzione della scena. Per questo scopo si è implementato un secondo modulo che permette l'elaborazione di nuvole, il calcolo delle normali, la registrazione (manuale, con ICP e quella basata su features) e l'utilizzo di un algoritmo che gestisca la fase back-end dello SLAM.

Finestra Principale

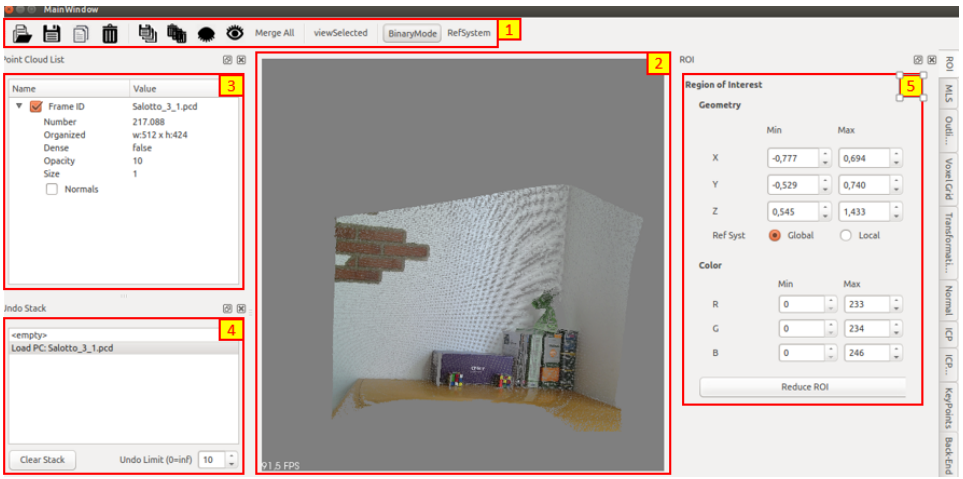


Figura 5.3: Schermata principale del software.

L' interfaccia principale del software, in figura 5.3, è composta da cinque parti:

1. Operazioni generali: apertura file, salvataggio e cancellazione sono di immediata comprensione. I file di tipo Point Cloud Data (.pcd), Polygon File Format (.ply) e i file di testo sono importabili nel programma. Le icone in cui sono rappresentati degli occhi possono rendere visibili o nascoste tutte le nuvole importate. Sono presenti inoltre i tasti: "Merge All", che unisce in un

unica nuvola tutte quelle importate, "View Selected", che colora in rosso nel visualizzatore la nuvola di punti selezionata, "Binary Mode", che permette il salvataggio del file in binario, e "RefSystem", che permette di visualizzare il sistema di riferimento.

2. Un visualizzatore tridimensionale messo a disposizione dalle librerie PCL, modificato ed integrato nell'applicativo.

3. Una lista delle nuvole importate in cui sono allocate le informazioni di ogni nuvola. Oltre al nome di ogni nuvola importata è presente: il numero di punti, il numero di righe e colonne della matrice in cui sono allocati i punti della nuvola (e quindi se essa risulta organizzata), l'informazione di densità della nuvola, cioè se tutti i punti appartenenti ad essa hanno valore x, y e z limitato, l'opacità nel visualizzatore, e la possibilità di permettere la visualizzazione delle normali se queste sono state calcolate. Le nuvole nella lista possono essere selezionate e trattate opportunamente con funzioni che permettono di estrarre ROI, di rimuovere outliers, di campionarle e di stimare le normali.

4. Il modulo dispone di una cronologia delle operazioni che permette di annullare un'operazione non desiderata. Conviene limitare il numero di stack salvati poiché un suo aumento porta al riempimento della memoria RAM del computer. Durante i test applicativi effettuati il limite di operazioni nella cronologia è stato impostato a 10.

5. Inoltre il software mette a disposizione dei pannelli a scomparsa in cui sono presenti tutti i parametri impostabili per l'utilizzo dei vari algoritmi che sono stati implementati in questo software.

Dopo aver presentato la schermata generale, per capire le potenzialità di questo programma è necessario descrivere le funzioni che esso è in grado si svolgere. Divideremo quindi la descrizione delle funzionalità in cinque macrocategorie: funzioni generali, registrazione manuale, registrazione con ICP, registrazione basata su features e BackEnd.

Funzioni Generali

Regione di Interesse: ROI

Tramite la funzione ROI, l'utente può eliminare alcuni punti della nuvola modificando, tramite le matrici in figura 5.4, i limiti massimi e minimi che un punto può avere in x, y, z o nei valori RGB. Qualunque punto che non si trova nei limiti impostati viene eliminato dalla nuvola. Inoltre ogni volta che si seleziona una

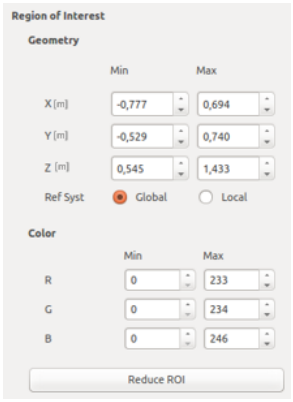


Figura 5.4: Finestra per la gestione delle ROI (Region Of Interest).

nuvola di punti vengono calcolati e riportati nelle due tabelle i limiti massimi e minimi riscontrabili nei punti della nuvola in x, y, z e RGB .

Rimozione Outliers

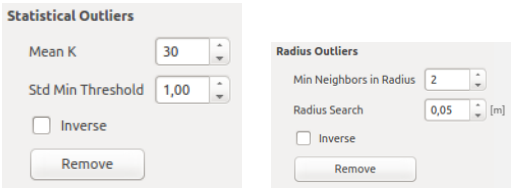
Gli outliers sporcano le caratteristiche locali dei punti della scansione, come le normali di superficie o i cambi di curvatura, e causano difficoltà nella registrazione. Anche se nelle acquisizione tramite Kinect V2, gli outliers sono poco presenti è stato scelto di implementare ugualmente due funzioni per la loro rimozione.

Il filtro Statisical Outliers Removal calcola la distanza media tra un punto e i suoi K vicini; se la media delle distanze calcolata risulta non appartenere ad un intervallo definito da una distanza globale media, propria della nuvola, ed una deviazione standard allora il punto è ritenuto un outliers. Tramite l'interfaccia sono impostabili il numero dei punti vicini K, utilizzato per calcolare la media delle distanze, e la deviazione standard soglia, espresso in metri.

Il filtro Radius Removal verifica che ci siano almeno K punti in un certo raggio r, espresso in metri; K e r sono entrambi parametri impostabili. Esempi dei risultati di questi filtri sono riportati in figura 3.5. Per entrambi gli algoritmi è possibile selezionare l'opzione "Inverse" che permette di eliminare gli inliers invece degli outliers. Le finestre che gestiscono la rimozione degli outliers sono visibili in figura 5.5.

Campionamento

Il campionamento è solitamente utilizzato per ridurre il tempo computazionale dell'ICP. Riducendo il numero di punti, si riduce il numero di corrispondenze possibili usate per registrare le nuvole di punti. Nel software è possibile utilizzare quattro algoritmi di questo tipo.



(a) Statistical Outliers Removal (b) Radius Removal

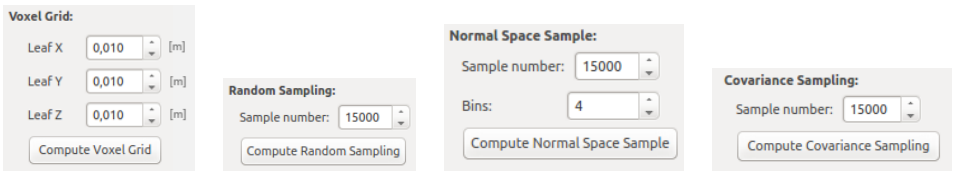
Figura 5.5: Finestre per il controllo dei parametri di due metodi per la rimozione di outliers.

Il campionamento *uniforme* è ricavato utilizzando la funzione *VoxelGrid*. Tramite i parametri che possono essere impostati, visibili in figura 5.6a, è possibile avere una diversa campionatura in x , in y ed in z.

Il secondo algoritmo utilizzabile è il campionamento *random*. L'unico parametro da impostare, in figura 5.6b, è il numero di punti che la nuvola avrà in uscita da esso.

Il campionamento *normal space* utilizza le normali per pesare in modo diverso le aree della nuvola di punti. Aree che hanno un maggior dettaglio contengono anche normali con direzioni differenti; perciò queste possono essere usate come guida per controllare il numero di campionamenti in una certa area. A differenza degli altri algoritmi, questo tende a mantenere più punti negli spazi in cui varia la direzione delle normali e quindi è presente un dettaglio. I parametri da impostare, in figura 5.6c, sono il numero di punti che la nuvola in uscita dovrà avere e un numero di bins utilizzato per il confronto di normali tra vicini; più il numero sarà elevato più l'algoritmo sarà sensibile a variazioni delle normali. Generalmente il numero di bins ha valore 4 o 16.

L'ultimo algoritmo di questo tipo è chiamato *covariance sampling*. In tale algoritmo l'unico valore da impostare, in figura 5.6d, è il numero di punti della nuvola in uscita dall'algoritmo.



(a) (b) (c) (d)

Figura 5.6: Finestre per il controllo dei parametri dei metodi per campionare una nuvola di punti: campionamento uniforme o Voxelgrid (a), campionamento Random (b), campionamento Normal Space (c), covariance Sampling (d).

Stima delle normali

Le normali svolgono una funzione importante nella registrazione di nuvole di punti. Queste infatti sono utilizzate sia dall'algoritmo più prestante dell'ICP (quello con il metodo PointToPlaneLLS) sia nella rilevazione e descrizione di punti tramite algoritmi 3D. In questo software si ha la possibilità di scegliere tra due diversi algoritmi in base alla struttura delle informazioni nella nuvola di punti considerata.

Se la nuvola non risulta organizzata è necessario utilizzare l'algoritmo che nelle PCL è chiamato *Normal Estimation* ($\rightarrow NormalEstimation$). La stima delle normali alla superficie in un determinato punto è effettuata comparando il punto stesso con i propri vicini. La determinazione dell'intorno che definisce i vicini del punto risulta allora fondamentale. Nel software sono implementati due metodi per la determinazione dell'intorno: il *radius search*, con il quale due punti risultano vicini se la loro distanza è minore di una distanza soglia r, o il *k-search*, in cui l'intorno è composto dai k vicini. Entrambi i metodi e i rispettivi parametri sono presenti nella figura 5.7.

Se la nuvola risulta organizzata, invece, è più conveniente in termini di tempo utilizzare l'algoritmo chiamato *Integral Image* ($\rightarrow IntegralImageNormal Estimation$) che calcola le normali con i vicini nel dominio bidimensionale organizzato. Per questo algoritmo è impostabile il massimo cambiamento di profondità per il calcolo delle normali nei punti di bordo degli oggetti.

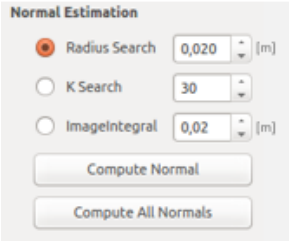


Figura 5.7: Finestra per il calcolo delle normali.

Registrazione manuale

I metodi di registrazione manuale sono scomodi ma ancora utilizzati da molti utenti per allineare due o più nuvole di punti. Per questo anche nel modulo sviluppato sono previsti due diversi metodi per la registrazione manuale. Prima di passare alla loro descrizione è necessario aggiungere che, per poterli utilizzare al meglio, sono stati definiti due sistemi riferimento per ogni nuvola, uno globale e uno locale. Il sistema di riferimento globale è comune a tutte le nuvole, mentre il sistema di riferimento locale è inizializzato in modo da combaciare con il sistema

di riferimento del sensore nel momento della acquisizione ed è quindi proprio di ogni nuvola.

Rotazioni e traslazioni attorno agli assi

Il primo metodo consiste in un susseguirsi di traslazioni e rotazioni attorno al sistema di riferimento globale o quello locale di una distanza, in metri, o di un angolo, in gradi, entrambi impostabili.

Per questo metodo si è utilizzato un'interfaccia, visibile in figura 5.8a, che può essere divisa in tre aree.

1. Nella prima sono presenti principalmente una serie di pulsanti. Con i pulsanti di sinistra, identificati con la lettera R, è possibile ruotare il sistema di riferimento locale della nuvola selezionata attorno gli assi del sistema di riferimento globale o locale, a seconda delle spunte nell'interfaccia. Con quelli di destra, identificati con una T, è possibile invece traslare il sistema di riferimento locale della nuvola selezionata lungo gli assi del sistema di riferimento globale o locale, sempre in funzione delle spunte nell'interfaccia. É possibile inoltre impostare la distanza per la traslazione, e l'angolo per la rotazione.

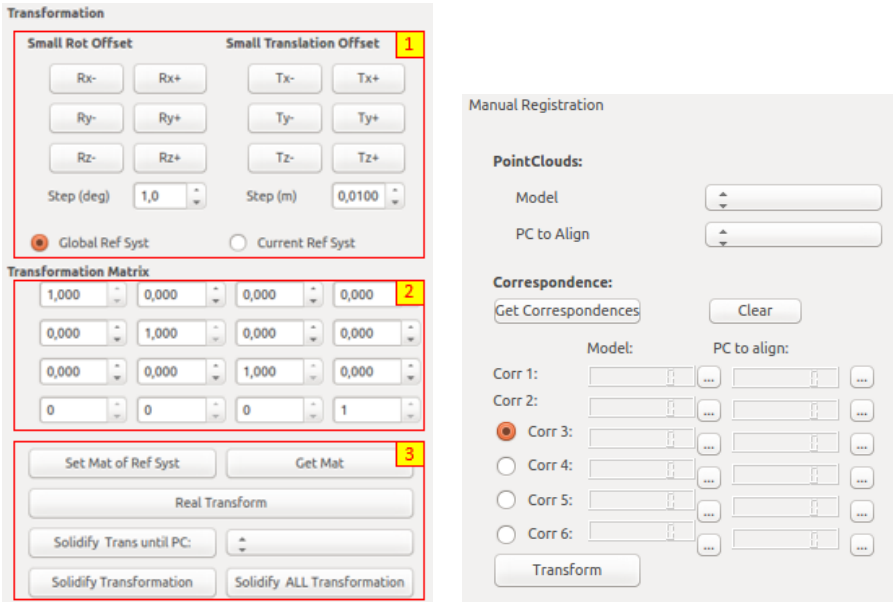
2. Nella seconda parte è presente una matrice 4×4 di valori che determina la matrice di rototraslazione tra il sistema di riferimento globale e quello locale.

3. Nella terza sono presenti delle funzioni per gestire i due sistemi di riferimento. In particolare le funzioni "Solidify" servono per ridefinire la posizione di tutti i punti della nuvola selezionata, o di una serie di nuvole, secondo il sistema di riferimento globale.

Trasformazione tramite corrispondenze

Il secondo metodo consiste nell'impostare a mano tre o più corrispondenze tra le due nuvole che si vuole registrare e stimare la trasformazione che le minimizza. I punti possono essere scelti tenendo premuto il tasto sulla tastiera "Ctrl" e cliccando nel punto opportuno appartenente all'immagine visualizzata.

Nell'interfaccia, visibile in figura 5.8b, sono presenti due box nel quale inserire la nuvole di punti source e target, alcuni pulsanti per il settaggio delle corrispondenze e gli indici dei punti di corrispondenza selezionati (nulli nell'immagine).



(a) (b)

Figura 5.8: Finestre per la gestione di una registrazione manuale.

ICP

L'interfaccia utente per l'utilizzo di ICP è già stata presentata in dettaglio nel capitolo precedente, alla sezione 4.1.1.

Registrazione basata su Features

La registrazione basata su features consiste principalmente nel ricavare punti di interesse da due nuvole tramite un keypoint detector, descriverli tramite un keypoint descriptor e infine, dopo aver trovato delle corrispondenze tra i feaures delle diverse nuvole, trovare la matrice di rototraslazione che permetta il corretto allineamento tra esse.

Le interfacce per il singolo utilizzo di keypoint detectors e per i keypoint descriptor sono già state presentate nei paragrafi 4.2.2 e 4.3.1. A queste sono stati aggiunti dei pulsanti per l'utilizzo del metodo di registrazione basato su features: l'interfaccia complessiva è visibile in figura 5.10.

Innanzitutto si è aggiunto il pulsante "*Registration of 2 PC*" che permette questo tipo di registrazione unicamente tra le nuvole source e target impostate. Allineare una serie di nuvole con questa funzione, dovendo cambiare ad ogni iterazione la nuvola di punti source e quella target, risulta scomodo. Per questo motivo è stata implementata la funzione "Registration All PC", la quale permette di allineare tutte le nuvole con un unico click del mouse. Questa funzione allinea

iterativamente la nuvola che nella lista è allocata all'indice i con quella avente indice $i + 1$, aumentando ad ogni ciclo l'indice di uno. Il primo allineamento sarà effettuato sulla nuvola di punti impostata nella tendina di selezione chiamata in figura "target", in base all'ordine della lista. Si ottiene così una catena di registrazione.

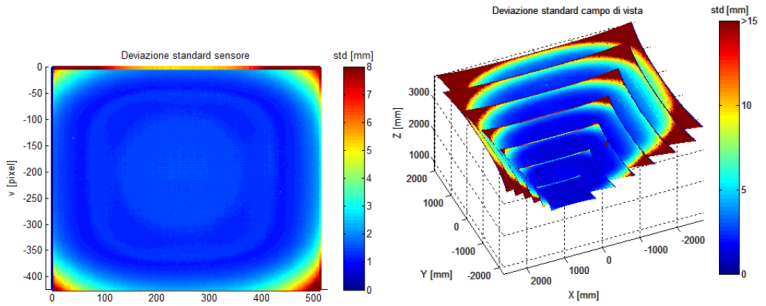
Ovviamente questo processo ha un grosso limite. Si ipotizzi che l'allineamento tra $i-1$ e i non sia corretto, e che sia l'errore di posizionamento ϵ_i tra la nuvola i e la scena reale che essa rappresenta, per la struttura della catena di registrazione, si avrà che tutti gli errori di posizione ϵ_j con $j > i$ saranno maggiori di ϵ_i, cioè $\epsilon_j > \epsilon_i$. All'aumentare del numero di acquisizioni l'errore tende a divergere. É necessario quindi controllare che l'allineamento tra due nuvole introduca un errore minimo.

Per questo motivo viene introdotta la funzione "Reg. Semi-Auto" con la quale sono possibili controlli sul corretto allineamento. Il primo è un controllo di tipo visivo. Alla fine di ogni iterazione vengono visualizzate le due nuvole allineate. La funzione fa comparire in schermata una finestra con la quale viene chiesto all'utente di fermare o continuare il processo. Se il processo viene interrotto sarà possibile ripartire dallo stesso allineamento impostando nella tendina "Target" la nuvola da allineare. Dopo lo stop è possibile migliorare l'allineamento in due modi:

1. l'errore può essere dovuto dalla poca efficacia di un determinato keypoint detector o descriptor utilizzato. Per questo è possibile riprovare ad allineare le nuvole utilizzando algoritmi diversi.

2. se l'errore risulta essere relativamente piccolo ma non ancora accettabile è possibile utilizzare un ICP per rifinire la posizione delle due nuvole. Si ricordi di impostare la distanza massima soglia a pochi centimetri in modo da rendere più veloce l'allineamento.

Il secondo controllo implementato consiste nel fare delle considerazione sulla distanza tra i punti delle nuvole. Viene calcolato il numero di punti della nuvola target che risultano più vicini di una certa distanza soglia d_{max} ad un qualsiasi punto della nuvola source. Se questo numero è maggiore di un valore soglia N allora le due nuvole risultano allineate e il processo continua, altrimenti viene fermato. I parametri di soglia d_{max} e N possono essere impostati ma la loro determinazione risulta tuttora difficile. Questi dipendono infatti dall'area di scena sovrapposta, legata al frame rate di acquisizione e alla velocità di spostamento durante l'acquisizione, e dall'errore accettabile introdotto da ogni allineamento. Introduciamo ora una caratteristica tipica dei sensori a tempo di volo descritto in [60]. Prima di procedere nell'usare tutti i punti derivanti dal Kinect V2 come



(a) (b)

Figura 5.9: Andamento dell'effetto casuale relativo alla misura di distanza valutato all'interno del sensore (a) e del campo di visione (b).

possibili candidati per delle corrispondenze è necessario fare un filtraggio in modo da eliminare i keypoints con una misura di profondità errata. L'errore della misura di profondità non è mediamente costante per ogni punto, ma cresce all'aumentare della distanza dal punto centrale della proiezione sul piano della telecamera della nuvola di punti. Per questo motivo l'utilizzo di tutti i keypoints ricavati può incrementare l'errore di registrazione.

Questa affermazione è supportata dai risultati della qualificazione metrologica dello strumento effettuata in [60]. La prova consiste nel porre il sensore a circa 1250 mm dalla parete e perpendicolare ad essa. Dopo un certo numero di acquisizioni nella stessa posa, viene calcolata la deviazione standard relativa ad ogni pixel acquisito e la si riporta in un'immagine avente lo stesso rapporto (in pixel) del sensore presente all'interno della telecamera IR del Kinect V2. Dalla figura 5.9a si nota come la deviazione standard cresca con il crescere della coordinata radiale. Una minore quantità di luce IR proveniente dalla periferia del cono di illuminazione, comporta una degradazione del rapporto segnale-rumore, il che si traduce, in corrispondenza dei pixel relativi alle zone adiacenti ai vertici del sensore, ad un aumento della rumorosità delle misure. Infine in figura 5.9b sono riportati i risultati ottenuti analizzando sette immagini di profondità relative ad una superficie planare acquisita traslando lungo l'asse Z della camera in un campo di visione esteso da 750 mm a 3750 mm. I punti appartenenti agli angoli del sensore, ai quali è associato un valore di deviazione standard maggiore di 15 mm, sono molto probabilmente relativi a zone della scena non illuminate dal cono di luce IR e quindi i loro valori di distanza, restituiti dal Kinect V2, presentano un'incertezza elevata.

Per questo nel software, impostando due valori che determinano i pixel tolti in u e in v per ogni lato, è possibile eliminare i keypoints che risultano essere

vicini al bordo dell'immagine. Nelle fasi successive di description e stima delle corrispondenze i keypoint eliminati non verranno considerati.

Allineando nuvole successive l'errore tenderà a divergere. Per questo motivo è necessario costruire una rete di corrispondenze tra nuvole che non si limiti al semplice legame temporale di acquisizione, creando cosi una struttura a catena, ma che prenda in considerazione la posizione spaziale di ogni nuvola, rilevando cioè quello che in letteratura è definito come loop closure. Si è implementato dunque nel software un metodo per creare questa rete di corrispondenze e tramite essa correggere la divergenza dell'errore.

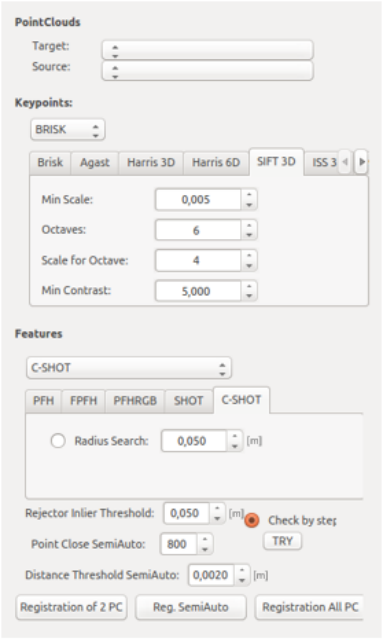


Figura 5.10: Finestra per la registrazione basata su features.

Back-End

Nell'implementazione di questa parte del software viene prima creata una rete di corrispondenze tenendo conto sia di un legame temporale, due nuvole acquisite in successione, sia di un legame spaziale, cioè una certa vicinanza tra i centroidi delle nuvole, poi viene corretta la divergenza dell'errore di allineamento tramite l'algoritmo LUM, come descritto nel paragrafo 2.5.2.

Viene creata una corrispondenza spaziale tra una nuvola A ed una nuvola B unicamente se:

1. la distanza tra i centroidi delle nuvole, d_{AB}, risulta minore di una certa distanza soglia d_{MAX} impostabile;

2. tra le due nuvole sono presenti almeno N acquisizioni, dove N è un valore impostabile;

3. non è presente una nuvola C, che rispetti le prime due condizioni, per cui risulti $d_{AC} < d_{AB}$.

In figura 5.11 è presentata l'interfaccia per l'utilizzo dell'algoritmo. Gli input necessari per l'algoritmo LUM sono tutte le nuvole di punti e una matrice di incidenza che descrive i legami, temporali e spaziali, tra le nuvole.

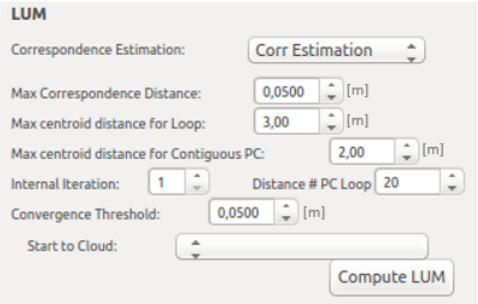


Figura 5.11: Finestra per l'utilizzo dell'algoritmo LUM.

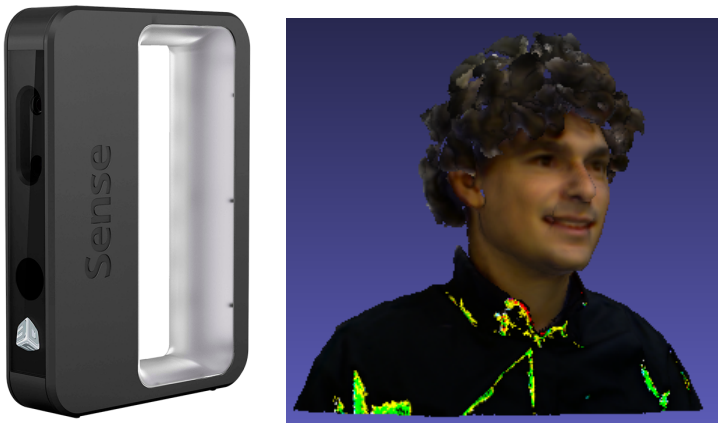
5.3 Strumenti di misura utilizzati

Viene ora svolta un'analisi comparata tra il Kinect V2 ed altri strumenti di acquisizione, tra cui alcuni metrologicamente qualificati. In questa sezione vengono introdotti gli strumenti di misura 3D utilizzati nelle due applicazioni pratiche svolte in questa tesi.

5.3.1 Sense

Il Sense [6], visibile in figura 5.12 è un dispositivo che utilizza la luce strutturata, il cui sensore è identico a quello del sensore del Kinect 1. L'azienda produttrice, la 3DZ, mette a disposizione un software per l'acquisizione e la gestione di questo scanner 3D.

Il sistema di scansione però non restituisce una nuvola di punti, bensì una mesh 3D. Le mesh 3D sono delle griglie 3D composte da una serie di vertici, edge, e triangoli/poligoni. Nello specifico i vertici sono punti nello spazio 3D, gli edge (o segmenti) sono le linee che li connettono fra loro e i poligoni sono le superfici che si vengono a formare quando 3 o più vertici sono uniti da edge. L'acquisizione tramite software proprietario, inoltre, vede il volume di acquisizione limitato ad un cubo $3 \times 3 \times 3$ m.



(a) (b)

Figura 5.12: Dispositivo Sense 3D in (a) e scansione del mio volto (b).

Utilizzando questo dispositivo, se ne è potuto apprezzare l'estrema maneggevolezza e portabilità. Anche l'usabilità è molto buona; il dispositivo dispone di una comoda impugnatura ed è connesso all'elaboratore con l'utilizzo di un solo cavo USB. Diverso è il discorso per il Kinect V2 che deve essere connesso anche all'alimentazione. Le specifiche tecniche di questo dispositivo sono riportate in tabella 5.1.

Caratteristiche principali del Sense.		
Risoluzione Camera IR	[pix]	240x320
Risoluzione Camera RGB	[pix]	240x320
Frame rate massimo	[Hz]	30
Campo visivo	[^{o}]	$45(H)$x$33(V)$x$69(D)$
Range di misura	[mm]	$800 - 3500$
Dimensioni	[mm]	129x178x50
Collegamento		USB 2.0/USB 3.0
Prezzo	[euro]	$\simeq 400$

Tabella 5.1: Sense - caratteristiche principali.

5.3.2 Laser scanner: Minolta VIVID 910

Il Minolta VIVID 910 [7], visibile in figura 5.13 è un laser scanner prodotto da Konica Minolta. Il dispositivo utilizza la tecnologia a lama laser per effettuare la scansione di oggetti (trangolazione attiva). La luce riflessa dal pezzo viene catturata da una telecamera CCD (Charge Couple Device) e poi i dati vengono creati in formato

Figura 5.13: Minolta VIVID 910.

3D sfruttando la triangolazione per determinare la distanza. Per la scansione del raggio laser si utilizza uno specchio galvanometrico ad alta precisione, con una risoluzione di 640×480 pixel. Sono integrati gli innovativi chip di immagine ad alta velocità Konica Minolta, per garantire scansioni relativamente rapide per questo tipo di dispositivi: in soli $2,5$ secondi si completa una scansione. Il punto di forza di questo strumento è l'accuratezza di misura che risulta inferiore a $\pm 0,1$[mm] a 60 centimetri di distanza. Le altre specifiche sono riportate in tabella 5.2.

Caratteristiche principali del VIVID 910.		
Risoluzione	[pix]	640x480
Tempo di acquisizione	[s]	da 0.3 fino a 2.5
Risoluzione RGB	[pix]	640x480
Campo visivo	[mm]	da 360x270 fino a 900x1200
Range di misura	[mm]	$600 - 2000$
Dimensioni	[mm]	213x413x271
Peso	[Kg]	11
Prezzo	[euro]	$\simeq 30000$

Tabella 5.2: VIVID 910 - principali caratteristiche

Durante le prove, si sono notati dei grossi limiti per l'utilizzo di questo strumento nella ricostruzione di una scena ampia.

- A differenza dello Stonex X300, il campo visivo è limitato. Per questo richiede un tempo per l'acquisizione dell'intera scena elevato, ed ha una fase di post processing computazionalemente onerosa sia per il tempo di calcolo che per l'attività richiesta all'operatore.

- Lo strumento non è utilizzabile a mano e necessita di un cavalletto molto stabile per il suo sostegno.
- Il range ottimale di misurazione risulta tra i 0.6 e i 2 metri, il che non rende possibile l'acquisizione e di ampie scene, scopo della tesi.

5.3.3 Laser scanner: Stonex X300

Lo Stonex X300 [5] è uno scanner laser a tempo di volo, utilizzato soprattutto in ambiti archeologici, di monitoraggio e di scansioni ambientali geografiche. All'interno dell'apparecchio è presente uno specchio posto in rotazione per mezzo di un motore elettrico, attorno ad un asse orizzontale, così da deflettere il raggio laser all'interno di un semipiano ed acquisire punti aventi il medesimo angolo. Contemporaneamente il dispositivo ruota attorno ad un asse verticale permettendo così di ottenere, grazie ai due movimenti combinati, un insieme di punti 3D. Nell'immagine 5.14 è possibile vedere a sinistra il dispositivo e a destra l'acquisizione a 360^o di una piazza.

(a) (b)

Figura 5.14: Dispositivo Stonex X300 in (a) e scansione di Piazza della Loggia a Bergamo in (b)

Il dispositivo è dunque in grado con una unica acquisizione di rilevare l'intera stanza da un unico punto di vista. Questo può rivelarsi un problema poiché se la scena è composta da oggetti a distanze diverse possono crearsi delle zone d'ombra nella scena. Per questo è utile eseguire più acquisizioni con punti di vista differenti e poi registrarli assieme.

Le specifiche tecniche di questo dispositivo sono riportate in tabella 5.3.

Caratteristiche principali di Stonex X300.		
Risoluzione Angolare	[']	1.35(H)x1.35(V)
Accuratezza a 50m	[mm]	< 6
Tempo di acquisizione	[punti/sec]	40000
Tempo di rotazione 20^o	[min]	1
Campo visivo Orizzontale	[^{o}]	360
Campo visivo Verticale	[^{o}]	90
Range di misura	[m]	$1.6 - 300$
Dimensioni	[mm]	215x170x430
Peso	[Kg]	6.15
Prezzo	[euro]	$\simeq$ 30000

Tabella 5.3: Stonex X300 - principali caratteristiche

5.4 Ricostruzione reperto archeologico

In questi ultimi anni, con l'avvento di nuove tecnologie, anche il sistema di istruzione universitario è cambiato. Proiettori, tablet, e lavagne multimediali hanno aumentato la richiesta di applicazioni interattive per garantire un miglior apprendimento e poter sfruttare a pieno queste tecnologie. Inoltre, la possibilità di visualizzare tridimensionalmente un oggetto permette una maggior comprensione dello stesso. Per questo sono sempre più richieste scansioni 3D che possano ricostruire al meglio opere d'arte, sculture e reperti archeologici. Il primo esempio di applicazione di questa tesi si riferisce a quest'ultimo ambito.

5.4.1 Descrizione dell'applicazione

Nel 2004 , in località Torretta di Noceto, un paese vicino a Parma, è stata scoperta una grande vasca rivestita di legno che non solo rappresenta una poderosa opera di ingegneria e carpenteria dell'età del Bronzo, ma il cui significato va probabilmente cercato nella sfera del sacro. Il reperto in eccezionale stato di conservazione, risalente al 1500 a.C., è visibile in figura 5.15.

Di questo enorme reperto, $12 \times 7 \times 3$ m, è stata eseguita diversi anni fa una ricostruzione tridimensionale, utilizzata principalmente in ambito scolastico ed educativo. Recentemente ulteriori scavi hanno portato alla luce una vasca limitrofa, sempre in legno, di dimensioni $9 \times 2 \times 3$ m.

Il direttore dei lavori, il professore Mauro Cremaschi dell' Università Statale di Milano, ha richiesto un rilievo tridimensionale del reperto a scopo didattico divulgativo. Il primo ambito applicativo della tesi dunque consiste nella ricostruzione 3D di questa seconda vasca, in modo poi da poterla allineare con la prima e di

Figura 5.15: Sito archeologico di Noceto.

avere un'unica rappresentazione 3D dell'intero sito archeologico. Per questo scopo gli strumenti a disposizione sono stati il laser scanner Minolta VIVID 910, il Sense ed il Kinect V2.

Una volta in loco però, osservando l'area di ricostruzione, si è compresa l'effettiva impossibilità dell'utilizzo del Minolta. Ricordiamo che il range di misura dello strumento ha limite massimo di 2 metri. Poiché intorno al reperto non vi erano zone pianeggianti, come visibile in figura 5.16, per effettuare una scansione con il Minolta si sarebbe dovuto installare il cavalletto nel mezzo del reperto rischiando di rovinarlo. Inoltre la conformazione della vasca necessita di un'acquisizione "dall'alto" impossibile da effettuare per il Minolta. Anche se l'acquisizione fosse stata possibile, con un campo visivo massimo di 900×1200 mm, si sarebbero impiegate molte ore. Per questi motivi le uniche ricostruzione realizzare sono quelle ricavate tramite Sense e il Kinect V2.

Figura 5.16: Foto scattata durante le acquisizioni del sito archeologico di Noceto.

5.4.2 Risultati con Sense

Inizialmente il sito archeologico è stato scansionato con il Sense. Il sistema di acquisizione, come già accennato, restituisce una mesh 3D, non una nuvola di punti, e limita il volume di acquisizione ad un cubo $3 \times 3 \times 3$ m. Sono state eseguite perciò 16 acquisizioni che poi sono state registrate tra loro.

Non risulta possibile, tuttavia, riallineare direttamente le mesh; è infatti necessario ricavare una nuvola di punti da essa per poi riallinearla. Le nuvole ricavate da questa operazione sono poi state riallineate dal Laboratorio di Antropologia e Odontoiatria Forense LABANOF, situato presso la Sezione di Medicina Legale del Dipartimento di Morfologia Umana e Scienze Biomediche Università degli Studi di Milano, che negli ultimi anni ricostruisce scene del crimine con la stessa tecnica.

Il software utilizzato per l'allineamento è Geomagic, uno tra i programmi più utilizzati per la gestione di nuvole di punti 3D che, utilizzando i relativi punti, riesce ad allineare le mesh. In figura 5.17 è possibile osservare l'output di questo processo. La registrazione risulta essere abbastanza corretta, ma il dettaglio di acquisizione del Sense porta l'intera ricostruzione ad essere poco apprezzabile.

Il Sense inoltre è molto sensibile alla variazione di luce; nell'immagine infatti sono presenti delle scansioni mediamente più chiare dovute ad aree più illuminate rispetto ad altre.

Figura 5.17: Ricostruzione di Noceto tramite l'utilizzo del Sense.

5.4.3 Risultati con Kinect V2 e RTAB-Map

In secondo luogo, lo scavo è stato acquisito tramite il software RTAB-Map ed il dispositivo Kinect V2. Il software estrae e descrive i keypoints per mezzo di algoritmi 2D, ricerca tra essi le corrispondenze che poi associa ai medesimi punti 3D trovati mediante la matrice di proiezione. L'utilizzo di soli algoritmi 2D garantisce una rapidità di registrazione che permette la visione della ricostruzione delle scene in tempo reale.

Si è anche notato come questo software sia robusto a repentini cambi di visuale. Quando infatti non trova abbastanza corrispondenze tra due immagini il processo va in pausa e segnala il mancato allineamento cambiando in rosso il colore dello sfondo del visualizzatore. A questo segnale l'utente può ritornare alla posa precedente e riprendere la registrazione dal punto in cui si era fermata. Questo permette un feedback ed una relativa sicurezza sulla buona riuscita dell'operazione.

La ricostruzione della scena è visibile in figura 5.18.

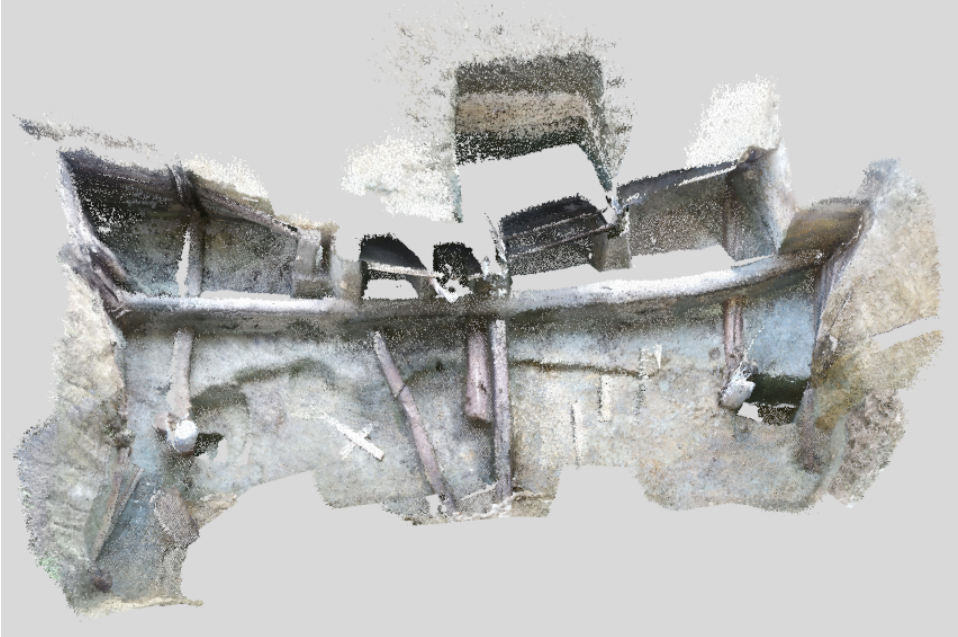


Figura 5.18: Ricostruzione di Noceto tramite l'utilizzo di RTAB-Map.

5.4.4 Risultati con Kinect V2 e Software Implementato

Grazie al software appositamente utilizzato, l'acquisizione risulta essere molto veloce. A differenza però di RTAB-Map non si ha un riscontro visivo in loco sulla buona riuscita della registrazione.

Nei giorni successivi all'acquisizione dunque si è cercato di allineare le nuvole con i diversi metodi di registrazione già descritti. Sapendo che il solo utilizzo dell'algoritmo ICP avrebbe impiegato un tempo eccessivo (paragrafo 4.1), si è pensato di utilizzare la registrazione features based per ottenere un allineamento grezzo prima di ricorrere ad un ICP di rifinitura.

A questo punto la difficoltà risiede nel scegliere l'accoppiamento tra i diversi metodi di keypoint detection e keypoint description, utilizzato per la registrazione a catena di nuvole. Pur provando diversi accoppiamenti, processi di registrazioni a catena non controllate hanno restituito ricostruzioni non accettabili. Se infatti in una registrazione a catena l'allineamento tra la nuvola i e la $i+1$ risulta errata, anche le posizioni di tutte le nuvole j con $j > i$ saranno errate.

Per questo è stato deciso di controllare la buona riuscita di ogni allineamento prima di passare al successivo. Grazie a questo controllo è stato notato come non esista un accoppiamento di metodi che fornisca un allineamento migliore di un altro, ma questo dipenda dal tipo di nuvole processate.

Per maggior chiarezza si faccia riferimento ai principali accoppiamenti utilizzati riassunti in tabella:

	Keypoint detector	**keypoint descriptor**
Processo 1	BRISK	C-SHOT
Processo 2	SIFT 3D	C-SHOT
Processo 3	SIFT 3D	FPFH

Tabella 5.4: Diversi tipi di pipeline utilizzati per la registrazione nel software implementato.

Se le nuvole rappresentano una superficie simile ad un piano, l'unico processo utilizzabile è il primo, poiché non vi sarà una grossa distinzione di forma. Se invece la nuvole di punti da allineare rappresentano travi o forme distinguibili si utilizzano preferibilmente i processi 2 e 3. Quando invece vengono riprese forme con diedri scuri, come per esempio l'angolo in figura 5.20, nessuno dei processi fornisce un allineamento accettabile.

Il problema è dovuto all'errore di misura introdotto dalle riflessioni multiple della telecamera a tempo di volo Kinect V2 discusso in 4.1. Si noti che RTAB-Map non è affetto da questo problema poiché, utilizzando keypoint descriptors 2D, non si basa sulle caratteristiche della superficie in cui il keypoint è immerso.

L'unico metodo utilizzabile per ottenere un corretto allineamento in queste zone dunque risulta essere la registrazione manuale; per questo l'intera fase di registrazione è durata circa 9 ore, nella quale si sono allineate circa 300 nuvole di punti, ottenendo la nuvola complessiva in figura 5.19.

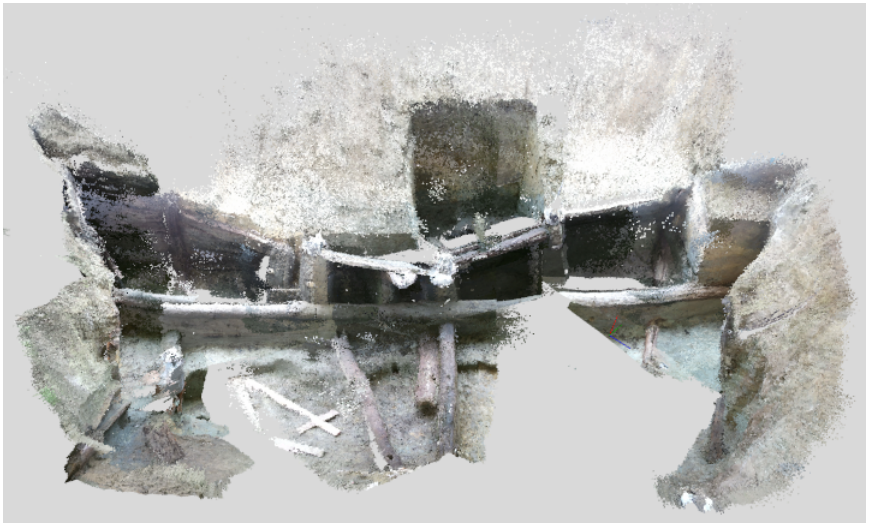


Figura 5.19: Ricostruzione di Noceto tramite il software implementato.

Questa risulta essere complessivamente corretta. Sono presenti però alcuni errori di allineamento commessi dall'utente durante il processo, uno di essi rappresentato in figura 5.20.

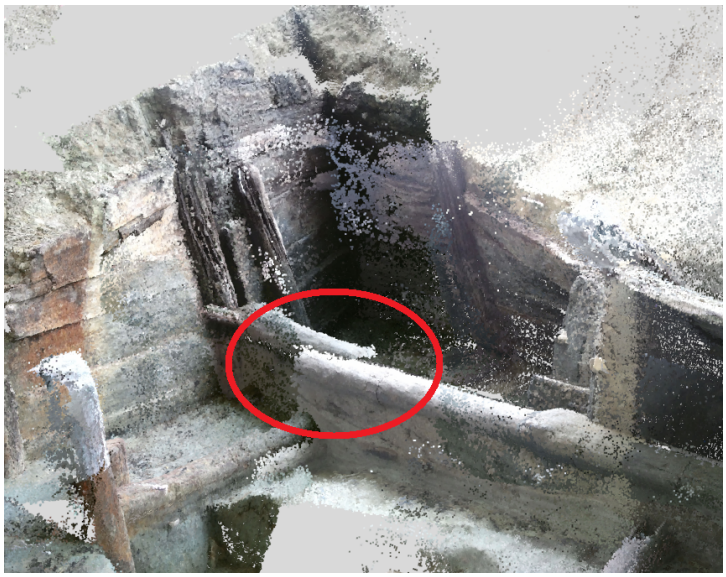


Figura 5.20: Allineamento errato di due nuvole di punti registrate manualmente.

La ricostruzione può essere definita ancora accettabile, anche se leggermente meno accurata di quella ricavata da RTAB-Map, ma lo stesso non si può dire del tempo impiegato per ricavarla, che risulta essere nettamente superiore.

In figura 5.21 vengono proposti dei dettagli della ricostruzione in modo da avere un idea dell'accuratezza dei due sensori. Mentre il Sense risulta essere poco accurato, nelle acquisizioni fornite dal Kinect V2 è possibile distinguere persino le crepe nel legno.

(a) Sense

(b) RTAB-Map e Kinect V2

(c) Software e Kinect V2

Figura 5.21: Dettaglio delle ricostruzioni di Noceto con diversi sistemi: Sense (a), Software e Kinect V2 (b), RTAB-Map e Kinect V2 (c).

5.5 Ricostruzione scena del crimine

Congelare la scena del crimine è sicuramente una delle prime, più importanti e delicate operazioni che deve essere messa in atto. Questa operazione permette infatti di cristallizzare la posizione degli oggetti, con lo scopo di preservare la scena del crimine dalle inevitabili contaminazioni e, soprattutto, per poter tenere traccia dei dati che saranno poi necessari nella successiva operazione di ricostruzione della scena del crimine nel modo più fedele possibile durante il processo investigativo.

Una delle procedure che viene utilizzata per assicurare il congelamento della scena del crimine è quella del rilievo topografico, ovvero la rappresentazione dimensionale dello stato dei luoghi. Nella grande maggioranza dei casi il rilievo viene realizzato nella maniera più semplice, ovvero con carta, matita e metro a nastro, disegnando schizzi e planimetrie per riportare la posizione di vittime, ambienti, oggetti e corpi di reato.

Oltre al semplice disegno planimetrico bidimensionale si è cominciato a comprendere l'utilità della ricostruzione in 3D, la quale è in grado anche di far capire le dinamiche evolutive di un certo evento. In questo modo è possibile ricostruire traiettorie di proiettili, effettuare analisi delle macchie di sangue, ipotizzare dinamiche dell'accadimento, fino a poter spostarsi virtualmente all'interno della scena ricostruita.

Da qualche anno il LABANOF, ha spinto la ricerca su sistemi in grado di ottenere una ricostruzione 3D di una scena indoor.

Il secondo ambito applicativo di questa tesi nasce dunque come un progetto di collaborazione tra il laboratorio LABANOF e il Politecnico di Milano con lo scopo di valutare e confrontare diverse tecniche per la ricostruzione di una scena del crimine utilizzando strumenti economici e di facile utilizzo.

Per confrontare le prestazioni di alcuni strumenti è stata ricreata una scene del crimine nel laboratorio del Politecnico di Milano in cui è stata principalmente eseguita questa tesi, il Vision Brick LAB (VBLAB). Della scena sono state eseguite delle prove di ricostruzione 3D con diverse tecniche e strumenti le cui caratteristiche sono riportate in tabella 5.5. Per rendere la scena più caratteristica si è usato del sangue finto e un manichino realistico come vittima, entrambi appositamente forniti dal dipartimento di Medicina Legale.

I dispositivi utilizzati per la ricostruzione della scena del crimine sono: il laser scanner Stonex X300, poiché risulta essere accurato ed ha capacità di acquisire la scena con un'unica scansione di 360^o; i dispositivi Sense e Minolta poiché tuttora utilizzati dal laboratorio LABANOF per ricostruire vere scene del crimine; e infine Kinect V2 per studiarne le sue potenzialità.

Prima di introdurre le prove svolte, sono necessarie delle precisazioni. Per ricavare le acquisizioni dallo strumento Minolta VIVID 910 è necessario l'utilizzo

	Scena 3D totale	Dettaglio	Portabilità	Prezzo
Minolta VIVID 910		X		
Sense			X	X
Stone X300	X	X		
Kinect V2 e RTAB-Map	X		X	X
Kinect V2 e Software	X		X	X

Tabella 5.5: Vantaggi di ogni sistema di ricostruzione.

di un software proprietario, chiamato PET. Esso permette la registrazione di nuvole acquisite consecutivamente. Considerando che ogni nuvola ha dimensioni di circa 700 MB, dopo poche acquisizioni la RAM è satura e i processi diventano lenti o impossibili. Per questo non è possibile tramite esso la ricostruzione di un'intera scena. Tuttavia è uno strumento molto apprezzato per la ricostruzione oggetti con un elevato grado di dettaglio, come un'arma, un'impronta o un'area intrisa di sangue.

L'altro strumento già impiegato da Labanof è il Sense. Questo permette l'acquisizione di un cubo di $3 \times 3 \times 3$ m, quindi anche in questo caso non è possibile ottenere una scena completa a meno di non registrare le diverse scansioni. In output infatti restituisce i dati organizzati sotto forma di mesh rendendo la registrazione un processo più complesso. Un vantaggio, però, di questo strumento è il fatto di essere molto leggero e maneggevole così da poter raggiungere anche zone anguste. Inoltre durante l'acquisizione non sono presenti cavi, oltre a quello tra il Sense e computer, che limitano il movimento durante l'acquisizione o rischiano di modificare prove presenti nella scena.

Il terzo strumento utilizzato è il Kinect V2. Come già osservato nelle prove di Noceto, esso ha fondamentalmente la stessa portabilità di Sense ma ha una maggior risoluzione e accuratezza. Le acquisizioni verranno registrate sia con RTAB-Map sia con il software implementato.

L'ultimo strumento utilizzato è lo Stonex X300. La possibilità di ricostruire tutta la scena in un'unica scansione rende la nuvola di punti in uscita dallo Stonex X300 complessivamente molto accurata poiché è esente da errori dovuti alla catena di registrazioni. Per questo viene preso come riferimento nelle prove metrologiche che verranno effettuate. Abbiamo già visto come, con questa tipologia di scanner, sia necessario effettuare più scansioni per limitare le zone d'ombra. Una scansione di 360^o con questo strumento è formata però da circa 3 milioni di punti, a differenza di quella acquisita mediante Kinect V2 che ne ha 217088, per questo motivo la registrazione risulta difficile.

Lo svantaggio di questo strumento è l'ingombro. Pur avendo dimensioni e peso accettabili, per rendere stabile lo strumento durante l'acquisizione dinamica,

è necessario un cavalletto ingombrante. In foto 5.22, è possibile vedere come, a volte, risulti difficile posizionare il dispositivo senza modificare la scena.

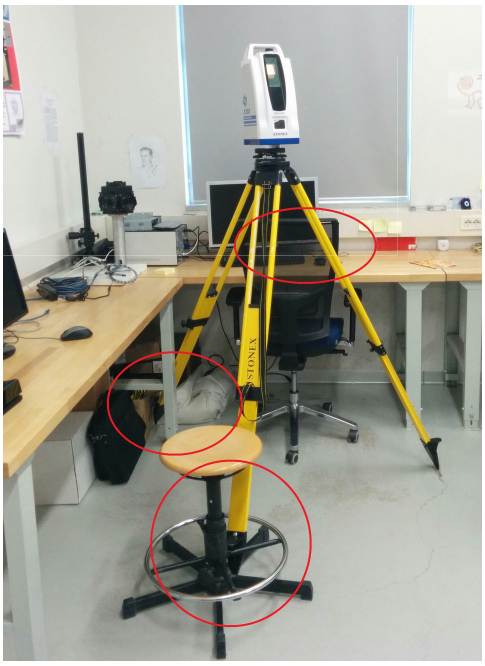


Figura 5.22: Esempio in cui il dispositivo Stonex X300 è risultato ingombrante.

5.5.1 Ricostruzione Completa

In primo luogo verranno comparate le ricostruzioni dell'intera stanza ricavate con i diversi strumenti. Per i problemi già descritti gli strumenti Minolta VIVID 910 e Sense non sono stati utilizzati per la ricostruzione della scena completa.

Al fine di confrontare le diverse scansioni con i diversi strumenti si è deciso di acquisirle nel modo più simile possibile. Allo scopo è stato posizionato un cavalletto in mezzo alla stanza sopra al quale è stato montato un Kinect V2. Le acquisizioni sono state effettuate, sia con il software implementato sia con RTAB-Map, iniziando la scansione dallo medesima posizione e ruotando intorno all'asse perpendicolare al pavimento fino a percorrere una rotazione di 360^o. Nella stessa posizione poi si è acquisita la scena, in un'unica scansione, tramite lo Stonex X300.

A differenza di Noceto, questa scena risulta avere forme con spigoli, angoli, geometrie definite e colori più decisi. Per ottenere una ricostruzione con il software implementato è risultata sufficiente una catena di registrazioni basata su features automatica, senza il controllo dell'utente. In particolare sono stati utilizzati gli algoritmi BRISK per la rilevazione di keypoints e C-SHOT per la relativa descrizione.

La registrazione di 250 nuvole di punti è durata circa 26 minuti, cioè quasi 6 secondi per ogni registrazione.

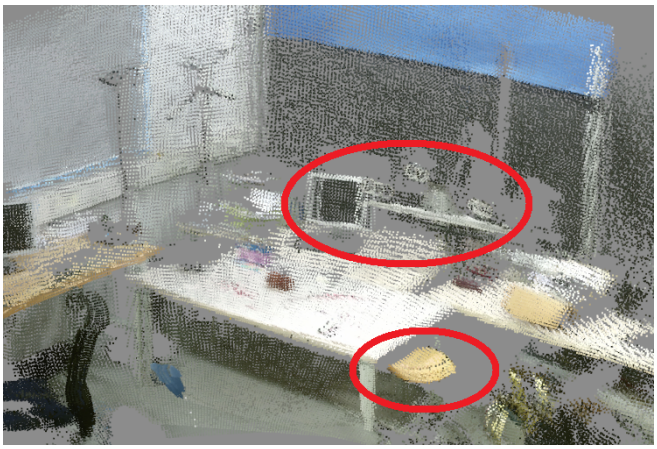
A onor del vero, anche se la quasi totalità delle registrazioni è stata eseguita nella maniera corretta, a circa metà del processo, un allineamento impreciso ha introdotto un errore che ha portato le nuvole successive ad essere leggermente scostate dalla posa ottimale. Per questo motivo abbiamo introdotto nel processo la possibilità di rifinire l'allineamento dopo ogni operazione tramite un ICP. Questa fase però allunga il tempo di calcolo, impostando 5 iterazioni ad ogni ICP, l'intero processo durato circa 2h.

Mentre la ricostruzione della scena con il software implementato ha necessitato di una fase di post processing di circa 2h, con RTAB-Map, grazie all'utilizzo di algoritmi 2D, l'allineamento è risultato completamente automatico e immediato. RTAB-Map inoltre grazie ad un sistema di memoria ottimizzato, riconosce se un certo features, e quindi un punto reale, è stato già rilevato in precedenza e imposta quindi un loop closure riallineando le nuvole all'interno di esso. Il grosso vantaggio del software RTAB-Map è, oltre al tempo di processo, la possibilità di avere un riscontro diretto e immediato della buona riuscita della ricostruzione.

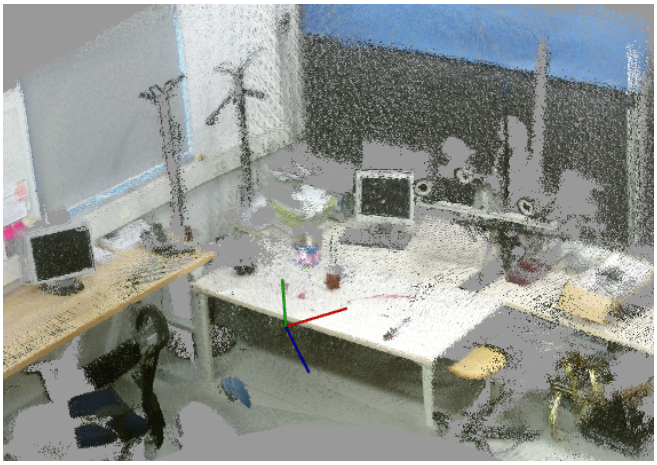
Analizzando le figure in 5.23 vengono proposte ulteriori considerazioni. Nelle immagini è possibile osservare un dettaglio della scena ricostruita dai tre sistemi di acquisizione. Anche se RTAB-Map risulta essere molto rapido, in alcuni casi restituisce allineamenti errati, come è visibile dalla prima figura (5.23a). Nel programma è possibile eseguire una lunga fase di post-processing mediante un ICP che rifinisce l'allineamento tra due nuvole successive, come quello eseguito dal software implementato. Nello specifico delle prove non è stato possibile utilizzare questa opzione in quanto il numero di nuvole acquisite ha portato rapidamente la RAM dell'elaboratore a saturarsi.

A differenza di RTAB-Map, la ricostruzione effettuata dal software implementato durante la tesi, risulta essere più coerente con la realtà, come è visibile in figura 5.23b.

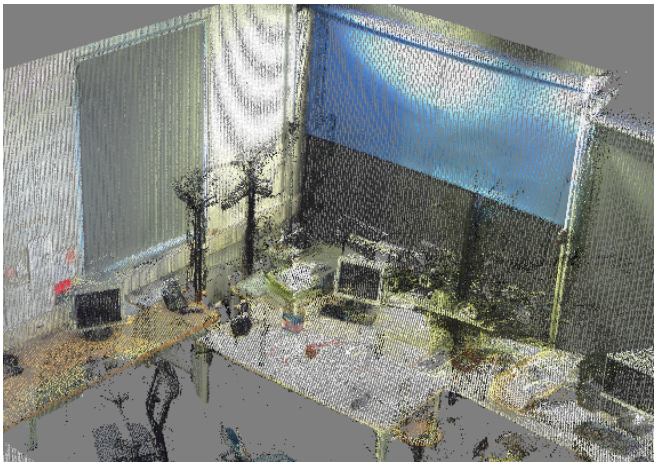
Utilizzando queste tecniche di ricostruzione è necessaria una fase di registrazione che risulta essere computazionalmente complessa e affetta da errori. Con il laser scanner Stonex X300 invece è possibile ottenere un'unica scansione senza dover passare da questa fase rendendo così la ricostruzione più accurata, come riportato in figura 5.23c.



(a) RTAB-Map

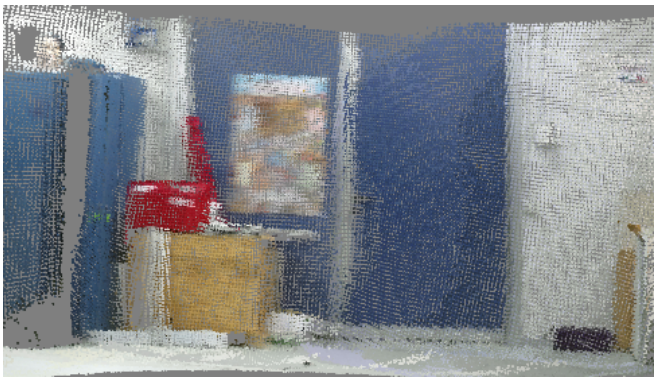


(b) Software implementato

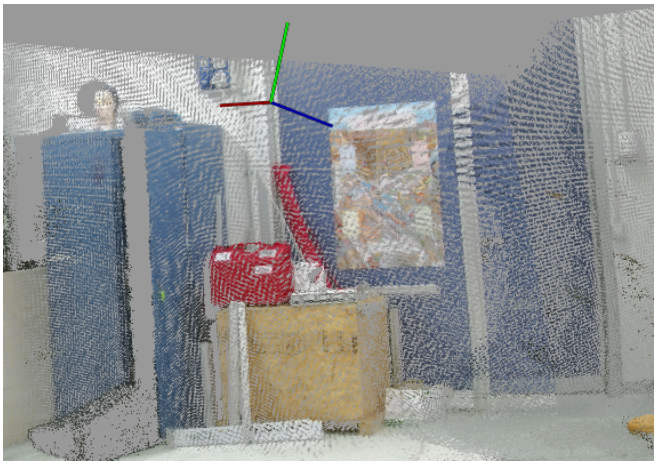


(c) Stonex X300

Figura 5.23: Stesso dettaglio ricostruito con diverse tecniche



(a) RTAB-Map



(b) Software implementato

(c) Stonex X300

Figura 5.24: Dettaglio della scena in cui avviene il loop closure, ricostruito con diverse tecniche.

Per come è stata eseguita la prova è chiaro come le prime e le ultime nuvole acquisite rappresentino la stessa porzione di stanza. RTAB-Map, tramite un algoritmo di loop closure detection, trova delle somiglianze tra la prima e l'ultima nuvola e tra esse aggiunge una corrispondenza. Nella fase successiva di back-end poi le sovrappone ridistribuendo l'errore tra tutte le nuvole presenti nella catena di registrazione.

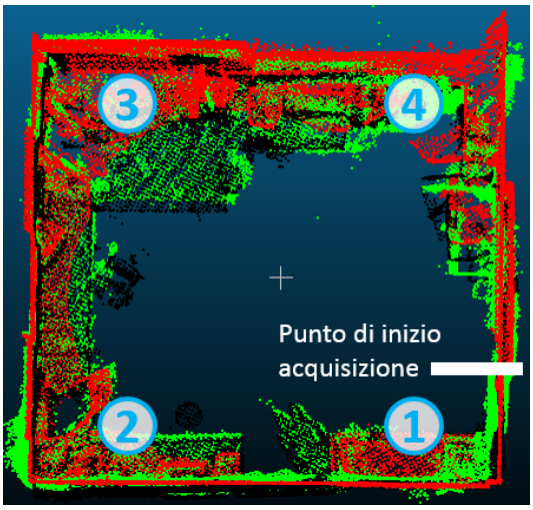
Nella ricostruzione ricavata dal software implementato invece non è presente, per ora, un algoritmo di backend. Si prevede quindi che nella parte di scena in cui è iniziata (e finita) l'acquisizione, RTAB-Map ricostruisca la scena in maniera più coerente con la realtà. Mettendo infatti a confronto l'area d'interesse delle diverse ricostruzioni, visibile in figura 5.24, è possibile notare come, grazie all'algoritmo di loop closure detector, RTAB-Map si comporti meglio del software implementato.

Confronto tra errori di propagazione

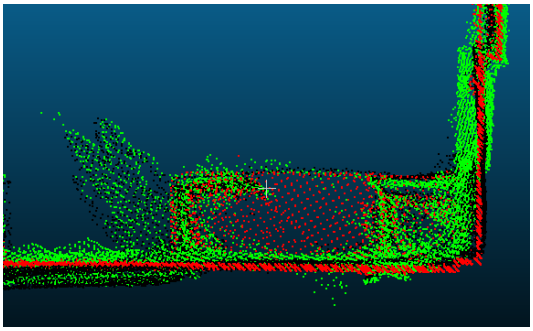
Per comprendere la diversa propagazione dell'errore tra le tecniche utilizzate vengono ora effettuate delle prove che mettono in relazione le nuvole ricostruite dai tre diversi sistemi.

Innanzitutto si è usato un filtro (Voxel Grid) per diminuire i punti delle nuvole ricavate, in quanto la gestione di 3 nuvole da 3 milioni di punti risulta essere pesante per il visualizzatore. Tramite poi il software chiamato *Cloud Compare* vengono allineate manualmente le diverse ricostruzioni, prendendo come nuvola di riferimento quella derivata dal laser scanner Stonex poiché, come già osservato, risulta più accurata. Le corrispondenze impostate per l'allineamento sono tutte state prese dall'area relativa al primo angolo della stanza incontrato durante l'acquisizione. I primi punti di ogni ricostruzione, dunque, saranno tra loro allineati, mentre quelli successivi si allontaneranno dal riferimento in funzione dall'errore di propagazione. Viene utilizzato un allineamento manuale poiché, per stimare l'errore di propagazione, è necessario che solo una parte della nuvola, il primo angolo, abbia vincoli di allineamento.

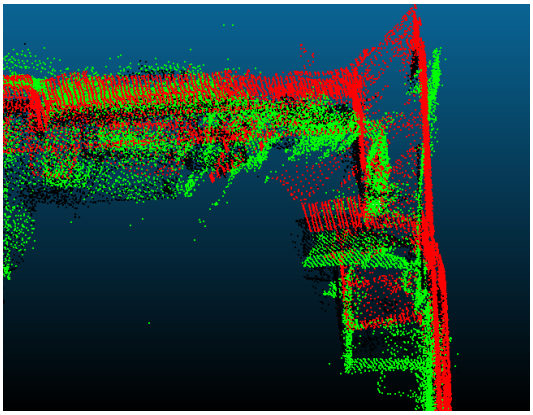
In figura 5.25a è possibile visualizzare le ricostruzioni dall'alto, in cui, per maggior chiarezza, sono stati inseriti i numeri che definisco la successione temporale di acquisizione degli angoli della stanza. É possibile ricavare un'ulteriore prova dell'accuratezza di Stonex X300 dal fatto che i punti della nuvola appartenenti alle pareti della stanza formano sulla pianta rette tra loro ortogonali.



(a) Visione dall'alto delle tre ricostruzioni.



(b) Dettaglio del confronto tra piante: primo angolo, le nuvole risultano sovrapposte.



(c) Dettaglio del confronto tra piante: quarto angolo, è visibile l'errore di propagazione, le nuvole si discostano .

Figura 5.25: Confronto tra le piante di diverse ricostruzioni: in rosso con Stonex X300, in nero con RTAB-Map e in verde con il software implementato.

Si concentri l'attenzione ora sul primo ed il quarto angolo della stanza. Come già detto, nel primo, in figura 5.25b, le nuvole risultano, per costruzione, essere sovrapposte. Nel quarto invece le ricostruzioni sono affette da errori di registrazione e, in funzione di essi, risultano essere tra loro disallineate.

In figura 5.25c è possibile osservare come si comportano le ricostruzioni nel quarto angolo. Prendendo come riferimento la ricostruzione con il laser scanner Stonex X300, si stima l'errore di propagazione delle tecniche utilizzate in funzione dello scostamento tra il riferimento e le nuvole ricavate da esse. Si può notare come la ricostruzione con RTAB-Map si discosti in modo minore dal riferimento rispetto a quella ottenuta tramite la sola registrazione a catena proposta dal software implementato.

La conclusione più semplice a cui è possibile arrivare è che RTAB-Map restituisca una miglior ricostruzione della scena rispetto al software implementato. Questo, però, non è del tutto esatto.

Si supponga che il perimetro della stanza vista dall'alto sia un rettangolo, come quello in figura 5.26a. Utilizzando un software per la ricostruzione basato su una catena di registrazioni è possibile che l'allineamento tra l'i-esima e la $i+1$-esima nuvola introduca un errore consistente, rappresentato dal cerchio grigio in figura 5.26b. Un errore di questo tipo è visibile in figura 5.23b nella ricostruzione di RTAB-Map. Si introduca poi un loop closure tra la prima e l'ultima nuvola. La fase di Backend non agisce sulla correzione tra l'i-esima e la $i+i$-esima nuvola, ma ridistribuire l'errore tra tutte le nuvole di punti in parti uguali. Il risultato può essere rappresentato come in figura 5.26c.

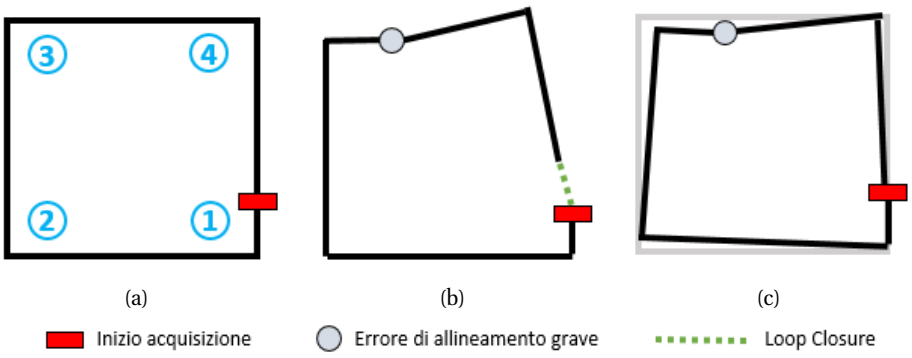


Figura 5.26: Rappresentazione del problema di loop closure.

Nella prova realizzata, l'errore visibile in figura 5.23b nella ricostruzione di RTAB-Map avviene tra il terzo e il quarto angolo. Questo significa che il loop closure ridistribuendo l'errore tra tutta la catena di nuvole, distorce anche le posi-

zioni corrette delle nuvole calcolate prima dell'allineamento scorretto e migliora le successive.

Confrontando infatti il comportamento delle diverse ricostruzioni nel terzo angolo, in figura 5.27, è possibile notare come la nuvola di punti ricavata utilizzando il software implementato (in verde) sia più simile al riferimento (in rosso), in confronto con quella ricavata da RTAB-Map (in nero).

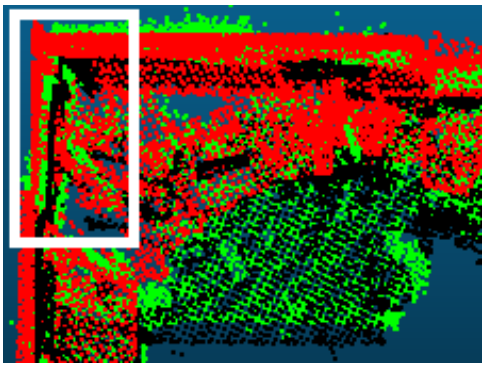


Figura 5.27: Dettaglio del confronto tra piante: terzo angolo, è visibile l'errore introdotto dal loop.

Dopo le considerazioni qualitative effettuate vengono stimati gli errori di propagazione nelle ricostruzioni ottenute dai due software che utilizzano la registrazione. In RTAB-Map è possibile ricavare una ricostruzione senza che venga trovato il loop closure e ottimizzato il grafo, utilizzando quindi unicamente la fase Front-End del processo. Anche da tale ricostruzione verrà ricavato l'errore di propagazione, per avere un confronto tra le fasi Front-End dei software RTAB-Map e il software implementato.

Gli errori di propagazione si dividono in errori di traslazione ed errori di rotazione. Entrambe le tipologie di errore vengono calcolate attraverso diversi passaggi:

1. Viene creata una nuvola di punti, $PC1_{copy}$, identica alla prima acquisita.
2. La nuvola $PC1_{copy}$ viene aggiunta alla fine della catena di registrazione e quindi allineata all'ultima nuvola di tale catena.
3. Viene calcolata tramite un ICP la matrice di rototraslazione T tra la prima nuvola e $PC1_{copy}$ che avrà la forma:

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. I valori t_x, t_y, t_z indicano la traslazione nei rispettivi assi tra le due nuvole. Da essi viene calcolata l'errore di traslazione d che rappresenta la distanza dei baricentri delle due nuvole:

$$d = \sqrt{(t_x)^2 + (t_y)^2 + (t_z)^2}$$

5. Gli errori di rotazione sono stimati con gli angoli di θ_z, θ_y e θ_x che rappresentano rispettivamente la rotazione attorno a z, a y e a x:

$$\theta_z = \text{atan}_2\left(r_{21}\ ,\ r_{11}\right)$$
$$\theta_y = \text{atan}_2\left(-(r_{31})\ ,\ \sqrt{r_{32}^2 + r_{33}^2}\right)$$
$$\theta_x = \text{atan}_2\left(r_{32}\ ,\ r_{33}\right)$$

   Gli angoli vengono riportati in gradi per facilitarne la comprensione.

Si sottolinea l'importanza di calcolare la matrice di rototraslazione tra due nuvole identiche, affinché tale matrice dipenda unicamente dall'errore dovuto alla catena di registrazione e non da una diversa posa di acquisizione.

Per ogni ricostruzione della scena vengono calcolati l'errore di traslazione e i tre errori di rotazioni, riportati nelle tabelle 5.6 e 5.7.

	$t_x\ [m]$	$t_y\ [m]$	$t_z\ [m]$	$d\ [m]$
Software Implementato	0.179	0.117	0.098	0.235
RTAB-Map Front-End	-0.094	0.324	-0.057	0.342
RTAB-Map	-0.015	-0.019	0.011	0.026

Tabella 5.6: Risultati della stima dell'errore di traslazione.

	$\theta_z\ [^o]$	$\theta_y\ [^o]$	$\theta_x\ [^o]$
Software Implementato	-2.297	2.6467	1.3235
RTAB-Map Front-End	2.006	-0.343	-2.693
RTAB-Map	0.572	0.114	0.057

Tabella 5.7: Risultati della stima dell'errore di rotazione.

L'utilizzo di un loop closure in RTAB-Map, porta ad errori di propagazione, sia in termini di traslazione che rotazione di molto inferiori rispetto ai metodi che non lo utilizzano. Infatti mentre RTAB-Map, sfruttando un approccio Back-End, presenta degli errori di rotazione nell'ordine dei decimi di grado e l'errore di traslazione nell'ordine dei centimetri, gli altri due metodi presentano errori di un ordine di grandezza superiore.

Viene ora effettuato un confronto unicamente tra le fasi Front-End dei due algoritmi, prendendo in considerazione gli errori di propagazione ricavati dalla ricostruzione del software implementato e da quella ottenuta con RTAB-Map senza la rilevazione di loop closure. Confrontando la distanza tra i baricentri possiamo notare che l'errore commesso da RTAB-Map è maggiore del 50% rispetto a quello commesso dal software implementato. Gli errori di rotazione, invece, sono dello stesso ordine di grandezza. Dalle prove effettuate risulta che, considerando unicamente la parte Front-End dei processi, il software implementato ricostruisce la scena in modo più coerente con la realtà rispetto a RTAB-Map.

Planarità e Ortogonalità delle pareti

Vengono ora effettuate ulteriori prove per il confronto tra le nuvole ricavate dai diversi sistemi utilizzando la planarità e l'ortogonalità delle pareti. Tramite il software applicativo commerciale per la modellazione 3D di superfici e la gestione delle nuvole di punti chiamato Rhinoceros, da ogni ricostruzione vengono ricavati i punti appartenenti ad ognuna delle pareti e viene poi calcolato il piano di best fitting passante da essi ricavando così quattro piani che approssimano i muri della stanza. Il lato della stanza tra gli angoli 3 e 4 però non è una struttura solida, ma un cartellone ondulato, che, per problemi logistici, divide una stanza più grande in due sottostanze; perciò il lato descritto non verrà preso in considerazione.

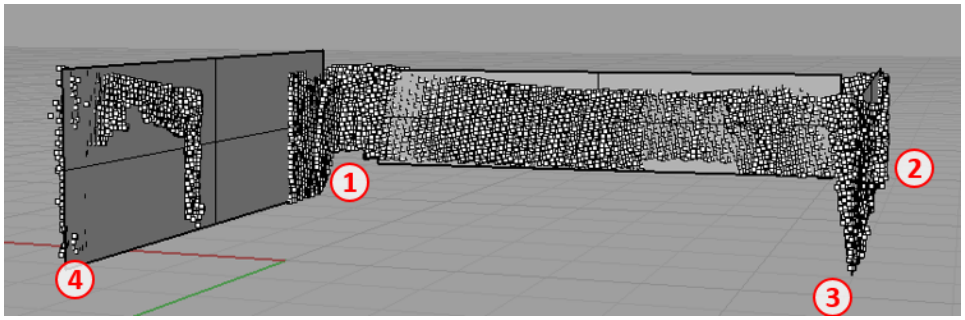


Figura 5.28: Calcolo dei piani di best fitting dei punti appartenenti al muro nella ricostruzione di RTAB-Map.

Si concentri inizialmente l'attenzione sul lato compreso fra gli angoli 1 e 2, poiché in esso, a causa dell'assenza di porte e finestre, si hanno più punti appartenenti al muro. Per ogni ricostruzione vengono calcolati gli scostamenti di ogni punto dal piano. Di queste poi si calcola la media, la mediana e la deviazione standard, le quali vengono riportate in tabella 5.8.

Inoltre si riportano in figura 5.30 le nuvole rappresentanti la ricostruzione del muro i cui punti sono colorati in base alla distanza tra essi ed il piano

	Media [mm]	Mediana [mm]	Dev. standard [mm]
RTAB-Map	11.6	10.1	7.6
Software Implementato	10.8	9.5	7.5
Stonex X300	3.0	2.6	2.3

Tabella 5.8: Risultati sulla planarità del lato tra gli angoli 1 *e* 2 *nelle ricostruzioni.*

interpolante; per una maggior visibilità sono stati aggiunti vettori proporzionali alla stessa distanza e ortogonali al piano.

Dai risultati si nota come il lato ricostruito con lo scanner Stonex X300 risulti molto più planare rispetto a quello ricostruito dagli altri due sistemi. I valori ricavati da RTAB-Map e il software implementato sono simili anche se il piano ricostruito dal secondo risulta leggermente più planare.

La deviazione standard così elevata nei due sistemi che utilizzano Kinect V2 è dovuta non solo dal processo di registrazione, ma anche dall'errore di misura del dispositivo. In figura 5.29 è possibile notare come i punti estremi di ogni nuvola registrata sporchino l'immagine complessiva, aumentando quindi la deviazione standard. Questo problema è rappresentato nell'immagine 5.9. L'errore della misura di profondità non è mediamente costante per ogni punto, ma cresce all'aumentare della distanza dal pixel centrale e dalla distanza del punto dal sensore. Nel paragrafo 5.2.2 è già stato descritto come in fase di registrazione questi punti non vengono considerati, ma sono visibili nella visualizzazione.

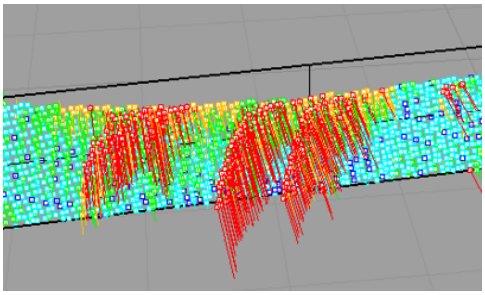


Figura 5.29: Distorsioni derivanti dai punti estremi di ogni nuvola di punti registrata.

Si passa ora allo studio sull'ortogonalità dei piani interpolanti in quanto si è verificato che i muri risultano essere ortogonali tra loro con ottima approssimazione. Una volta calcolati, per ogni ricostruzione, i tre piani che meglio approssimano i punti acquisiti che appartengono ai tre muri presi in considerazione, ne vengono calcolati gli angoli compresi.

Si definiscono θ_1 il primo angolo della scena acquisito (riferirsi alla figura 5.28), θ_2 il secondo e θ_{par} l'angolo tra le due pareti idealmente parallele. Per costruzione si deve avere $\theta_1 \simeq \theta_2 \simeq 90^o$ e $\theta_{par} \simeq 0$.

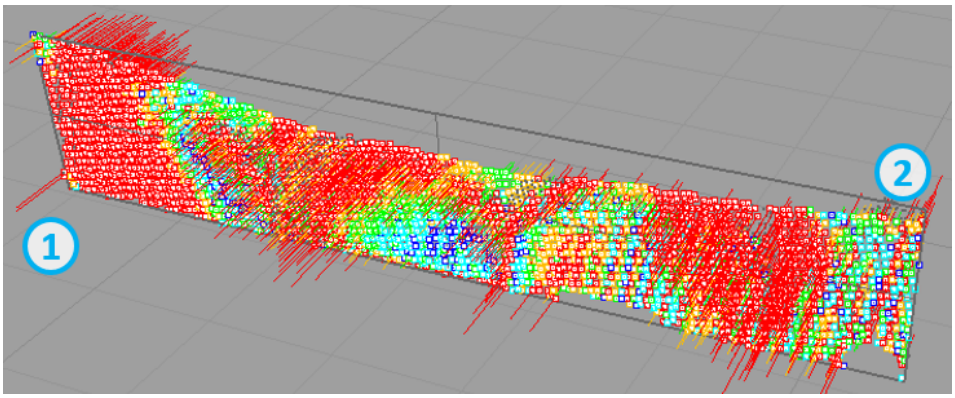
Prima di analizzare i risultati è possibile effettuare delle previsioni sul comportamento dei due sistemi di ricostruzione basati sul Kinect V2. Il software implementato non ottimizza il grafico tramite il loop closure, perciò l'errore nell'angolo θ_1 sarà maggiore di quello trovato con RTAB-Map. Per l'effetto già descritto in figura 5.26 invece si avrà un errore minore in θ_2 nella ricostruzione effettuata con il software implementato.

In figura 5.31 e in tabella 5.9 è possibile vedere i risultati ottenuti, che risultano essere coerenti con le previsioni e sostengono la scelta dell'utilizzo della ricostruzione effettuata con lo scanner Stonex X300 come riferimento.

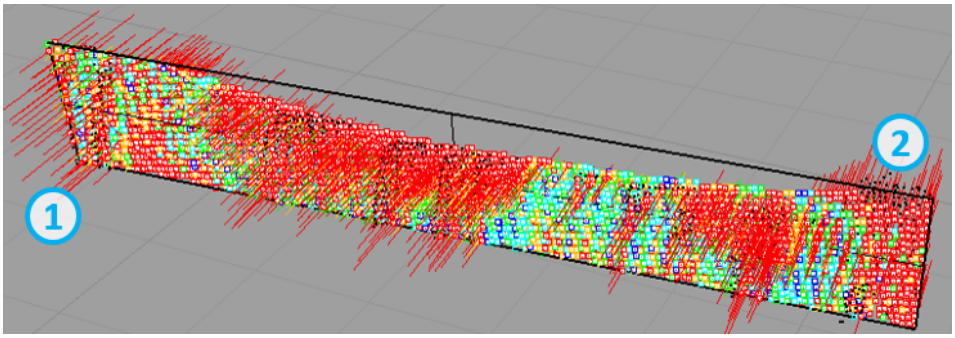
	θ_1 $[^o]$	θ_2 $[^o]$	θ_{par} $[^o]$
Stonex X300	90.0	90.1	0.1
RTAB-Map	91.5	86.9	1.5
Software Implementato	92.7	87.4	0.13

Tabella 5.9: Risultati sull'ortogonalità dei piani ricostruiti.

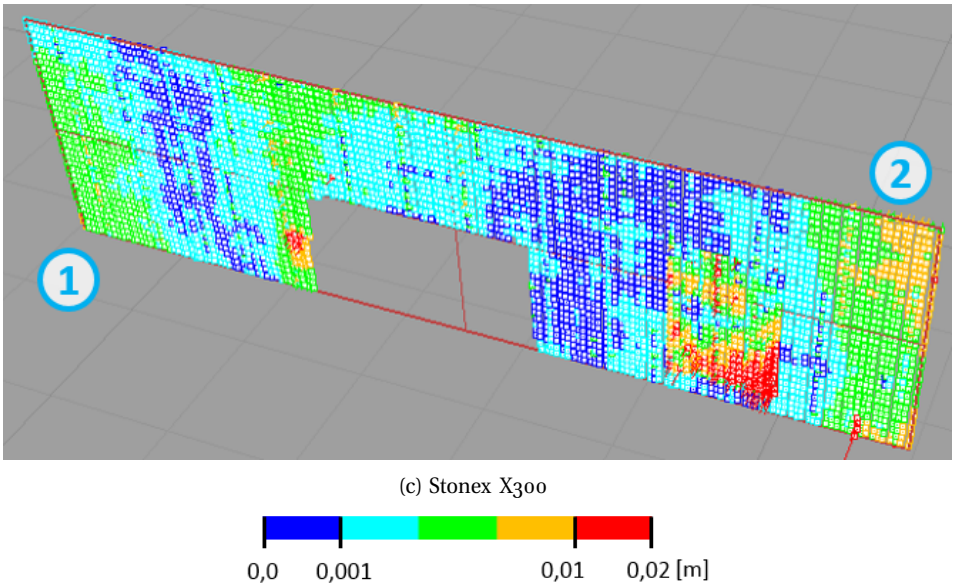
Anche se gli errori risultano essere di pochi gradi, le grandezze in questione possono portare ad uno scostamento di alcuni centimetri. Supponendo che il muro sia lungo 6 metri e sia presente un errore dell'angolo di 2^o, la fine del muro si discosterà di circa 20 cm dalla scena.



(a) RTAB-Map

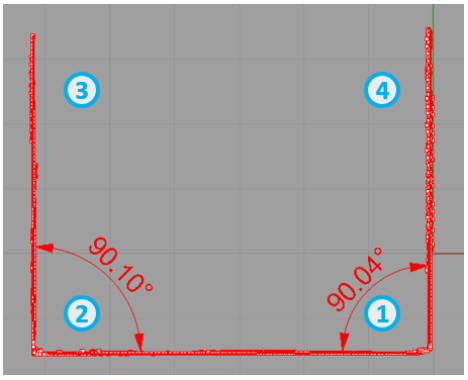


(b) Software Implementato

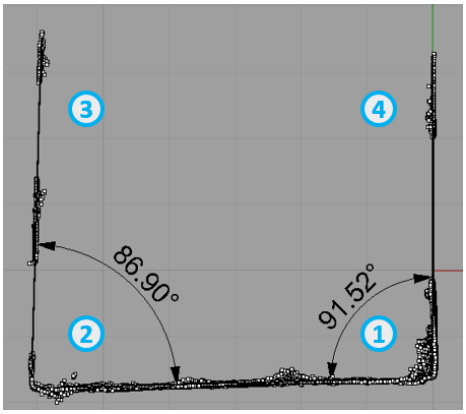


(c) Stonex X300

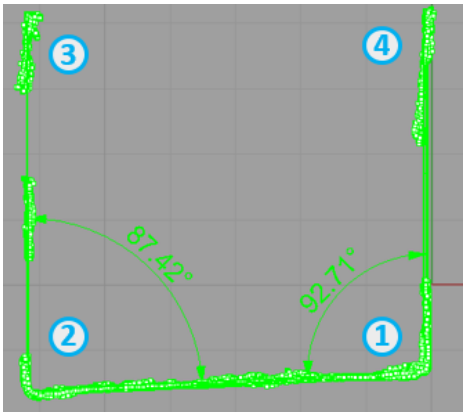
Figura 5.30: Risultati delle prove di planarità delle ricostruzioni del muro tra gli angoli 1 *e* 2.



(a) Stonex X300



(b) RtabMap



(c) Software Implementato

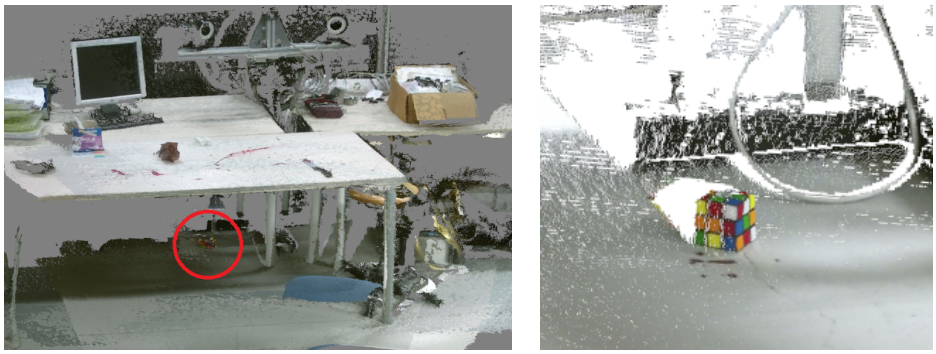
Figura 5.31: Risultati delle prove di ortogonalità.

5.5.2 Portabilità

Dopo aver confrontato le diverse tecniche sulla base della ricostruzione di una scena a 360^o si concentra l'attenzione su un'altra caratteristica che un dispositivo deve avere per una completa ricostruzione: la portabilità.

In ambito medico legale un particolare di una scena del crimine può essere determinante tra una condanna e un'assoluzione. Per questo è necessario che durante la ricostruzione tridimensionale della scena risulti possibile muovere il sensore nella stessa al fine di acquisire particolari importanti e di limitare le zone d'ombra.

Nella scena ricostruita sono stati posizionati due marker (cubi di Rubik) nella stanza, uno dietro l'antina di un armadio aperto e l'altro sotto un tavolo. Gli unici dispositivi in grado di acquisire queste aree sono stati il Sense e il Kinect V2. In figura 5.32 è visibile uno di questi marker acquisito con il Kinect V2.



(a) (b)

Figura 5.32: Posizione (a) e dettaglio (b) di un marker.

La ricostruzione della scena completa vista nel paragrafo 5.5.1, è stata effettuata limitando i gradi di libertà ad uno: la rotazione attorno all'asse perpendicolare al pavimento. L'acquisizione con Kinect V2 però può essere effettuata utilizzando tutti e 6 i gradi di libertà, con la possibilità di ruotare attorno agli assi e traslare la propria posizione. É possibile quindi avvicinarsi ad un oggetto in modo da ottenere una risoluzione maggiore in quel punto, come è visibile nella figura 5.32b.

Sia RTAB-Map che il software implementato riescono a ricostruire la scena anche quando l'utente ha a disposizione una mobilità di 6 gradi di libertà. La ricostruzione effettuata tramite quest'ultimo è visibile in figura 5.33.

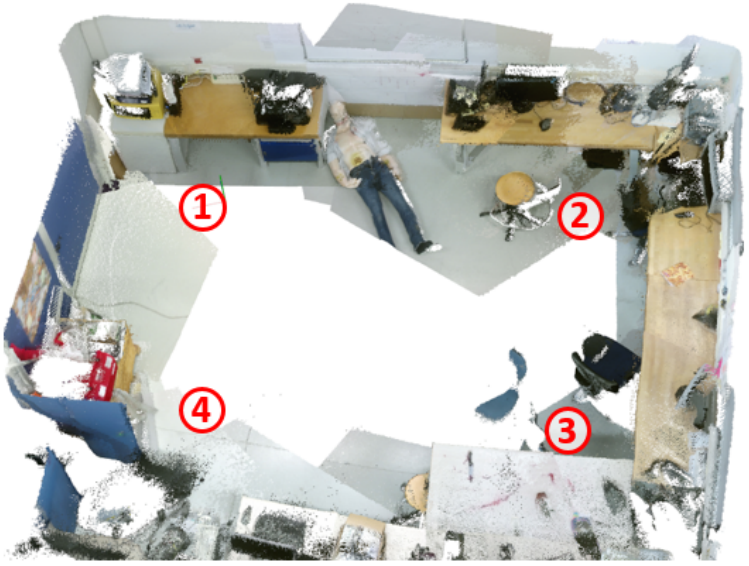


Figura 5.33: Ricostruzione tramite il software implementato utilizzando 6 gdl di movimento.

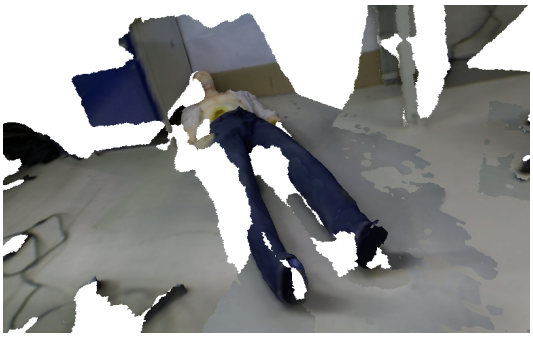
5.5.3 Dettaglio

In una scena del crimine alcune zone sono necessariamente più importanti di altre. In questa per esempio si concentra l'attenzione sulla zona in cui è situata la vittima.

In primo luogo si sono confrontati la capacità di acquisizione di un dettaglio tra i due dispositivi commerciali: il Kinect V2 e il Sense. La risoluzione della nuvola di punti dipende dalla distanza di acquisizione. Il Sense, come già visto durante la ricostruzione di Noceto, fornisce una mesh poco dettagliata in cui i colori sono sensibili alla luce e mediamente più scuri rispetto al visibile. Dalle acquisizioni eseguite con il Kinect invece, anche se la distanza di acquisizione non è ottimale, risulta possibile vedere le tracce di sangue sulla parete e sulla lavagna. Si capisce dunque perché, oltre al Sense, il laboratorio LABANOF ha bisogno di un ulteriore strumento per l'acquisizione di zone in cui è necessario un maggior dettaglio.

Il confronto fra il Minolta e lo Stonex risulta più difficile poiché non è stato possibile posizionare nello stesso punto i due dispositivi ed inoltre sono due strumenti complementari: il Minolta fa acquisizioni di piccole aree ad alta risoluzione mentre lo Stonex è dedicato ad acquisizione di scene ampie. Nelle nuvole di punti ricavate da questi dispositivi, in figura 5.35, è possibile notare come siano distinguibili anche le più piccole tracce di sangue sul muro. Per questo si può dire che entrambi i dispositivi sono in grado di percepire i dettagli importanti che si

possono trovare in una scena del crimine.



(a)

(b)

Figura 5.34: Dettaglio della scena ricostruito con il Sense (a) e con Kinect V2 ed il software implementato (b).

(a)

(b)

Figura 5.35: Dettaglio della scena ricostruito con lo Stonex X300 (a) e con il Minolta VIVID 910 (b).

5.5.4 Ricostruzione in presenza di scarsa illuminazione

La differenza principale di metodo tra RTAB-Map e il software implementato è che il primo si basa interamente su algoritmi di keypoints detection e descriptor 2D. Questi permettono una registrazione tra nuvole di punti colorate estremamente rapida poiché trovano corrispondenze tra i pixel dell'immagine i quali, come già spiegato, tramite la matrici di proiezione del modello pinhole di una telecamera e l'informazione di distanza ricavata dal sensore, vengono proiettati nello spazio 3D.

Si è provato a ripetere la prova descritta nella sezione 5.5.1 in condizione di bassa illuminazione. Come previsto, RTAB-Map, basando il suo algoritmo su rilevazioni di variazioni di intensità luminosa, non è riuscito a registrare le diverse nuvole di punti.

Dopo aver acquisito la stanza nelle stesse condizioni si sono utilizzati gli algoritmi di rilevazione e descrizione 3D, implementati nel software sviluppato durante la tesi, per ricostruire la scena. In particolare: per la rilevazione di keypoints viene utilizzato l'algoritmo SIFT 3D, poiché risulta aver maggiore ripetibilità rispetto agli altri keypoint detector 3D; nella fase di descrizione viene utilizzato l'algoritmo SHOT poiché risulta essere il più rapido tra i keypoint descriptors 3D. Anche se il tempo impiegato risulta essere decisamente troppo lungo, dovuto al carico computazionale per la rilevazione e descrizione 3D, la ricostruzione della scena, visibili in figura 5.36, risulta essere consistente con la scena stessa.

Vengono poi calcolati, dalla ricostruzione a bassa luminosità, effettuata dal software implementato, gli errori di propagazione tramite le formule presentate alla fine della sezione 5.5.1. Nelle tabelle 5.10 e 5.11 vengono riportati i risultati.

	$t_x\ [m]$	$t_y\ [m]$	$t_z\ [m]$	$d\ [m]$
Software Implementato	0.060	0.287	-0.146	0.327

Tabella 5.10: Risultati della stima dell'errore di traslazione nella ricostruzione in presenza di scarsa illuminazione.

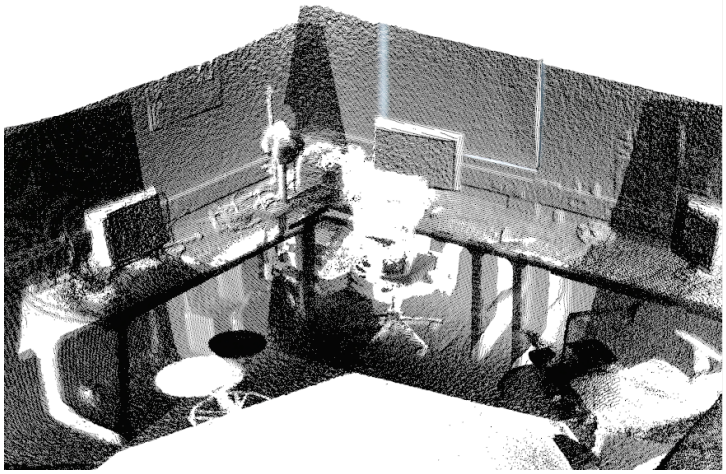
	$\theta_z\ [^o]$	$\theta_y\ [^o]$	$\theta_x\ [^o]$
Software Implementato	6.435	2.231	1.720

Tabella 5.11: Risultati della stima dell'errore di rotazione nella ricostruzione in presenza di scarsa illuminazione.

Si confrontano questi risultati con quelli ricavati nelle tabelle 5.6 e 5.7. Mentre gli errori di traslazione risultano comparabili, gli errori di rotazione nella ricostruzione effettuata in condizione di bassa luminosità dal software implementato, risultano essere maggiori di qualche grado. Il motivo della presenza di tale errore

è l'assenza dell'informazione colore; senza esso infatti la descrizione del punto risulta essere meno caratterizzante e la ricerca di corrispondenze tra le features diventa più difficoltosa.

Pur non riuscendo a raggiungere la stessa accuratezza di propagazione, il metodo sviluppato in questa tesi risulta essere il solo a poter ricostruire attraverso tecniche SLAM una scena in presenza di scarsa illuminazione, grazie all'utilizzo di rilevatori e descrittori 3D.



(a) Dettaglio del secondo angolo.

(b) Dettaglio del terzo angolo.

Figura 5.36: Ricostruzione della scena in condizione di bassa luminosità con il software implementato.

Capitolo 6

Conclusioni

In questa tesi è stata sviluppata una metodologia volta all'acquisizione e alla ricostruzione tridimensionale di ampie scene basandosi su tecniche SLAM a 6 gradi di libertà e implementando metodi di registrazione basati sulla rilevazione di keypoints in 2D e 3D e la loro descrizione in 3D.

A seguito di un'approfondita ricerca iniziale, durante la quale sono stati vagliati differenti dispositivi e diverse tecnologie attualmente disponibili sul mercato ed in grado di ricostruire scene tridimensionali, la scelta è ricaduta sul Kinect V2 prodotto da Microsoft. Tale scelta è stata favorita dal fatto che lo strumento è stato qualificato metrologicamente in un precedente lavoro svolto all'interno del VBLab [10] e quindi era noto che avesse prestazioni sufficienti per l'applicazione considerata in tesi. Il Kinect V2 integra una telecamera a tempo di volo, in grado di ricostruire la nuvola di punti 3D relativa alla porzione di scena inquadrata, alla frequenza di 30 Hz, con una risoluzione di 512×424 pixel. Per una maggior comprensione delle caratteristiche del dispositivo sono stati descritti i modelli matematici di una telecamera e di un dispositivo a tempo di volo ad onda continua.

In seguito viene descritto l'approccio della registrazione di scansioni multiple tipico della Simultaneous Localization And Mapping (SLAM). Dopo aver introdotto la formulazione probabilistica della tecnica, sono state presentate due strutture che ne permettono la descrizione: la rete bayesiana dinamica, che tratta il processo stocastico tramite un grafico orientato, e il Graph-Based, che ne sottolinea maggiormente la struttura spaziale. Il Graph-Based viene poi scomposto in due fasi: Front-End e Back-end. Il Front-End si occupa dell'allineamento tra le nuvole e della realizzazione di un grafo basato su legami temporali (nuvole di punti acquisite in istanti successivi) e spaziali (nuvole di punti che assumono posizione vicine). Il Back-End consiste nel determinare la miglior configurazione delle pose data la struttura ed i legami del grafo. Di quest'ultima fase viene proposta la formulazione del problema e una soluzione tramite il metodo denominato LUM.

La tesi continua focalizzando lo studio sulle tecniche di registrazione, necessarie per l'allineamento di due nuvole di punti. Per questo scopo, oltre alla registrazione manuale, vengono studiate ed analizzate due diverse tecniche: l'Iterative Closest Point (ICP) e la features based.

Per quanto riguarda la tecnica ICP sono state eseguite diverse prove finalizzate alla valutazione e alla comparazione dei suoi diversi algoritmi. In ogni caso, si ha a che fare con un processo iterativo; le differenze stanno nella definizione della funzione obiettivo da minimizzare. Si è valutata la bontà della soluzione in termini di distanze tra i punti di una nuvola rispetto a quelli della nuvola allineata e in termini di tempi computazionali. A parità di nuvole in ingresso ai diversi algoritmi, si è ottenuto un valore di distanza paragonabile ma tempi molto inferiori utilizzando l'algoritmo *PointToPlainLLS*, ritenuto quindi il migliore. A parità di algoritmo utilizzato, si è notato un valore di convergenza minore per tutte quelle nuvole rappresentanti scene senza geometrie schematizzabili con diedri concavi. Questo è dovuto a errori di riflessioni multiple caratteristici delle telecamere a tempo di volo. Tali prove hanno permesso di concludere, inoltre, che il tempo di registrazione dipende fortemente dalla posizione iniziale delle due nuvole. É consigliabile quindi un preallineamento grezzo e un successivo rifinimento tramite ICP.

La tecnica di registrazione feature based è scomponibile in due fasi principali. La prima fase, chiamata keypoint detection, rileva dei particolari punti di interesse nella scena, la seconda, chiamata keypoint description, li descrive basandosi sulle informazioni ricavate nel loro intorno. Tali fasi possono essere applicate sia in ambiente 2D, sfruttando immagini, che in ambiente 3D, sfruttando nuvole di punti. Diverse prove sono state effettuate per valutare le prestazioni nella fase di detection di keypoints 2D e 3D. Sulla base di immagini e nuvole acquisite tramite Kinect V2, sono valutati, per diversi algoritmi descritti in letteratura (quali per esempio SIFT, HARRIS, e BRISK), i tempi computazionali e soprattutto la ripetibilità nell'identificazione dei punti notevoli. Questa è definita come la capacità di ottenere lo stesso set di keypoints in istanze temporali differenti, è di fondamentale importanza nel processo di registrazione perché permette di stimare in maniera robusta le corrispondenze tra punti di una nuvola e l'altra. Si è compreso come l'errore immesso dal Kinect V2, dovuto all'incertezza della misura della distanza e alle riflessioni multiple, renda il processo di rilevazione di keypoints 3D meno ripetibili rispetto a quello 2D e come quest'ultimo sia computazionalmente molto più rapido.

Sono state poi effettuate prove per valutare il carico computazionale di diversi keypoint descriptors 2D e 3D. Si è tenuto conto non solo del tempo effettivo di descrizione ma anche del tempo impiegato durante il calcolo delle corrispondenze poiché esso dipende di descrittore utilizzato. Dai risultati è emerso come gli

algoritmi SHOT e C-SHOT siano computazionalmente più veloci rispetto agli altri.

Si è passati poi all'analisi dei software per la ricostruzione di scene già presenti sul mercato e all'implementazione di due moduli: il primo modulo è dedicato all'acquisizione ed al salvataggio di nuvole di punti ricavate dal sensore Kinect V2 e il secondo alla gestione delle nuvole di punti e alla ricostruzione della scena.

L'utilizzo di tali moduli viene presentato in dettaglio per due casi applicativi. Il primo riguarda la ricostruzione di un sito archeologico situato a Noceto (PA), dove è stata ritrovata una vasca costruita per scopi religiosi intorno al 1500 a.C. Far rivivere un reperto archeologico grazie alla fruizione del suo modello 3D virtuale è una richiesta sempre più frequente negli ultimi anni da parte di grandi musei e università di Archeologia.

Il secondo caso applicativo presentato, nasce dall'esigenza di valutare e confrontare diverse tecniche per la ricostruzione di una scena del crimine.

Lo scopo di queste ricostruzioni è quello di congelare la scena e la posizione degli oggetti presenti in essa in modo da poterla analizzare in assenza di inevitabili contaminazioni e soprattutto per poter tenere traccia dei dati che saranno poi necessari nello sviluppo delle indagini. Entrambe le attività sono state svolte in collaborazione con il LABANOF il Laboratorio facente parte della Sezione di Medicina Legale dell'Università degli Studi di Milano.

I dispositivi utilizzati per il confronto sono stati: il Sense, il laser scanner Minolta VIVID 910, e lo Stonex X300; inoltre le ricostruzioni sono state anche eseguite con il Kinect V2, sia mediante il software implementato, che con RTAB-Map, il software allo stato dell'arte più performante per la ricostruzione di ampie scene e vincitore del premio IROS2014 Microsoft Kinect Challenge. In tale software è presente una fase di Back-End che rileva un loop closure e di conseguenza ottimizza il grafo tenendo in considerazione il legame spaziale trovato. Mentre RTAB-Map utilizza unicamente rilevatori e descrittori 2D, il programma implementato impiega descrittori 3D, oltre che rilevatori 2D e 3D.

Dai test applicativi è emerso come i dispositivi Sense e Minolta non siano adatti per la ricostruzione di scene: il primo per lo scarso dettaglio e l'alta sensibilità alla variazione di luce; il secondo per il campo visivo limitato (900×1200 mm), che rende complessa l'acquisizione dell'intera scena. La possibilità di ruotare attorno ad un asse rende il dispositivo Stonex molto accurato in quanto nella sua ricostruzione non è presente l'errore di propagazione, proprio delle tecniche SLAM, dovuto alle successive registrazioni di nuvole di punti. Inoltre, tale ricostruzione è risultata essere la migliore nello studio sulle planarità e ortogonalità delle pareti della stanza ricostruita. Per questi motivi la ricostruzione effettuata con Stonex è stata presa come riferimento nelle prove sulla stima dell'errore di propagazione.

Dai risultati, si è potuto notare come la presenza di loop closure in RTAB-Map diminuisca l'errore di propagazione ma, ridistribuendo equamente tale errore

tra tutti gli allineamenti delle nuvole, distorca anche quelli che risultano corretti, restituendo così una nuvola di punti visivamente poco apprezzabile. Confrontando unicamente la parte Front-End di RTAB-Map con il software implementato, questo risulta avere un errore di propagazione minore.

Alla luce dei risultati il software implementato si è dimostrato idoneo per la ricostruzione di scene ampie, anche se, per la natura dei descrittori più lento di RTAB-Map. Inoltre il metodo sviluppato in questa tesi risulta essere il solo a poter ricostruire attraverso tecniche SLAM una scena in presenza di scarsa illuminazione, grazie all'utilizzo di rilevatori e descrittori 3D.

6.1 Sviluppi Futuri

Sebbene il sistema di ricostruzione sviluppato in questa tesi risulti essere più accurato e semplice da utilizzare di alcuni sistemi già in uso, e riesca a sostenere dignitosamente il confronto impegnativo con un software come RTAB-Map, sono necessari ulteriori studi per renderlo più prestante.

Nel corso di questa tesi l'attenzione è stata concentrata alla fase di Front-End, la quale si occupa principalmente della registrazione di nuvole di punti acquisite in successione. Nel software è però già stato implementato un modulo per la gestione della fase Back-End dello SLAM, che però non ha dato i risultati sperati. É necessario quindi uno studio approfondito sulle tecniche di ottimizzazione del grafo che non si limitino al LUM, ma anche ad algoritmi più recenti, come l'ELCH e il G2O.

La differenza fondamentale tra il software implementato e RTAB-Map risulta essere il tempo di ricostruzione della scena. L'utilizzo dei descrittori 3D rende necessariamente più lento il processo. Nel software implementato l'intero carico computazione viene svolto dalla CPU, è possibile però attraverso una tecnica chiamata *GPU computing* affiancare la GPU alla CPU nell'esecuzione di particolari istruzioni informatiche. Una GPU è capace di offrire una maggiore capacità di elaborazione rispetto a un normale processore di pari fascia di potenza in quanto le GPU si caratterizzano per un elevato grado di parallelizzazione della unità di elaborazione. La diversa architettura interna, infatti, permette di accorciare il tempo di esecuzione di istruzioni complesse, consentendo dunque di completare l'elaborazione delle informazioni in minor tempo. Ciò è possibile, però, solo nel caso in cui il codice del programma sia stato ottimizzato per l'esecuzione parallela anziché per l'esecuzione seriale delle istruzioni. In alcuni casi limite, con codice ottimizzato e scritto appositamente per la parallelizzazione delle istruzioni, le GPU sono in grado di garantire prestazioni 100 volte superiori rispetto a quelle delle CPU di pari potenza. Mentre RTAB-Map utilizza questo metodo, il software per la

ricostruzione creato in questa tesi utilizza unicamente la CPU. Accorciare di circa 50-100 volte i tempi di esecuzione renderà il software implementato sicuramente più performante.

Durante il corso di questa tesi si è speso molto tempo per riuscire ad elaborare un'architettura per il software che lo renda robusto e aiuti ad ottenere una miglior evolvibilità. Il software è pensato per essere un hub di moduli per la gestione di nuvole di punti. All'interno del VBLab, infatti, è attualmente in corso l'implementazione di un modulo per il riconoscimento di oggetti in una scena che verrà poi inserito nel software implementato durante il corso di questa tesi.

La tecnica di ricostruzione di ampie scene utilizzando i descrittori 3D potrebbe diventare la più utilizzata solamente fra qualche anno, quando i processori saranno più potenti e i dispendiosi calcoli per la descrizioni 3D potranno essere svolti in real time, e quando saranno presenti in commercio sensori di misura più accurati che garantiscano una incertezza di misura ridotta in tutti i punti della nuvola. Per essere più competitivi nel breve termine e per testare diverse tecniche di registrazione, può essere utile implementare nell'algoritmo anche la possibilità di utilizzare keypoint descriptors 2D.

Per quanto riguarda invece il sistema di acquisizione del Kinect V2, sia implementato che fornito da Microsoft, è possibile un miglioramento nell'accuratezza di misura. Dalle prove effettuate dal VBLab, descritte in 5.2.2, è riscontrabile un errore sistematico che aumenta in funzione della distanza dal pixel centrale e dalla distanza del punto dal sensore. Questo errore, essendo sistematico, può essere compensato. É dunque necessario uno studio allo scopo di modellare matematicamente l'errore sistematico in modo da eliminarlo durante le acquisizioni, o meglio ancora un modulo da aggiungere al software già implementato che dalla scansione di una superficie piana ricavi il piano di best fitting e tramite la distanza di ogni punto da esso calcoli i parametri della formula e compensi l'errore.

Bibliografia

[1] 3DVision. http://www.mecc.polimi.it/fileadmin/user_upload/File/Laboratori/TSLab_3DVision.pdf.

[2] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *Intelligent Transportation Systems Magazine, IEEE*, vol. 2, no. 4, pp. 31–43, 2010.

[3] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Robotics-DL tentative*, pp. 586–606, International Society for Optics and Photonics, 1992.

[4] PCL. http://pointclouds.org/.

[5] "Stonex x300." http://www.stonexpositioning.com/index.php/it/prodotti/laser-scanner/x300-detail.

[6] Sense. http://www.3dz.it/prodotti/sense-3d/.

[7] "Minolta vivid 910." http://www.konicaminolta.it/measuring-instruments/products/for-3d-measurement/non-contact-3d-digitizer/vi-910.html.

[8] RTAB-Map. http://introlab.github.io/rtabmap/.

[9] LABANOF. http://www.labanof.unimi.it/.

[10] A. Corti, S. Giancola, G. Mainetti, and R. Sala, "A metrological characterization of the kinect v2 time-of-flight camera," *Robotics and Autonomous Systems*, 2015.

[11] D. C. Brown, "Decentering distortion of lenses," *Photometric Engineering*, vol. 32, no. 3, pp. 444–462, 1966.

[12] L. Li, "Time-of-flight camera–an introduction," *Technical White Paper, May*, 2014.

[13] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous robot vehicles*, pp. 167–193, Springer, 1990.

[14] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *AAAI/IAAI*, pp. 593–598, 2002.

[15] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, pp. 206–211, IEEE, 2003.

[16] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *Robotics, IEEE Transactions on*, vol. 23, no. 1, pp. 34–46, 2007.

[17] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters for view-based slam," *Robotics, IEEE Transactions on*, vol. 22, no. 6, pp. 1100–1114, 2006.

[18] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[19] M.-L. Doaa, A. Mohammed, M. Salem, H. Ramadan, and M. I. Roushdy, "Comparison of optimization techniques for 3d graph-based slam," *Recent Advances in Information Science*, 2013.

[20] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular slam," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[21] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.

[22] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[23] D. Borrmann, J. Elseberg, K. Lingermann, A. Nuchter, and J. Hertzberg, "The efficient extension of globally consistent scan matching to 6 dof," *Knowledge Based Systems*, vol. 1, p. 20, 2008.

[24] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[25] RadiusOutlierRemoval. http://docs.pointclouds.org/trunk/classpcl_1_1_radius_outlier_removal.html#details.

[26] A. P. Witkin and P. S. Heckbert, "Using particles to sample and control implicit surfaces," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 269–277, ACM, 1994.

[27] J. S. Vitter, "Faster methods for random sampling," *Communications of the ACM*, vol. 27, no. 7, pp. 703–718, 1984.

[28] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 145–152, IEEE, 2001.

[29] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the icp algorithm," in *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pp. 260–267, IEEE, 2003.

[30] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[31] D. J. Meagher, *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer.* Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory, 1980.

[32] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[33] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 5, pp. 698–700, 1987.

[34] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pp. 2724–2729, IEEE, 1991.

[35] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, 2004.

[36] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp.," in *Robotics: Science and Systems*, vol. 2, 2009.

[37] M. Brown and D. G. Lowe, "Invariant features from interest point groups.," in *BMVC*, no. s 1, 2002.

[38] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1615–1630, 2005.

[39] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[40] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–511, IEEE, 2001.

[41] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1508–1515, IEEE, 2005.

[42] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *Computer Vision–ECCV 2010*, pp. 183–196, Springer, 2010.

[43] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[44] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2548–2555, IEEE, 2011.

[45] I. Sipiran and B. Bustos, "Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes," *The Visual Computer*, vol. 27, no. 11, pp. 963–976, 2011.

[46] A. Flint, A. Dick, and A. Van Den Hengel, "Thrift: Local 3d structure recognition," in *dicta*, pp. 182–188, IEEE, 2007.

[47] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3d object recognition," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 689–696, IEEE, 2009.

[48] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3d range scans taking into account object boundaries," in *Robotics and automation (icra), 2011 ieee international conference on*, pp. 2601–2608, IEEE, 2011.

[49] F. Stein and G. Medioni, "Structural indexing: Efficient 3-d object recognition," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 2, pp. 125–145, 1992.

[50] Y. Sun and M. A. Abidi, "Surface matching by 3d point's fingerprint," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, pp. 263–269, IEEE, 2001.

[51] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent point feature histograms for 3d point clouds," in *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, pp. 119–128, 2008.

[52] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 3212–3217, IEEE, 2009.

[53] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Computer Vision–ECCV 2010*, pp. 356–369, Springer, 2010.

[54] S. Filipe and L. A. Alexandre, "A comparative evaluation of 3d keypoint detectors in a rgb-d object dataset," in *9th International Conference on Computer Vision Theory and Applications. Lisbon, Portugal*, Citeseer, 2014.

[55] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in *Computer Vision–ECCV 2012*, pp. 759–773, Springer, 2012.

[56] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3d local feature descriptors," *International Journal of Computer Vision*, pp. 1–24, 2015.

[57] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pp. 127–136, IEEE, 2011.

[58] M. Labbé and F. Michaud, "Memory management for real-time appearance-based loop closure detection," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1271–1276, IEEE, 2011.

[59] M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *Robotics, IEEE Transactions on*, vol. 29, no. 3, pp. 734–745, 2013.

[60] A. CORTI, "Qualificazione metrologica di telecamera a tempo di volo microsoft kinect v2 per applicazioni in ambito neuroriabilitativo," 2015.