ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA
SEDE DI BOLOGNA
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Studio e sviluppo di descrittori locali per nuvole di punti basati su proprietà geometriche

TESI DI LAUREA IN

COMPUTER VISION AND IMAGE PROCESSING M

RELATORE:
PROF. Luigi Di Stefano

CANDIDATO:
Andrea Accordino

CORRELATORI:
DOTT. ING. Riccardo Spezialetti
DOTT. ING. Samuele Salti

III SESSIONE
ANNO ACCADEMICO 2017/2018

Ai miei genitori.
Sapevate che ci sarei riuscito, ma avrei voluto che lo vedeste.
Mi mancate.

Indice

Introduzione

La computer vision è la disciplina che studia e realizza sistemi in grado di emulare la percezione visiva umana sia dal punto di vista della rappresentazione, cioè la creazione di un modello che sia quanto più possibile simile a quello percepito nella scena osservata, che dal punto di vista della semantica, estraendo quindi dalla rappresentazione quante più informazioni possibili sulla natura degli oggetti che la compongono.

Gli scenari applicativi di questa disciplina sono innumerevoli, basti pensare al semplice fatto che un un sempre più grande numero di dispositivi elettronici di ultima generazione come smartphone, tablet e laptop, essendo dotati di una fotocamera sono grado di rispondere automaticamente ad uno stimolo esterno di tipo visivo.

Se ciò è ormai una realtà consolidata per il grande pubblico, lo è a maggior ragione per settori produttivi e di pubblica utilità, come: processi industriali, videosorveglianza, analisi del traffico, automotive e molti altri.

In uno scenario del genere, considerando inoltre l'aumento di capacità computazionale dei computer moderni, non ci si limita più ad elaborare immagini in due dimensioni. Infatti, da diversi anni ormai, esistono dispositivi in grado di acquisire scene tridimensionali che iniziano ad occupare una fascia di mercato sempre più consistente.

Uno di questi dispositivi noto a livello commerciale è Kinect (in figura 1), un accessorio distribuito assieme alla console video-ludica della Microsoft *Xbox 360* che consente un'esperienza di gioco in cui la funzione del controller fisico è sostituita dalla mappatura 3D del giocatore e dei suoi movimenti.

Figura 1: Kinect: un accessorio della Xbox 360 in grado di acquisire scene tridimensionali utilizzate per far interagire un utente con un videogioco senza l'uso di un terminale fisico

Altri dispositivi meno noti nel grande mercato sono, però, indispensabili nel settore healthcare ed in particolare nella diagnostica per immagini: TAC, risonanze magnetiche e molti altri esami necessitano una ricostruzione quanto più dettagliata delle superfici tridimensionali degli organi interni del corpo umano.

Esattamente come avviene nelle immagini in due dimensioni, esistono delle applicazioni della computer vision incentrate sulle superfici tridimensionali:

- Registrazione: sovrapposizione di visioni parziali (dette 2.5D) in un unico modello full 3D (vedi fig. 2)

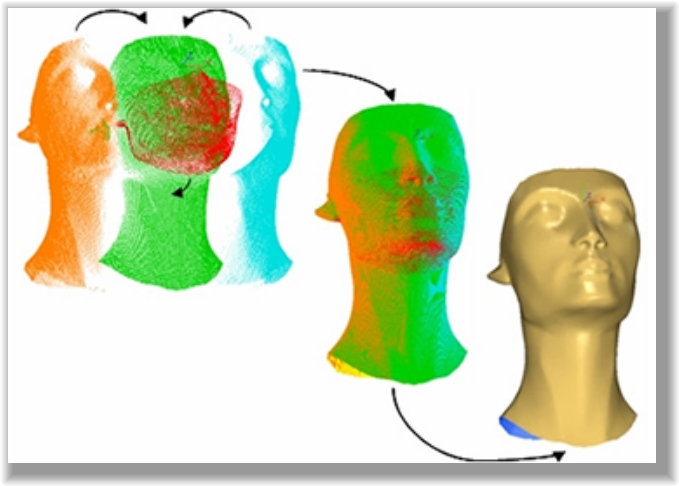

Figura 2: Esempio di registrazione di due superfici 3D parziali

- SLAM: costruzione incrementale una mappa dell'ambiente circostante (vedi fig. 3)

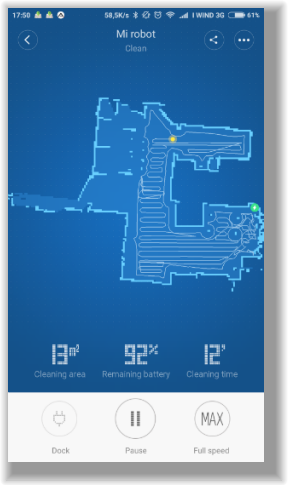

Figura 3: L'app di un robot aspirapolvere che permette di visualizzare la mappa di un appartamento ricostruita con algoritmi di SLAM

- Retrieval: riconoscimento una superficie come istanza di una determinata classe di oggetti
- Recognition: individuazione della presenza o meno di un oggetto all'interno di una scena a prescindere dalla sua posizione e rotazione (vedi fig. 4)

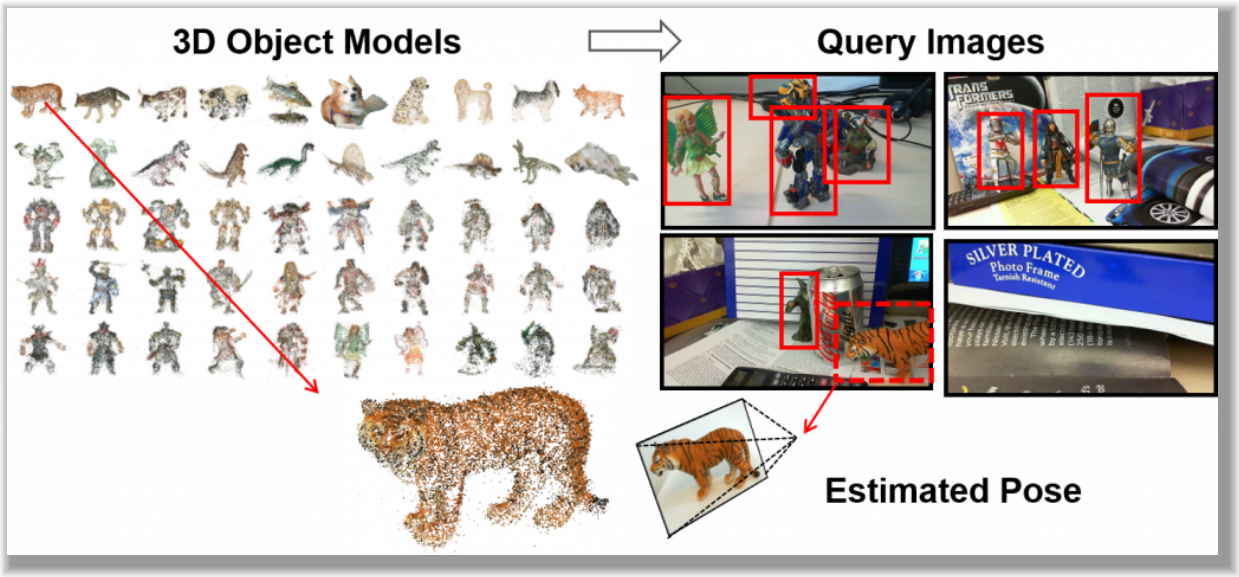

Figura 4: Esempio di 3D recognition

- Segmentazione semantica: estrazione di parti di una scena in base a determinate proprietà semantiche

Per ciascuno di questi task è necessaria una fase di estrazione di informazioni dalla scena in grado di collegare delle superfici esistenti al suo interno con dei modelli già classificati e presenti in memoria. Queste informazioni sono legate a dei punti della superficie ed in particolare alla forma della stessa attorno a quel punto. È necessario quindi quantificare queste informazioni e inserirle in un'apposita struttura dati detta *descrittore*.

Questo lavoro di tesi ruota attorno a questo strumento, ne testeremo alcuni preesistenti e li compareremo ad alcuni nuovi che saranno qui proposti ed implementati. I test comparativi riguardano le prestazioni in termini di efficacia e tempo di calcolo.

Quest'ultimo indicatore è di cruciale importanza sia perché la computazione di un descrittore va iterata per un grande numero di punti sia perché un algoritmo snello trova applicazioni anche tra i dispositivi mobili che, com'è noto, dispongono di una capacità limitata sia dal punto di vista energetico che computazionale.

Introduciamo brevemente il contenuto di questo lavoro di tesi.

Il primo capitolo contiene una parte introduttiva sulla computer vision. Considereremo i limiti della rappresentazione bidimensionale e i mezzi utilizzati per superarli: sensori, rappresentazione dei dati in 3D ed uno sguardo alla pipiline del feature matching, necessaria per qualunque tipo di risvolto pratico su questo tema.

Il secondo capitolo entra nel dettaglio di una delle fasi del feature matching: i descrittori. Dopo averne stabilito definizione, utilità e alcune classificazioni si passerà ad illustrare i principi di funzionamento di tre di essi utilizzati nei test.

Nel terzo capitolo saranno illustrati dei nuovi metodi, proposti per migliorare le performance dell'intera pipeline di feature matching.

Gli strumenti utilizzati per l'implementazione e la valutazione dei nuovi metodi rispetto alle versioni originali, saranno descritti nel quarto capitolo.

Infine, il quinto capitolo mostrerà i risultati sperimentali ottenuti dai test effettuati.

Capitolo 1

Computer Vision

Al fine di poter rappresentare ed elaborare informazioni riguardanti oggetti e, più in generale, superfici che giacciono in uno spazio tridimensionale è necessario disporre di uno strumento in grado di catturare informazioni sulla profondità di un determinato punto sulla scena.

Nella sua approssimazione pinhole (vedi schema in fig. 1.1), ovvero in cui l'obiettivo è costituito da un foro di dimensioni infinitesimali, approssimabile ad un punto, la fotocamera cattura un'immagine bidimensionale. Se immaginiamo il segmento che va da un punto del piano focale all'obiettivo, infatti, ci rendiamo conto che tutti i punti della scena che giacciono su quella retta verranno rappresentati sullo stesso punto nel piano focale e quindi nell'immagine.

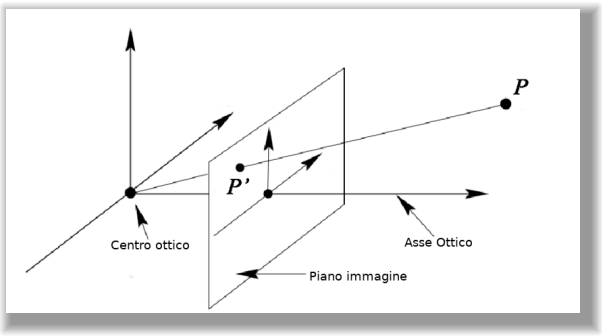

Figura 1.1: Approssimazione pinhole di una fotocamera tradizionale

Questo fatto fa perdere qualunque informazione sulla profondità della scena e ciò è inevitabile dal momento che questa tecnica prevede che la rappresentazione della scena perda una dimensione per essere catturata e visualizzata su superfici bidimensionali come uno schermo o una fotografia.

Un modo di aggirare questo problema è quello di ricavare le informazioni sulla profondità ricorrendo a più esposizioni della stessa scena da diversi punti di vista, ma come vedremo ce ne sono anche altri.

1.1 Acquisizione dei dati

Negli ultimi anni sono stati inventati dei sensori che fanno uso di diverse tecniche di computer vision: essi possono essere suddivisi in due principali categorie, quelle dei sensori **attivi** e dei sensori **passivi**.

I sensori attivi ricavano le informazioni sulla profondità dei punti presenti all'interno della scena tramite l'analisi di un segnale emesso dal sensore, in questa categoria ricadono:

- LIDAR: un dispositivo emette un impulso luminoso, l'informazione di profondità è calcolata in base al round-trip time, cioè al tempo che impiega il fascio a colpire l'oggetto e a ritornare al sensore. (fig. 1.2)

Figura 1.2: Lidar Velodyne VLS-128

- Sensore Time-of-Flight: analogamente al precedente, genera un impulso luminoso ma l'informazione della profondità è calcolata in base al decremento di potenza subito lungo il tragitto.

Figura 1.3: Time-on-Flight Camera MESA SwissRanger 4000

- Sensore a Triangolazione laser: viene emesso uno o una serie di impulsi laser direzionati in modo da formare un angolo noto con l'asse ottico del sensore in ricezione. La distanza della superficie target dal sensore è calcolata in base alle regole di trigonometria. (fig. 1.4)

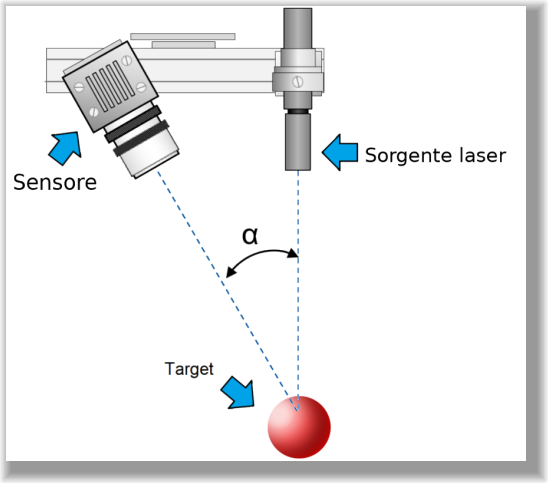

Figura 1.4: Schema di un sistema a triangolazione laser

- Sensore Structured Light: si proietta un pattern 2D noto, una fotocamera ricostruisce la profondità dalla distorsione prospettica applicata al pattern. (fig. 1.5)

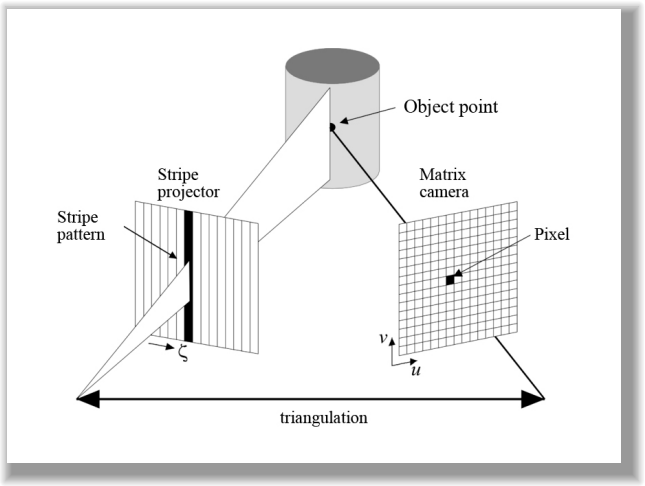

Figura 1.5: Schema di un sensore structured light

I sensori passivi, invece ricavano le informazioni sulla profondità dei punti presenti all'interno della scena tramite l'acquisizione di immagini bidimensionali da diverse angolazioni, in questo caso i sensori sono comuni fotocamere i cui dati sono elaborati secondo alcune tecniche di computer vision:

- Visione Stereo: una coppia di fotocamere a distanza nota cattura la scena, un algoritmo riconosce i punti corrispondenti nelle due acquisizioni ricostruendo le informazioni sulla profondità con la triangolazione. (fig. 1.6)

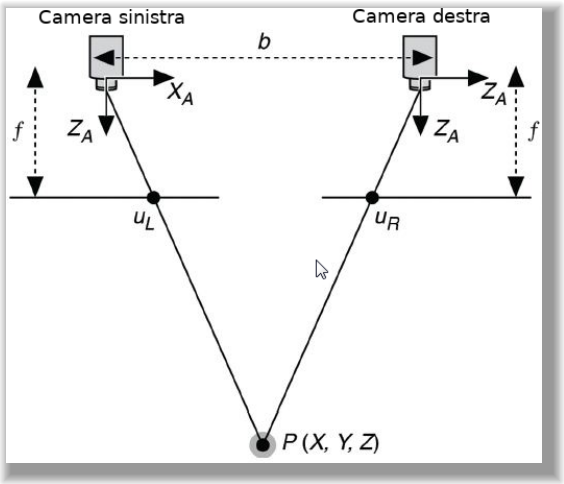

Figura 1.6: Principio della visione stereo

- Spacetime stereo: si tratta di una variante della tecnica di visione stereo. Prevede la proiezione di un pattern casuale sulla scena e l'aggiunta di informazioni catturando, nel tempo, più viste della stessa.
- Strutture from motion: un'unica camera si muove attorno all'oggetto e si ricorre ad un algoritmo che ricava le informazioni sulla profondità dallo spostamento relativo dei punti delle superfici inquadrate. (fig. 1.7)

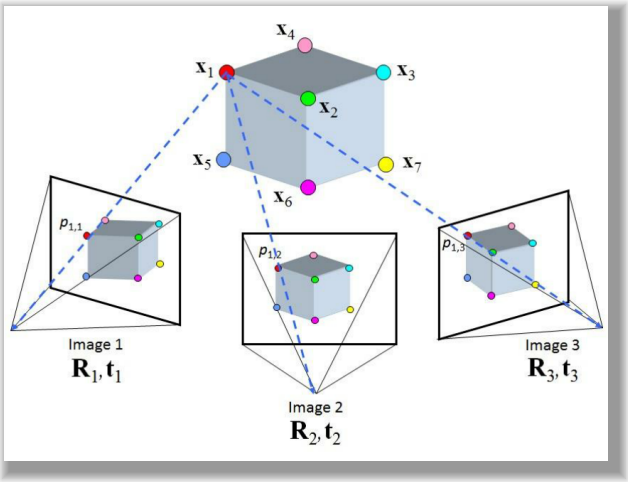

Figura 1.7: Schema di un sistema structure from motion

1.2 Rappresentazione dei dati

I dati acquisiti dai sensori appena descritti, necessitano di essere memorizzati in opportune strutture dati al fine di poter essere gestiti ed elaborati. Esistono due categorie di strutture dati: quelle **organizzate** e quelle **non organizzate**. Nella prima categoria ricade, ad esempio, la rappresentazione **range image** (fig. 1.8), una matrice bidimensionale (interpretabile anche come un'immagine) in cui il livello di grigio ogni pixel rappresenta la distanza del punto della scena dal sensore con il quale è stata acquisita.

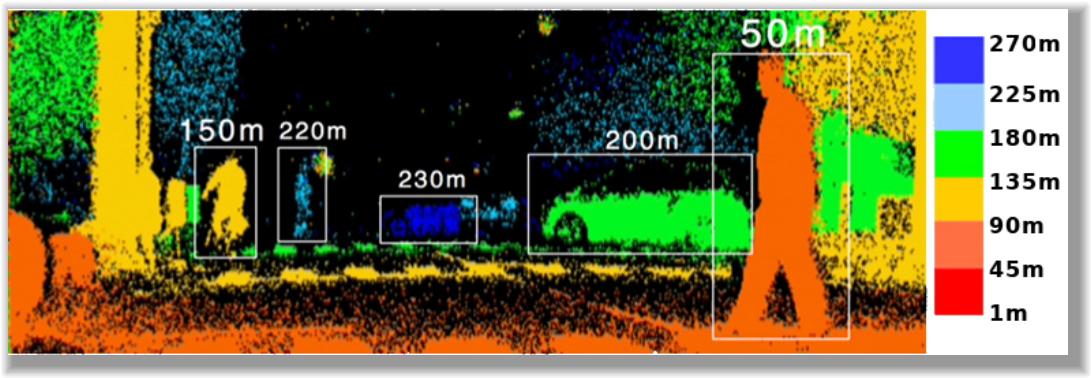

Figura 1.8: Esempio di range image

Nella seconda categoria troviamo invece:

- Point cloud, una lista di coordinate di punti 3D senza alcuna informazione topologica.
- Mesh poligonale, una collezione di vertici, spigoli e facce (tipicamente triangoli) che vanno a definire la superficie di un oggetto.

In figura 1.9 un confronto tra il render di una superficie rappresentata con le 2 strutture appena descritte.

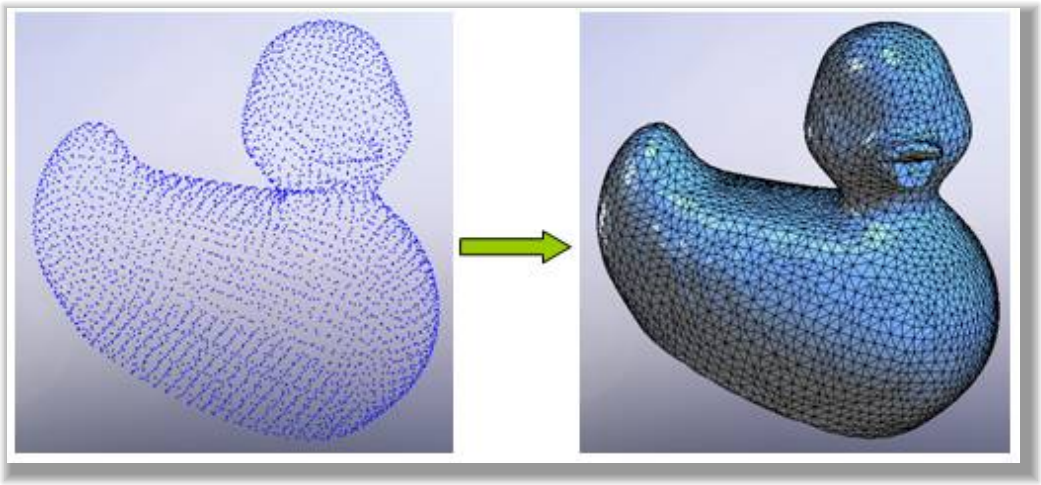

Figura 1.9: Comparazione tra nuvola di punti (a sinistra) e mesh poligonale (a destra)

Inoltre, alle singole coordinate è possibile affiancare anche informazioni sul colore in formato RGB o RGBA ma nei test realizzati sui nostri descrittori, seppur presenti in alcune scene, queste informazioni non sono state utilizzate.

La rappresentazione più utilizzata è la point cloud, essa offre alcuni vantaggi come una minore perdita di informazioni dovuta alla quantizzazione e un consumo ridotto di memoria nel caso di oggetti con grandi differenze di densità. Ma anche degli svantaggi come un alto costo computazionale per la ricerca dei vicini di un punto e la difficoltà nel discriminare tra parti interne ed esterne della superficie a causa dell'assenza di informazioni topologiche

1.3 Registrazione

Ciascuno dei sensori introdotti fin'ora fa uso della luce riflessa per la cattura delle informazioni. Ciò implica che ci ritroviamo di fronte al limite posto dal loro posizionamento, a meno che esso non circondi fisicamente tale oggetto (il che è il più delle volte di difficile attuazione).

Questo limite, dato dal *field of view*, comporta la necessità di effettuare diverse acquisizioni dette *viste 2.5D*, per poter avere una rappresentazione intera della scena tridimensionale detta invece *full 3D*.

La ricostruzione di un modello 3D completo a partire da acquisizioni della stessa scena da più viste 2.5D è detta *registrazione*. (vedi fig. 1.10)

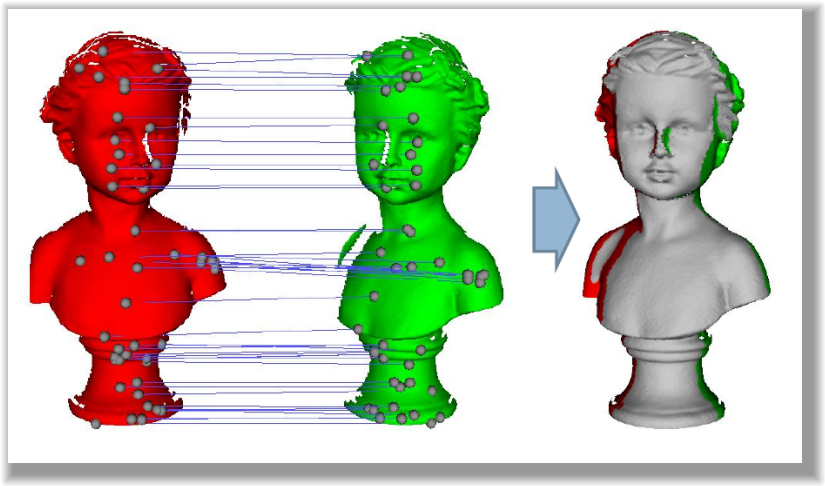

Figura 1.10: Esempio di registrazione di due viste 2.5D

La registrazione unisce i punti provenienti da più viste in un unico sistema di riferimento coerente con la scena. A questo scopo è necessario individuare un numero minimo di corrispondenze, cioè, punti di due o più viste 2.5D che corrispondono allo stesso punto nella scena.

1.4 Feature matching

La ricerca di corrispondenze tra più viste 2.5D è detta "Feature matching" ed è l'operazione preliminare alla registrazione. Essa è generalmente suddivisa in una pipiline di tre fasi:

1. **Keypoint detecting**: in questa fase vengono ricercati alcuni punti, appartenenti alla superficie, che possiedano spiccate caratteristiche distintive generalmente basate su informazioni ricavate dai punti vicini, questi punti sono definiti *keypoint*.
2. **Keypoint description**: in questa fase viene attribuito ad ogni keypoint una struttura dati detta *descrittore* e, al suo interno, inseriti dei valori in grado di rappresentare quantitativamente le caratteristiche rilevate.

3. **Descriptor matching**: in quest'ultima fase si ricercano le corrispondenze tra i keypoint di viste diverse in base alla somiglianza dei loro descrittori. Possono essere utilizzate diverse metriche, la più popolare è la distanza euclidea nello spazio dei descrittori.

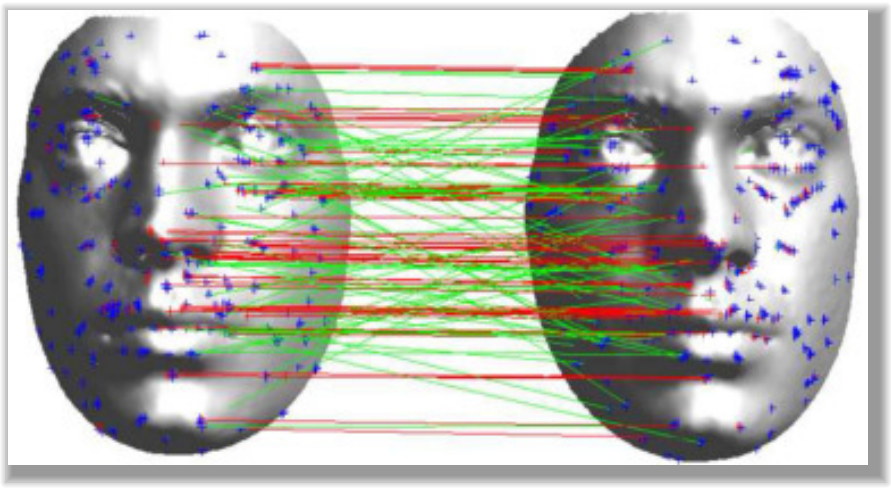

Figura 1.11: Corrispondenze tra i keypoint di due scene

La prima fase è svolta da alcuni algoritmi come Harris3D [1] e KPL [2]. Nel nostro caso, comunque, ci siamo concentrati sullo studio delle migliori proprietà geometriche per un descrittore e abbiamo scelto di isolare le sue performance da quelle del detector. Questo perché in letteratura è stato ampiamente studiato il rapporto tra detector e descrittori [3]. Quindi abbiamo adottato altri metodi di selezione dei keypoint che vedremo in seguito.

Per quanto riguarda l'ultima delle tre fasi, si esegue un confronto tra i tutti i descrittori di tutte le coppie di viste possibili appartenenti alla scena. Essendo il descrittore interpretabile come un vettore a n dimensioni (con n numero di valori della sua la struttura dati), se ne calcola la distanza che, tanto più sarà bassa, tanto più saranno simili i relativi keypoint. La ricerca di vicini tra vettori a molte dimensioni è computazionalmente onerosa, per aggirare questo collo di bottiglia si utilizzano i k-d-tree. Il K-dimensional-tree è un albero binario di ricerca multidimensionale in cui ogni nodo indicizza un punto in k dimensioni. Inoltre, come vedremo, questo metodo può essere utilizzato anche nello spazio delle coordinate per la formazione del supporto.

Per la fase di keypoint descriptor, essendo l'argomento centrale di questo lavoro di tesi, si rimanda la lettura al capitolo seguente.

Capitolo 2

Descrittori locali

Il descrittore è una rappresentazione compatta e distintiva di un punto su una superficie tridimensionale. Quando un descrittore è calcolato sulla base delle informazioni estratte dai punti che giacciono nell'intorno euclideo del suo keypoint, è detto *locale*, altrimenti si dice *globale* [4]. Nel primo caso i punti dell'intorno sono detti *vicini* e il sottoinsieme di punti da essi formato è detto *supporto*.

Il supporto di un punto può essere costituito dai suoi primi k punti in ordine crescente di distanza o, indipendentemente dalla densità, da tutti quelli che cadono all'interno di una sfera di raggio r dal keypoint.

Per le immagini esistono descrittori 2D, come SIFT [5] e SURF [6] che interpretano la griglia dei pixel di cui l'immagine è costituita come funzione a due dimensioni, e si basano principalmente su operatori di tipo differenziale su tali funzioni. Esse utilizzano dei pixel posti fino ad una distanza k dal keypoint (k-vicinato).

In strutture dati non organizzate come le point cloud, abbiamo una limitazione data dall'alta complessità computazionale nella ricerca di un vicino. Per far fronte a questa complicazione si utilizza, come per la ricerca dei vicini tra descrittori nella feature matching, il k-d-tree.

2.1 Classificazione dei descrittori

Nel corso del tempo sono state fornite diverse classificazioni per i descrittori allo stato dell'arte. Noi ne useremo due, la prima [7] ci permette di distinguere tra:

- Descrittori con sistema di riferimento locale: i descrittori di questo tipo necessitano di una fase preliminare in cui è necessario calcolare un sistema di riferimento per ogni keypoint e relativo supporto.
- Descrittori senza sistema di riferimento locale: in questo caso, si sceglie una feature rotation invariant che non necessita, quindi, di tale sistema.

La seconda classificazione divide i descrittori in:

- Descrittori ad attributi geometrici: in questa categoria le proprietà utilizzate per la costruzione del descrittore sono dette *feature* e si basano su angoli e distanze tra i punti del supporto.
- Descrittori a densità di punti: qui invece, le proprietà utilizzate per la costruzione del descrittore si basano sulla quantità di punti contenuti nel supporto.

Gli angoli considerati nei descrittori ad attributi geometrici sono generalmente ricavati considerando i punti p_i del supporto considerati orientati secondo la direzione del versore $\hat{n}_i$ normale al piano tangente alla superficie per p_i. Questi versori sono inseriti in un'altra struttura dati avente, quindi, la stessa cardinalità di quella contenente la point cloud.

Prima di illustrare i descrittori utilizzati in questo progetto di tesi, al fine di utilizzare una notazione omogenea definiamo alcune quantità:

- k è il numero punti contenuti nel supporto ad eccezione del keypoint
- p_0 è il keypoint su cui calcolare il descrittore
- $p_i \quad (i = 1, ..., k)$ l'i-esimo vicino nel supporto
- $\hat{n}_i$ è il versore normale al piano tangente di p_i
- $\hat{x}, \hat{y}, \hat{z}$ sono i tre assi del sistema di riferimento locale, qualora richiesti.

2.2 Descrittori utilizzati

2.2.1 Fast Point Feature Histogram

FPFH è un'evoluzione di PFH [8], si tratta di un descrittore ad attributi geometrici con sistema di riferimento locale. Si applica a point cloud contenenti esclusivamente coordinate ma esiste un'implementazione, PFHRGB, che opera analogamente anche nello spazio dei colori RGB.

Il descrittore PFH considera tutte le coppie di punti p_i e p_j presenti nel supporto creando una topologia fully connected come mostrato in figura 2.1.

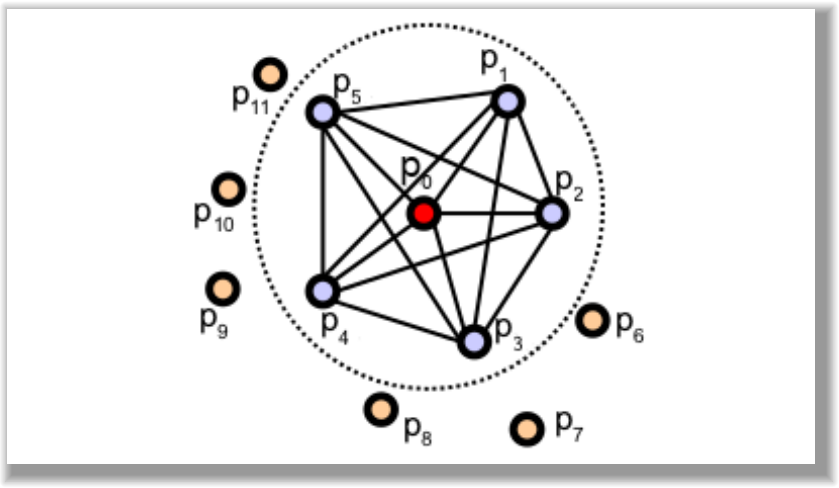

Figura 2.1: Diagramma di un supporto in PFH con $k = 5$

Per ciascuna coppia così considerata, calcola il sistema di riferimento locale illustrato in figura 2.2:

$$\hat{x} = \hat{n}_i$$
$$\hat{y} = \hat{x} \times \frac{(p_i - p_j)}{\|p_i - p_j\|_2}$$
$$\hat{z} = \hat{x} \times \hat{y}$$

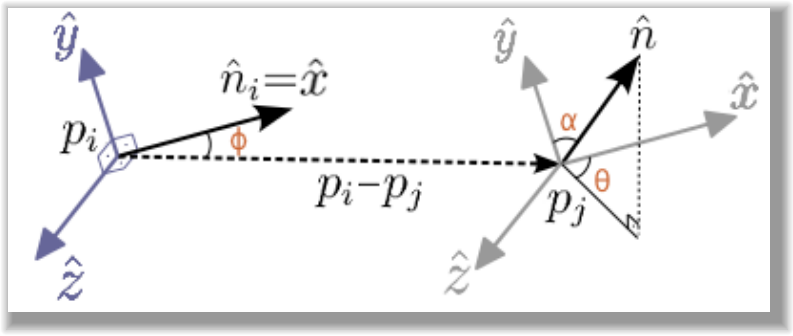

Figura 2.2: Schema del sistema di riferimento tra una coppia di punti in PFH

Utilizzando il frame appena definito, vengono calcolate le feature che consistono nella differenze tra le normali $\hat{n}_i$ e $\hat{n}_j$, questa quantità è rappresentata come tripla di angoli α, ϕ, θ:

$$\alpha = \hat{v} \cdot \hat{n}_j$$

$$\phi = \frac{p_i - p_j}{d}$$

$$\theta = arctan(\hat{z} \cdot \hat{n}_j, \hat{u} \cdot n_j)$$

ed opzionalmente la distanza euclidea $d = \|p_i - p_j\|_2$.[1]

Questi 3 valori vengono inseriti in un istogramma partizionato in 5 intervalli. La struttura dati che contiene l'istogramma così ottenuto contiene, dunque, $5^3 = 125$ valori in virgola mobile per ciascun punto.

L'algoritmo appena descritto, considerando tutte le coppie all'interno del supporto, ha una bassa efficienza computazionale per unità di punti $O(k^2)$, ciò ha spinto a creare una versione più efficiente, ovvero FPFH.

FPFH infatti riduce la complessità computazionale di PFH a $O(k)$ calcolando α, ϕ, θ non più per ogni coppia di punti nel supporto ma solo tra p_0 e p_i per ognuno dei k vicini, questa operazione è chiamata $SPFH(p_0)$.

[1] `http://pointclouds.org/documentation/tutorials/pfh_estimation.php`

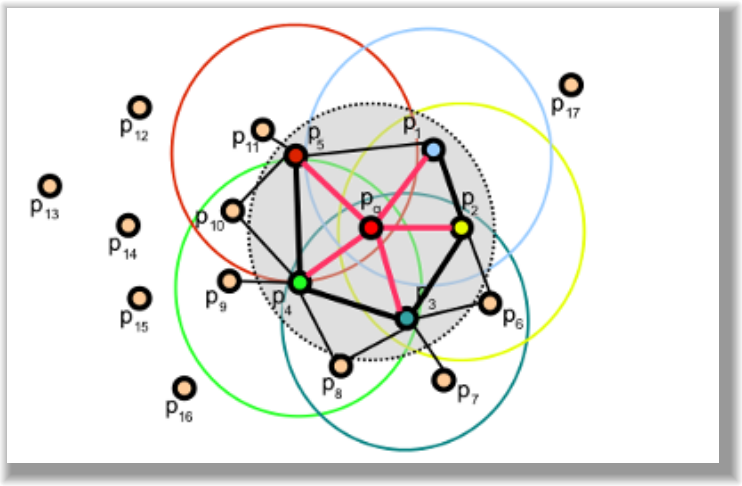

Figura 2.3: Diagramma di un supporto in FPFH con $k = 5$, i cerchi colorati corrispondono alle regioni di influenza dell'i-esimo vicino

Il descrittore finale viene calcolato come media pesata di $SPFH(p_i)$ per ognuno dei k vicini secondo la seguente formula:

$$FPFH(p_0) = SPFH(p_0) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_i}SPFH(p_i)$$

In cui w_i è una funzione della distanza tra p_0 e p_i definibile dall'utente.

Il numero di partizioni dell'istogramma è variabile, l'implementazione PCL di questo descrittore utilizza 11 intervalli di valori, ottenendo così una struttura dati di $3\times11 = 33$ valori in virgola mobile per keypoint.

2.2.2 Signature of Histograms of OrienTations

SHOT è un descrittore ad attributi geometrici con sistema di riferimento locale. Può essere utilizzato nelle point cloud contenenti esclusivamente coordinate, ma ha anche un'estensione per nuvole di punti con informazioni sul colore RGB: CSHOT.

Gli autori di SHOT classificano i descrittori preesistenti in due categorie (vedi fig. 2.4):

- signature: in questa categoria rientrano i primi descrittori in cui le feature sono raccolte ed ordinate in strutture dati spazialmente ben localizzate. Ciò conferisce caratteristiche di unicità ad ogni keypoint ma poca fault tollerance a variazioni anche piccole del Local Reference Frame.
- histogram: in questa categoria rientrano i lavori successivi, i valori sono accumulati in partizioni di un istogramma associate ad un intervallo minimo e massimo. Questa scelta, sebbene comporti una perdita di potere descrittivo, aumenta la tolleranza a piccole variazioni dell'LRF.

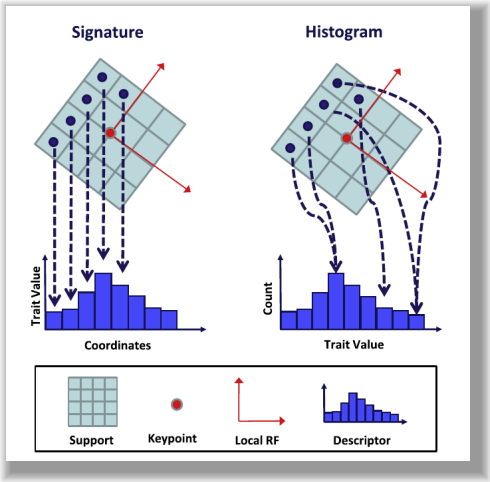

Figura 2.4: Schema esemplificativo della differenza tra descrittori a signature e ad histogram

SHOT viene classificato come un ibrido delle due categorie visto che benché sia utilizzato un istogramma come struttura dati, la scelta a monte di un LRF unico e non ambiguo gli conferisce caratteristiche di unicità tipiche della signature.

Il sistema di riferimento locale di SHOT è calcolato utilizzando la matrice di covarianza M definita sulle coordinate dei punti appartenenti al supporto del keypoint p_0. Per motivi di efficienza computazionale si evita il calcolo delle coordinate del centroide del supporto, utilizzando al loro posto quelle di p_0. L'i-esimo punto del supporto contribuisce alla formazione di questa matrice in modo inversamente proporzionalmente alla sua distanza euclidea $d_i = \|(p_0 - p_i)\|_2$:

$$M = \frac{1}{k}\sum_{i=1}^{k}(R - d_i)(p_0 - p_i)(p_0 - p_i)^T$$

Gli assi del sistema di riferimento vengono individuati mediante Singular Value Decomposition (SVD) o Eigen-value Decomposition (EVD) della matrice M. Dagli autovettori si ricava il LRF: la direzione dell'asse x sarà data dall'autovettore corrispondente all'autovalore maggiore, mentre per l'asse z si usa quello dall'autovalore minore.

Il verso dato dall'output della decomposizione, invece, è casuale, perciò, al fine di avere un sistema di riferimento locale unico e ripetibile gli assi vengono disambiguati [9]. Consideriamo x^+, y^+, z^+ gli autovettori x^-, y^-, z^- i loro rispettivi opposti, per decidere il verso creiamo due insiemi:

$$S_x^+ \triangleq \{i : d_i \leq R \wedge (p_i - p_0) \cdot x^+ \geq 0\}$$
$$S_x^- \triangleq \{i : d_i \leq R \wedge (p_i - p_0) \cdot x^- > 0\}$$

Interpretabili geometricamente come segue: considerando il piano yz, S_x^+ conterrà i punti che giacciono nel semispazio posto in direzione di x^+ mentre S_x^- quelli nell'altro semispazio, quindi scegliamo come verso:

$$x = \begin{cases} x^+ & \text{se } |S_x^+| > |S_x^-| \\ x^- & \text{se } |S_x^+| < |S_x^-| \end{cases}$$

Essendo yz un piano generato in base alla distribuzione statistica dei punti del supporto per essere all'incirca a metà tra i due insiemi, il caso non infrequente in cui $|S_x^+| = |S_x^-|$ è gestito costruendo altri due insiemi:

$$\tilde{S}_x^+ \triangleq \{i \in M(k') \wedge (p_i - p_0) \cdot x^+ \geq 0\}$$
$$\tilde{S}_x^- \triangleq \{i \in M(k') \wedge (p_i - p_0) \cdot x^- > 0\}$$

Con $M(k') = i : |m - i| \leq k', m = argmedian_j d_j$ ovvero l'insieme formato dagli indici dei

k' vicini la cui distanza euclidea dal keypoint è più vicina a quella mediana. Quindi anche in questo caso avremo:

$$x = \begin{cases} x^+ & \text{se } |\tilde{S}_x^+| > |\tilde{S}_x^-| \\ x^- & \text{se } |\tilde{S}_x^+| < |\tilde{S}_x^-| \end{cases}$$

In cui non dobbiamo gestire il caso di pari cardinalità dei due insiemi se scegliamo k' dispari. Si ricava infine $y = x \times z$.

Una volta calcolato il sistema di riferimento locale si procede alla formazione dell'istogramma relativo al keypoint, considerando i k punti che cadono all'interno del supporto. La feature da inserire è il coseno dell'angolo formato tra l'asse z e la normale dell'i-esimo vicino $\hat{n}_i$, calcolata mediante prodotto scalare $z \cdot \hat{n}_i$.

L'istogramma si forma dividendo la sfera contenente il supporto in 8 divisioni per azimut, 2 per elevazione e 2 radiali (vedi fig. 2.5), ognuna di tali suddivisioni contiene 11 partizioni di istogramma per un totale di $8 \times 2 \times 2 \times 11 = 352$ valori in virgola mobile per keypoint (1344 nella versione CSHOT).

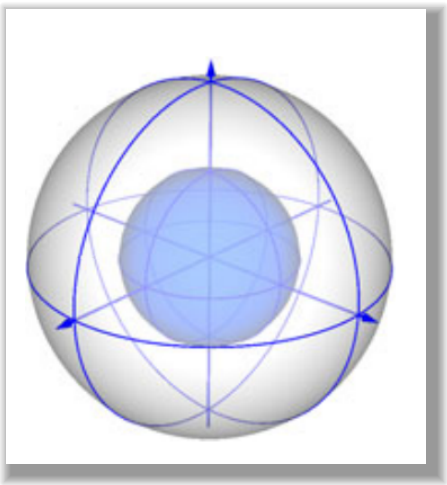

Figura 2.5: Suddivisione della sfera contenente il supporto per la formazione delle partizioni dell'istogramma in SHOT, per semplicità sono rappresentate solo 4 suddivisioni azimutali

Al fine di ridurre il rumore di quantizzazione, viene effettuata un'interpolazione quadrilinea-

re tra le partizioni adiacenti a quella in cui sarebbe dovuto cadere tale valore, in pratica, ogni coseno viene inserito proporzionalmente in quattro partizioni adiacenti in percentuali inversamente proporzionali alla distanza del valore dal centro della partizione (vedi fig. 2.6). Infine per avere robustezza alla variazione di densità di punti, viene effettuata la normalizzazione, dividendo l'istogramma per la sua norma L2.

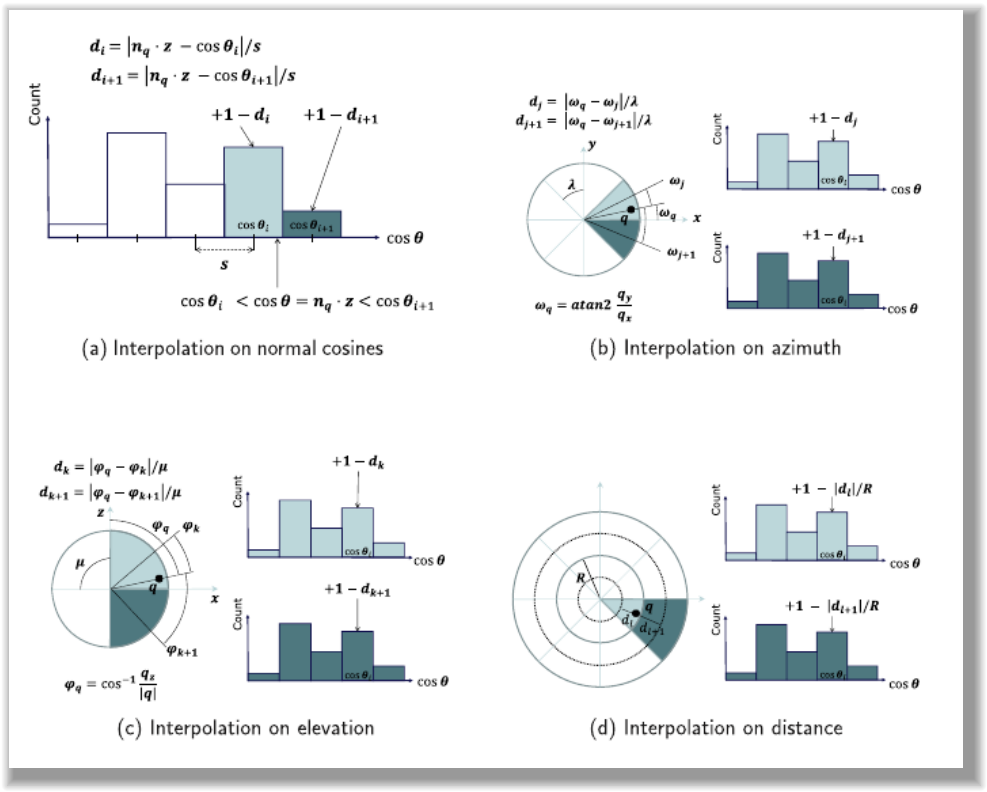

Figura 2.6: Schema di interpolazione del un valore di un coseno tra normale del keypoint e normale del vicino tra partizioni adiacenti nell'istogramma

2.2.3 SpinImages

Si tratta di uno dei primi descrittori per point cloud, a densità di punti e non fa uso di un sistema di riferimento locale.[10]

Per limitare gli effetti delle occlusioni presenti nelle viste 2.5D parziali, si applica dapprima un filtro detto *support angle* che scarta i punti del supporto p_i le cui normali $\hat{n}_i$ formano con la normale del keypoint $\hat{n}_0$ un angolo superiore ad una data soglia A_s.

I restanti punti p_i del supporto contribuiscono alla formazione dell'istogramma inserendovi, per ciascuno di essi, le coordinate cilindriche α_i e β_i [11] (vedi figura 2.7):

- α_i, la coordinata radiale, corrisponde alla distanza euclidea tra p_i e la retta su cui giace $\hat{n}_0$ mediante la formula:

$$\alpha_i = \sqrt{\|p_0 - p_i\|^2 - (\hat{n}_0 \cdot (p_0 - p_i))^2}$$

- β_i, la coordinata di elevazione, corrisponde alla distanza con segno tra p_i e il piano tangente alla superficie passante per p_0 (e perpendicolare a $\hat{n}_0$), dato dalla formula:

$$\beta_i = \hat{n}_0 \cdot (p_0 - p_i)$$

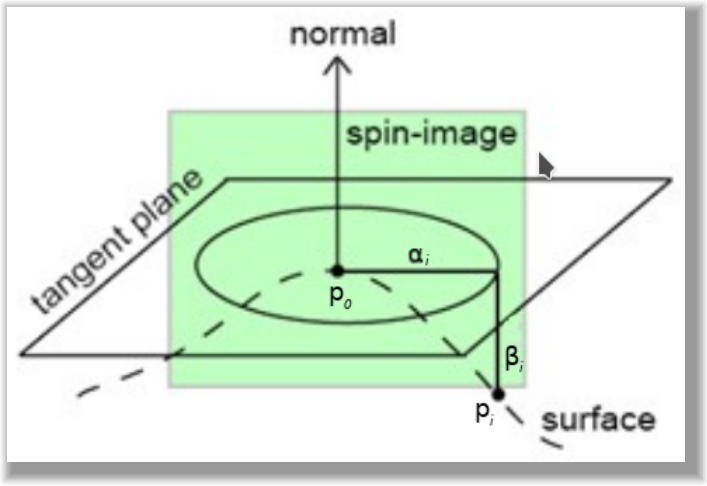

Figura 2.7: Rappresentazione geometrica dei valori α e β

Similmente a SHOT, l'inserimento dei valori nell'istogramma avviene effettuando un'interpolazione bilineare tra partizioni adiacenti per ridurre il rumore di quantizzazione introdotto dal partizionamento.

L'istogramma così ottenuto ha due dimensioni, il suo numero di partizioni ha subìto cambiamenti nelle varie implementazioni fino ad arrivare a quello attuale di 17 per la componente α e 9 partizioni per la componente β. Quindi per ogni keypoint viene istanziata una struttura dati a 2 dimensioni con un totale di $17 \times 9 = 153$ valori in virgola mobile. Essa può

essere interpretata come una proiezione 2D dei punti della superficie sul piano tangente, come si può notare in figura 2.8.

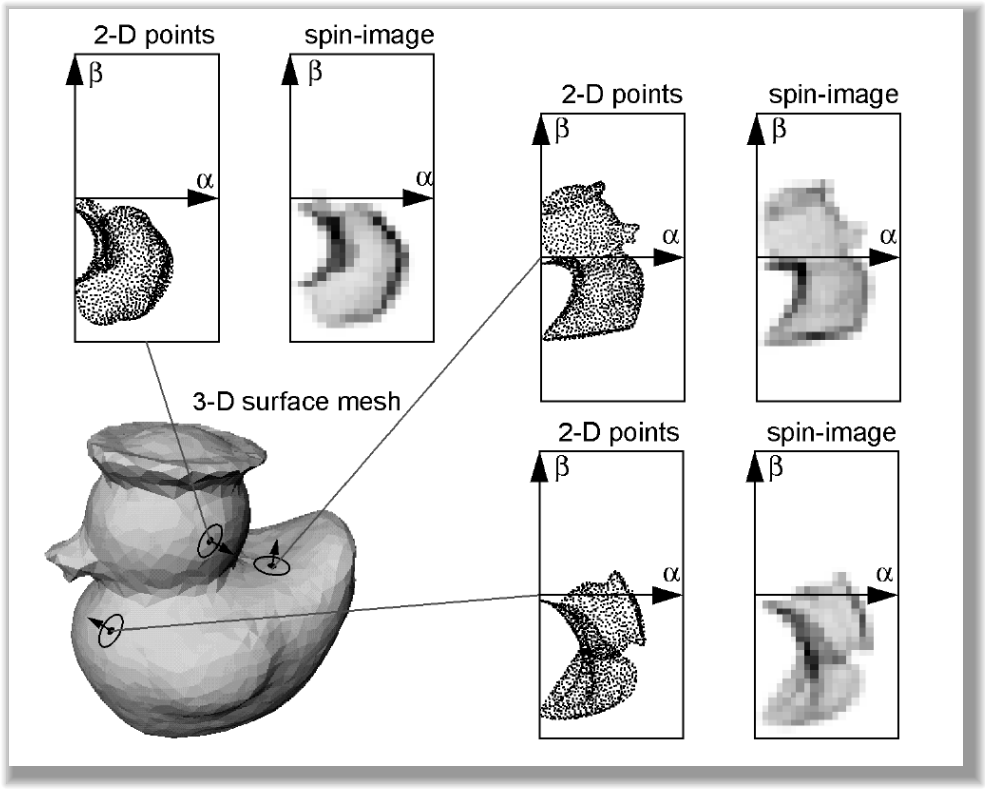

Figura 2.8: Rappresentazione del descrittore SpinImage

Capitolo 3

Metodo proposto

In questo capitolo introdurremo dei nuovi descrittori locali implementati per questo lavoro di tesi: ReSHOT e KPL-Descriptor. Successivamente saranno illustrate delle idee concepite per migliorare le performance dei descrittori discussi nel capitolo 2 che riguardano: un nuovo sistema di riferimento locale, l'uso di una strategia per filtrare il contenuto informativo del supporto ed infine un nuovo operatore per la fase di descriptor matching introdotta nel capitolo 1.

3.1 ReSHOT

ReSHOT, è una variante del descrittore SHOT. Le sue feature traggono spunto da un'estensione [12] di Point Pair Feature [13], un descrittore basato su posizione ed orientamento relativi tra due punti.

L'autore di PPF ha originariamente concepito il suo descrittore utilizzando un istogramma a quattro dimensioni, una per ciascuna delle seguenti 4 feature illustrate in figura 3.1:

- $\alpha = \angle(\hat{n}_i, \hat{n}_0)$
- $\beta = \angle(\hat{n}_0, p_i - p_0)$
- $\gamma = \angle(\hat{n}_i, p_i - p_0)$

- $\delta = \|p_i - p_0\|$

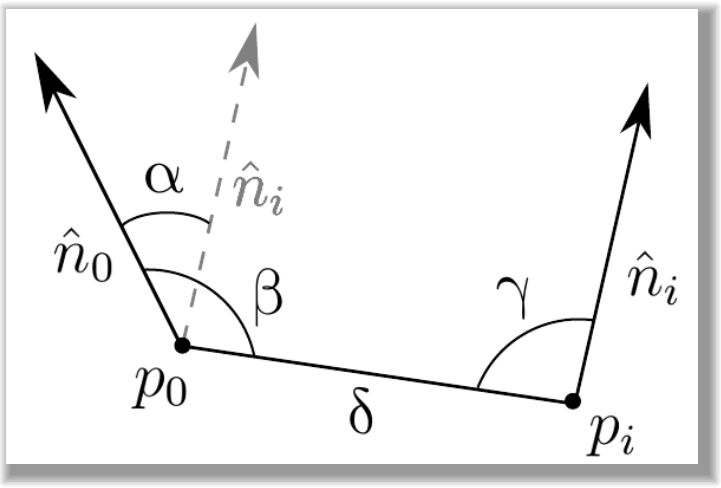

Figura 3.1: Rappresentazione geometrica della quattro feature di PPF: α, β, γ, δ

Successivamente, i test effettuati in [12] hanno verificato che: γ apportava un contributo di gran lunga maggiore delle altre feature alla performance del descrittore, mentre α e β avevano un contributo praticamente trascurabile.

Abbiamo quindi deciso di sostituire alla feature di SHOT (che coincide con il coseno di α) il coseno di γ. Il resto del descrittore è stato mantenuto identico alla sua versione originale.

3.1.1 Support angle

Un'altra idea, valutata per incrementare la performance di ReSHOT, è stata quella di ignorare, in fase di creazione dell'istogramma, i punti difficilmente ripetibili data la conformazione della superficie.

L'idea, già utilizzata in SpinImages e proposta nello stesso articolo da cui trae ispirazione la feature utilizzata in ReSHOT [12], deriva dalla considerazione che la maggior parte dei dispositivi per l'acquisizione superfici 3D, si basa su viste 2.5D, quindi, esisteranno dei punti posti in determinate zone che non potranno essere catturati dal sensore, poiché la particolare forma dell'oggetto in quel punto crea delle zone d'ombra o rispetto alla sorgente d'illuminazione o rispetto all'asse ottico del sensore (figura 3.2).

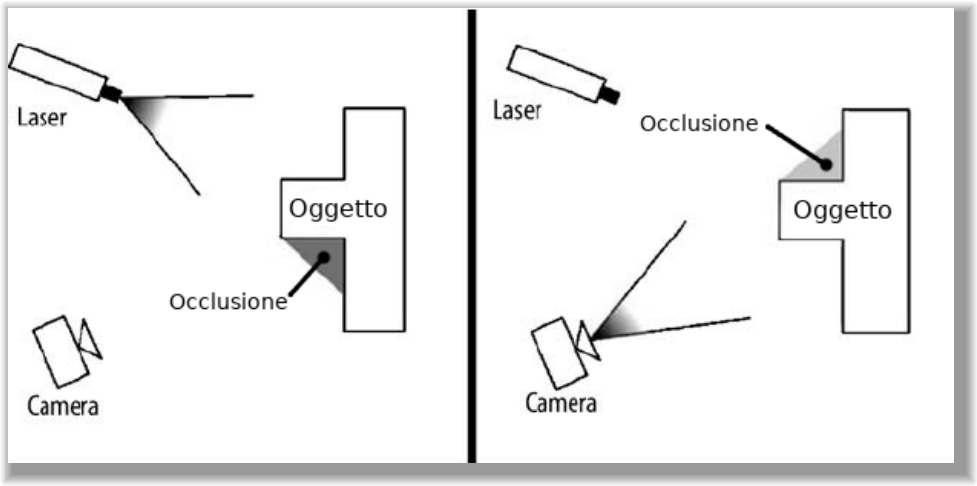

Figura 3.2: Esempio di superfici parzialmente occluse nel caso di un sistema di cattura a triangolazione laser, si noti come questo fenomeno dipende dalla posizione dell'oggetto sia rispetto al sensore che alla sorgente di illuminazione

Da questo ragionamento possiamo dedurre che, se un punto è stato acquisito correttamente da un sensore, vuol dire che la sua normale e quella dei suoi vicini differiscono al più di un certo valore.

Sono stati effettuati dei test con diverse soglie, per fare ciò è stato necessario apportare delle modifiche all'implementazione del descrittore in modo che per ogni punto del supporto si calcoli dapprima il coseno dell'angolo formato tra le normali $cos(\angle(\hat{n}_i, \hat{n}_0))$, quindi se il confronto a soglia fallisce vuol dire che l'angolo risulta troppo grande e di conseguenza la sua feature non viene considerata. Diversamente la si inserisce nell'istogramma come di consueto. In figura 3.3 è possibile vedere un esempio di supporto filtrato. Siano dunque T la soglia, D il descrittore a cui applicare questo filtro, d_i la feature dell'i-esimo punto della point cloud, il nuovo descrittore $\tilde{D}$ sarà formato da:

$$\tilde{D} = \{d_i \in D : \hat{n}_0 \cdot \hat{n}_i \geq T \quad \forall i \in (1...k)\}$$

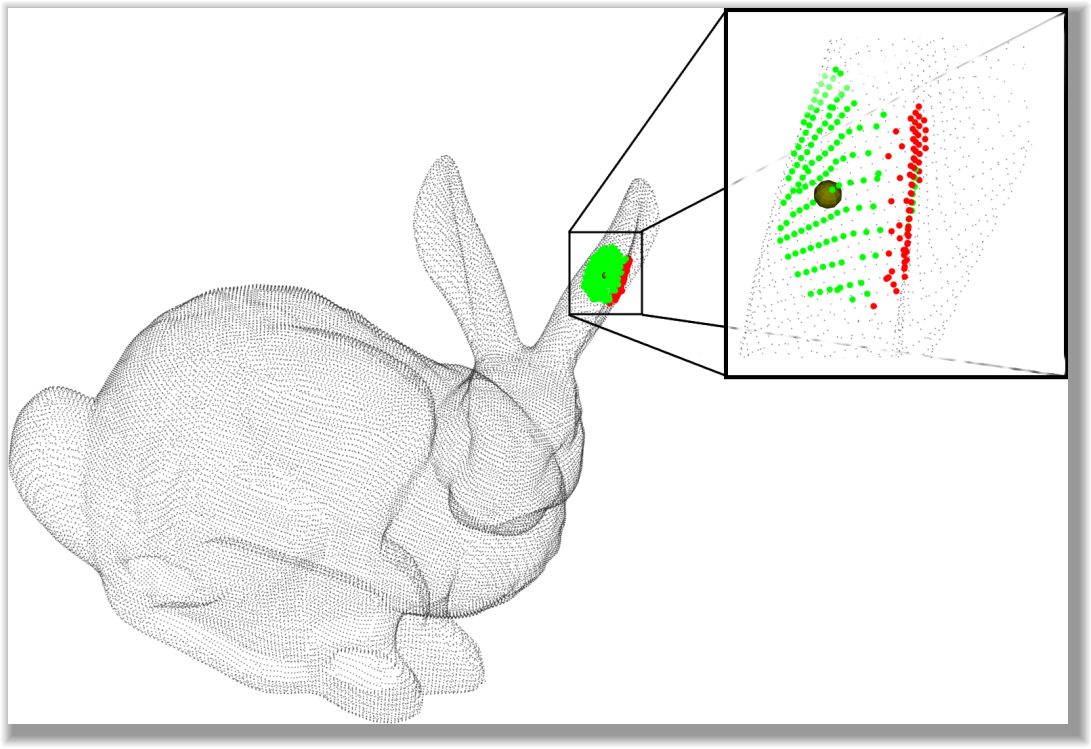

Figura 3.3: Esempio di filtro al supporto su Bunny, per far risaltare l'effetto del filtro è stata applicato sulla scena già registrata. La sfera gialla è il keypoint, in verde i punti del supporto sopra la soglia, in rosso quelli sotto la soglia e quindi da scartare. Si noti come i punti scartati siano orientati in una direzione fortemente angolata rispetto al keypoint.

3.1.2 Sistema di riferimento locale

Le performance di un descrittore dipendono dal sistema di riferimento locale, come già dimostrato in [14]. Per questo motivo si è deciso di investigare su come le performance di ReSHOT migliorino con un sistema di riferimento locale diverso da quello con cui è stato proposto.

Il LRF testato si chiama Flare [15] e viene calcolato considerando dapprima un supporto ridotto dato dai punti posti fino ad una distanza di $5mr$ dal keypoint con mr pari alla metà della risoluzione media della point cloud. Il piano tangente al keypoint xy sarà dato dalla regressione lineare dei punti di questo sottoinsieme.

Con riferimento al metodo *Board*[1] [14] si calcola dapprima la direzione dell'asse z come retta normale al piano xy, otteniamo dunque due valori possibili per z che chiamiamo z^+ e z^-. Si calcola, quindi, la media delle normali dei k' punti del supporto ridotto:

$$\tilde{n} = \frac{1}{k'} \sum_{i=1}^{k'} \hat{n}_i$$

z sarà il versore il cui prodotto scalare con $\tilde{n}$ è positivo, ovvero:

$$z = \begin{cases} z^+ & \text{se } \tilde{n}_i \cdot z^+ \geq 0 \\ z^- & \text{se } \tilde{n}_i \cdot z^+ < 0 \end{cases}$$

Per l'estrazione dell'asse x si considerano i punti del supporto originale p_i e la loro proiezione p'_i sul piano xy, in particolare siamo interessati al punto p_{max} la cui distanza con segno D dal piano xy è maggiore, ovvero:

$$p_{max} = \max_i(\|p_i - p'_i\|_2) \quad \forall i = (i...k)$$

L'asse x sarà il vettore normalizzato che congiunge il keypoint e la proiezione di p_{max}:

$$x = \frac{p'_{max} - p_0}{|p'_{max} - p_0|}$$

Infine analogamente al metodo di SHOT si calcola l'asse y come prodotto vettoriale tra x e z.

In figura 3.4 è presente un esempio del LRF calcolato su un keypoint di una porzione di superficie.

[1] Il nome è quello scelto successivamente al paper e messo nella relativa implementazione nella libreria PCL

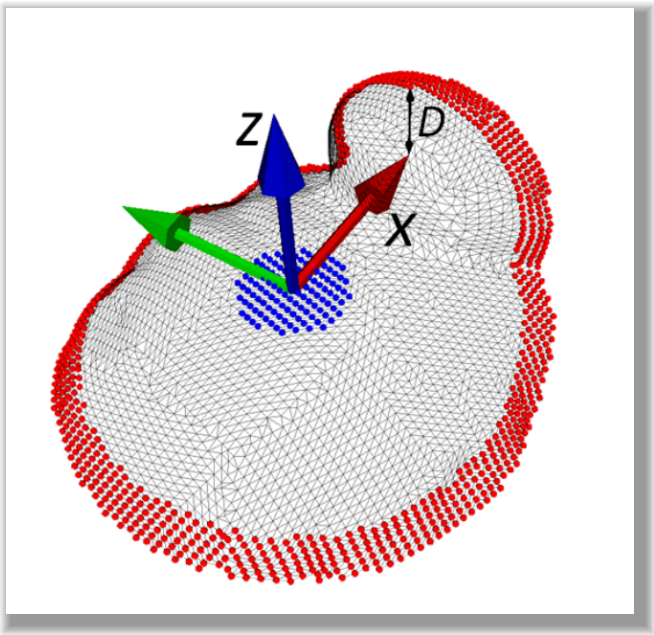

Figura 3.4: Rappresentazione dei sottoinsiemi del supporto di un keypoint in Flare per il calcolo del sistema di riferimento locale: in rosso i punti utilizzati per l'estrazione dell'asse x, in blu quelli per l'asse z

3.2 KeyPoint Learning Descriptor

KPL-Descriptor è un descrittore implementato per questo lavoro di tesi che usa la feature di KPL, un algoritmo di keypoint detection basato su machine learning [2], che addestra un agente di intelligenza artificiale alla selezione dei keypoint all'interno di una point cloud. La feature è la stessa di SHOT ovvero il coseno dell'angolo tra $\hat{n}_0$ e $\hat{n}_i$.

Diversamente da SHOT però, la sfera contenente i punti del supporto non viene suddivisa in azimut ed elevazione, ma solo in senso radiale. Questo tipo di suddivisione genera una serie di corone circolari (vedi figura 3.5) che rendono il descrittore rotation invariant e ne abbassando il tempo di computazione, dato che non è più necessario il calcolo del LRF.

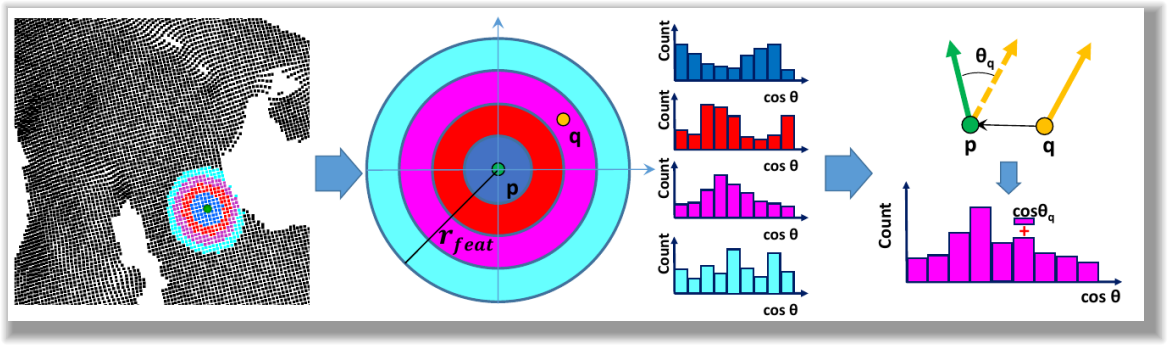

Figura 3.5: Rappresentazione schematica della formazione di KPL-Descriptor, a sinistra la suddivisione del supporto rappresentato su una nuvola di punti, al centro il collegamento tra suddivisioni e partizionamento dell'istogramma, a destra un esempio di incremento di una partizione

Anche in questo caso viene effettuata un'interpolazione bilineare per ridurre il rumore di quantizzazione ed infine normalizzato l'istogramma. Nella nostra implementazione sono state scelte 5 divisioni radiali ad ognuna delle quali corrisponde un istogramma con 10 partizioni. Quindi la struttura dati di questo descrittore sarà composta da $5 \times 10 = 50$ valori in virgola mobile per keypoint.

3.3 KdTree con distanza chi quadrato

Quest'ultimo metodo proposto, diversamente dai precedenti, non interviene nella fase di computazione del descrittore, ma nella creazione delle corrispondenze tra keypoint di due viste.

In questa fase, viene valutata la somiglianza tra i descrittori. Data una coppia di viste 2.5D dette rispettivamente *sorgente* e *target*, se prendiamo un keypoint da ciascuna di esse che corrispondono allo stesso punto della scena, i corrispettivi descrittori saranno tra di loro simili.

Come già anticipato nel capitolo precedente, i descrittori sono rappresentati da strutture dati contenenti n valori in virgola mobile, Quindi possiamo considerare un descrittore come un vettore che appartiene ad $\mathbb{R}^n$.

Solitamente, la metrica utilizzata per calcolare la loro similarità è la distanza euclidea nello spazio dei descrittori. All'interno di questo test, abbiamo considerato come metrica la distanza chi quadro. Come proposto in [12].

Per descrivere matematicamente la differenza tra le due metriche è utile stabilire una notazione, definiamo pertanto:

- D il vettore rappresentativo del descrittore
- n la dimensione del vettore d
- s e t rispettivamente le scena sorgente e target

Il primo operatore è la distanza L_2, ovvero la norma L_2 del vettore differenza:

$$D_{L_2} = \sum_{i=1}^{n} (D_i^{(s)} - D_i^{(t)})^2$$

Mentre il secondo è dato dalla distanza χ^2 definito da:

$$D_{\chi^2} = \sum_{i=1}^{n} \frac{(D_i^{(s)} - D_i^{(t)})^2}{D_i^{(s)} + D_i^{(t)}}$$

Capitolo 4

Strumenti utilizzati

In questo capitolo presentiamo gli strumenti e i dataset sui quali abbiamo operato all'interno di questo progetto di tesi. Gli algoritmi sono stati implementati in linguaggio C++ utilizzando la libreria PCL.

4.1 Point Cloud Library

PCL è una libreria open source sviluppata in C++ e rilasciata sotto licenza BSD, contiene una serie di algoritmi applicabili alle point cloud mediante i quali è possibile effettuare: filtraggio, stima delle feature, ricostruzione di superfici, registrazione, model fitting, segmentazione e molto altro. Essa è supportata da una comunità internazionale di ricercatori sulla computer vision e sulla robotica. La versione utilizzata per i nostri test è la 1.8. [16]

Figura 4.1: Logo di PCL

4.2 Dataset

Per la validazione degli algoritmi proposti sono stati utilizzati due dataset, rispettivamente della Stanford University e del gruppo Princeton Vision & Robotics, entrambe sono disponibili gratuitamente in rete. Abbiamo scelto due dataset eterogenei per verificare quanto il metodo proposto sia robusto.

Ognuno dei file dei dataset rappresenta una vista 2.5D ed è detto *fragment*. I file di 3DMatch sono raggruppati per scene mentre quelli stanford repository per oggetto.

4.2.1 Stanford 3D Scanning Repository

Questo dataset è stato pubblicato dall'università di Stanford, California, ed è formato da nove scene raffiguranti delle statue di modeste dimensioni, acquisite inizialmente come range image, successivamente convertite a nuvole di punti e registrate con una versione

modificata dell'algoritmo ICP [17]. I dati della registrazione di ogni scena sono contenuti in una lista di vettori per la traslazione e di quaternioni per la rotazione.[1]

Nei nostri test abbiamo utilizzato quattro dei nove oggetti disponibili nel dataset, tutti acquisiti negli Stanford University Computer Graphics Laboratory con lo scanner Cyberware 3030 MS in figura 4.2:

Figura 4.2: Scanner Cyberware 3030 MS utilizzato per

- Stanford Bunny (figura 4.3): è suddiviso in 10 fragment e contiene oltre 350000 punti.

[1] The Stanford 3D Scanning Repository: `http://graphics.stanford.edu/data/3Dscanrep/`

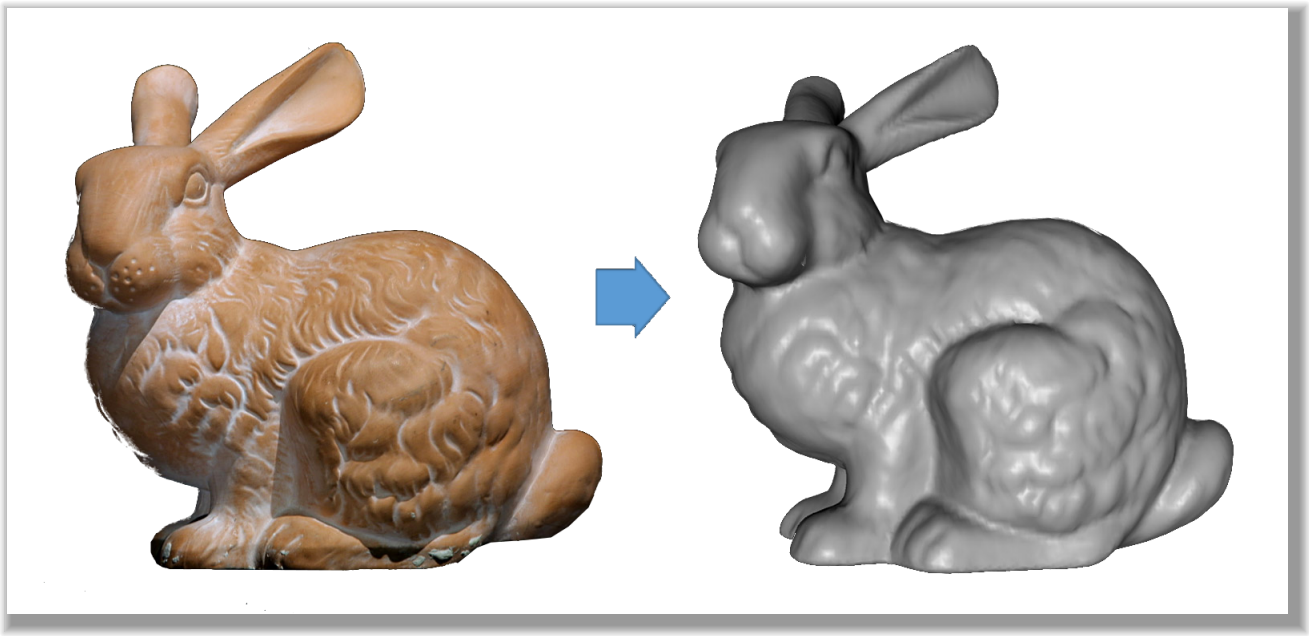

Figura 4.3: Immagine e Rappresentazione di Stanford Bunny

- Armadillo (figura 4.4): è suddiviso in 114 fragment e contiene un totale di oltre 3300000 punti.

Figura 4.4: Rappresentazione di Armadillo

- Happy Buddha (figura 4.5): è suddiviso in 60 fragment e contiene un totale di oltre 4500000 punti.

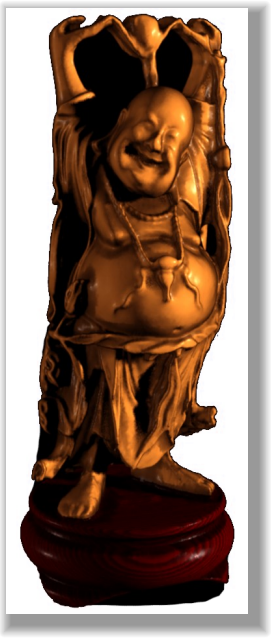

Figura 4.5: Immagine di HappyBuddha

- Dragon (figura 4.6): è suddiviso in 70 fragment e contiene un totale di oltre 2700000 punti.

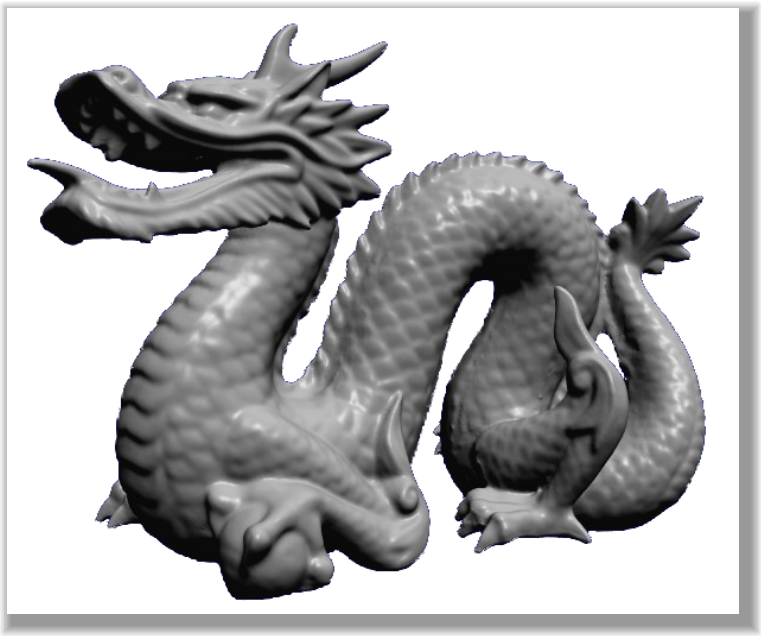

Figura 4.6: Rappresentazione Dragon

4.2.2 3DMatch

Questo dataset è stato e pubblicato dal laboratorio Princeton Vision & Robotics appartenente all'università di Princeton, New Jersey, per un progetto finalizzato alla realizzazione dell'omonimo descrittore ad attributi geometrici per point cloud con informazioni RGB-D basato sul machine learning. [18]

È composto da 62 scene raffiguranti delle intere stanze arredate, 54 sono state utilizzate per l'addestramento e 8 per il benchmark di 3DMatch, ed è su queste ultime che noi abbiamo svolto i nostri test.

Esse provengono a loro volta da 2 dataset, il primo è 7 Scenes: realizzato nell'ambito di un progetto di ricerca di Microsoft [19], è stato acquisito tramite una camera Kinect RGB-D che usa un sensore structured light, con risoluzione di 640×480, delle sette scene di cui è formato è stata utilizzata solo Red Kitchen che è suddivisa in 60 fragment.

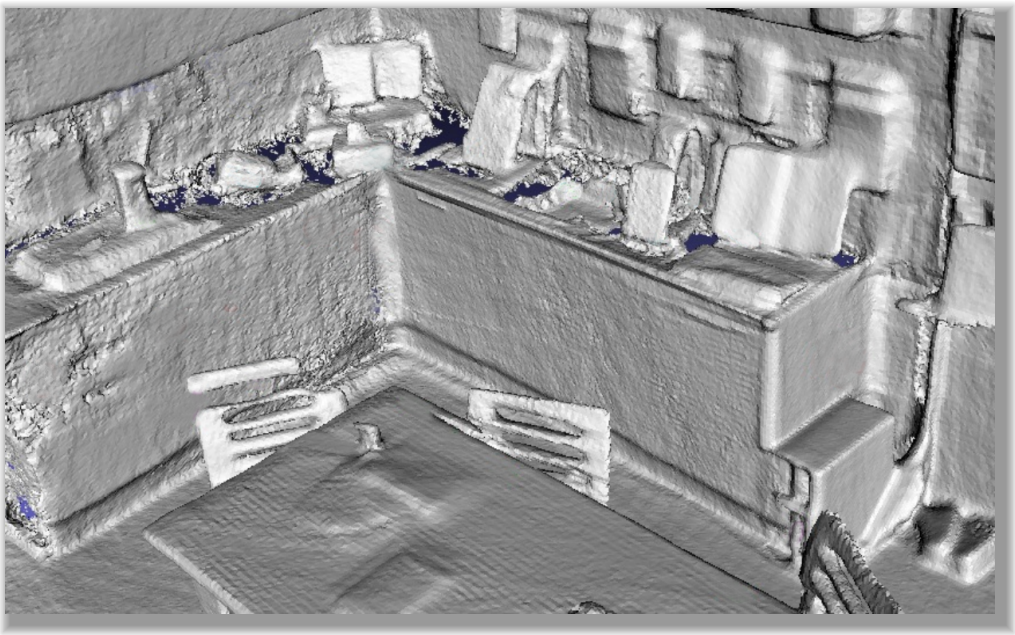

Figura 4.7: Rendering di una vista di Red Kitchen

Il secondo dataset è SUN3D: un'ampia raccolta di scene realizzate dalla stessa Princeton University, acquisite tramite una camera ASUS Xtion PRO LIVE posizionata su un Laptop e portata a spalla da un operatore (vedi figura 4.8) per mimare un'esplorazione umana, anch'essa basata su un sensore a luce strutturata con risoluzione 640×480 ed è distribuito assieme ad un tool che permette agevolmente di etichettare gli oggetti 3D e facilitarne, così, la personalizzazione per test di machine learning. È formato da 254 scene, di cui ne sono state utilizzate soltanto 7, ognuna di esse contiene un numero variabile di fragments che va da 39 a 60, in figura 4.9 i rendering di alcune scene di questo dataset.

Figura 4.8: Posizionamento del sensore e posa dell'operatore per l'acquisizione del dataset SUN3D

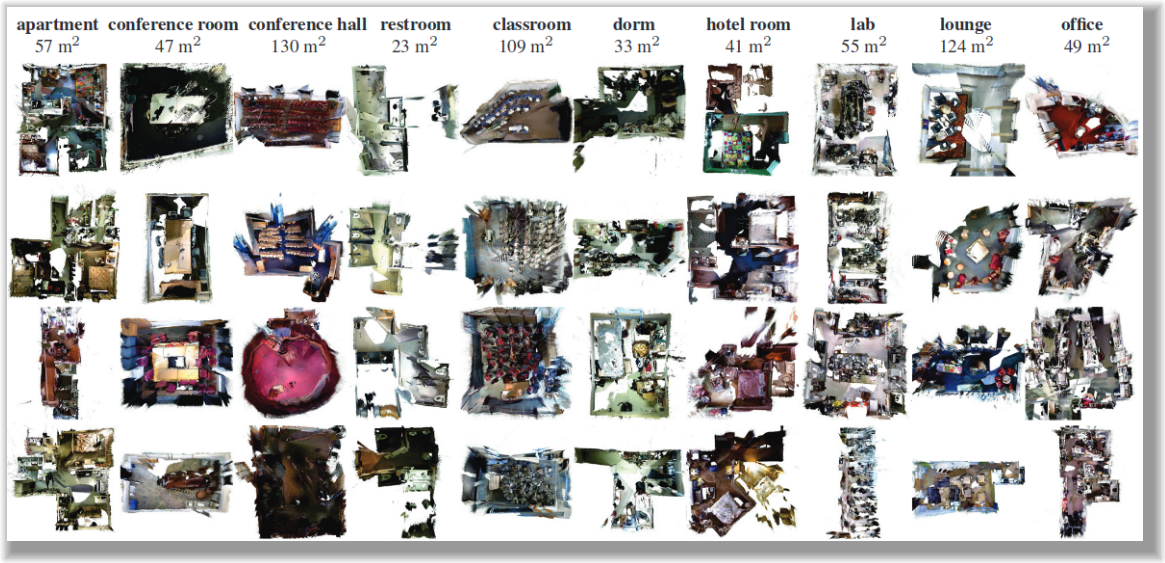

Figura 4.9: Esempi di rendering dal dataset SUN3D

Capitolo 5

Risultati sperimentali

In quest'ultimo capitolo descriveremo il protocollo utilizzato per testare i descrittori discussi nei capitoli 2 e 3. Successivamente introdurremo le metriche per la valutazione dei test di cui, infine, verranno mostrati i risultati finali.

5.1 Protocollo di test

Descriveremo ora il metodo con cui sono stati condotti i test sull'intera pipeline di feature matching.

Per ogni dataset, sono stati importati i fragment di ogni scena e con essi formate tutte le possibili coppie. Ogni coppia è costituita da una vista *Sorgente* S e una *Target* T a cui si aggiunge la matrice di ground truth M, che rappresenta lo spostamento rigido tra le due viste.

I keypoint sono stati selezionati tra i punti delle due viste in modo che fossero uniformemente distribuiti (criterio *Uniform Sampling*) e tra quelli di S è stato formato il sottoinsieme O per cui esiste una corrispondenza in T:

$$O = \{p \in S \land \exists q \in T : pM = q\}$$

Quindi avremo che i punti $q_p = pM$, sono le corrispondenze reali dei punti p le cui coordinate sono espresse con il sistema di riferimento di T.

Infine sono stati calcolati e confrontati i descrittori D_p e D_q rispettivamente per i punti di S e T, per cercare tra di essi le corrispondenze stimate:

$$\tilde{q}_p = \min_{q \in T}(dist(D_p, D_q))$$

Quindi, per ogni punto $p \in S$ avremo una corrispondenza stimata $\tilde{q}_p \in T$ data dal descrittore D_q a minima distanza da D_p.

L'output del test è la distanza euclidea tra q_p e $\tilde{q}_p$, per ciascuna coppia confrontata. I test sono stati condotti sui seguenti descrittori:

- FPFH (capitolo 2.2.1)
- SHOT(capitolo 2.2.2)
- SpinImages (capitolo 2.2.3)
- ReSHOT (capitolo 3.1)
- KPL-Descriptor con angolo SHOT (capitolo 3.2)
- KPL-Descriptor con angolo ReSHOT

5.2 Metriche

5.2.1 Percentuale di corrispondenze esatte

Per questa metrica si imposta una distanza di match d_m e si considera esatto un match tra p e $\tilde{q}_p$ se la distanza tra q_p e $\tilde{q}_p$ non supera d_m.

La distanza di match è stabilita in base alla rumorosità del sensore con cui sono state catturate le viste del dataset.

Si considera l'insieme O_{true} dei keypoint la cui corrispondenza è considerata esatta:

$$O_{true} \triangleq \{p \in O : \|\tilde{q}_p - q_p\| < d_m\}$$

E $O_{false} = O \setminus O_{true}$, quindi il valore finale del test è dato da:

$$P = \frac{m_{true}}{m_{true} + m_{false}} \quad \text{con } m_{true} = |O_{true}| \quad \text{e con } m_{false} = |O_{false}|$$

ovvero rapporto tra le corrispondenze esatte e quelle totali.

Il valore P restituito dal test è la *Precision*. Esso è mediato su tutte le coppie di viste, su tutte le scene del dataset ed infine espresso in percentuale.

5.2.2 Percentuale di viste registrate

Questa metrica è stata utilizzata per testare PPF-Foldnet [20], un descrittore basato sull'apprendimento non supervisionato, che è stato addestrato sugli stessi dataset utilizzati nei nostri test.

L'idea è quella di valutare la percentuale di registrazioni correttamente eseguite effettuandole con il metodo RANSAC, un algoritmo altamente affidabile ma che richiede un grande numero di iterazioni.

Tuttavia, essendo interessati al confronto tra descrittori piuttosto che all'esito della registrazione vera e propria, abbiamo snellito la procedura stabilendo soltanto la inlier ratio, ovvero la percentuale τ di match esatti tra due fragment secondo la metrica definita nel paragrafo precedente.

Con $\tau = 5\%$, il metodo RANSAC raggiunge, effettuando più di 50000 iterazioni, il 99,9% di confidenza trovando le 3 corrispondenze necessarie ad effettuare la registrazione. Perciò, se un descrittore effettua i match tra due viste superando questa soglia, avremo un'altissima probabilità che la registrazione avverrebbe con successo senza doverla realmente eseguire.

Consideriamo l'insieme F dei fragment di una scena ed S quello formato da tutte le sue possibili coppie:

$$S \triangleq \{(f_i, f_j) \quad f_i \in F, f_j \in F \quad \forall i, j = (1, \ldots, |F|) \wedge i \neq j\}$$

Definiamo S_τ, come sottoinsieme di S la cui percentuale di match esatti supera τ, ovvero:

$$S_\tau \triangleq s_i \in S \wedge r_{in}(s_i) > \tau$$

L'output R di questo test, ovvero la *Recall*, è dato dalla percentuale di coppie con match esatti sopra la soglia τ:

$$R = \frac{|S_\tau|}{|S|}$$

Il valore scelto per il nostro test, sia sul dataset Stanford View che 3DMatch è di $\tau = 5\%$.

5.3 Test eseguiti

In tabella 5.1 sono riportati i parametri utilizzati e nei paragrafi a seguire, i grafici di tutti i test.

Parametro	Valore per Stanford	Valore per 3DMatch	Descrizione
$leaf$	1mm	1cm	Densità di sottocampionamento
d_m	7mm	10cm	Soglia di match
r_{det}	vedi tuning	30cm (*)	Densità di keypoint
r_{desc}	vedi tuining	30cm (*)	Raggio del supporto

Tabella 5.1: Parametri usati nel test

(*) Il raggio del supporto per il dataset 3DMatch, è quello testato in [20].

5.3.1 Tuning dei raggi

Questo primo test è stato effettuato sul dataset Stanford, e permette di confrontare come ciascun descrittore muta la propria performance al variare del raggio del supporto. I raggi

testati sono: 2cm, 4cm e 6cm.

I risultati in figura 5.1 evidenziano che FPFH ha delle performance marcatamente migliori con raggi più piccoli, al contrario di SpinImages. La differenza di performance degli altri descrittori, invece, è meno pronunciata.

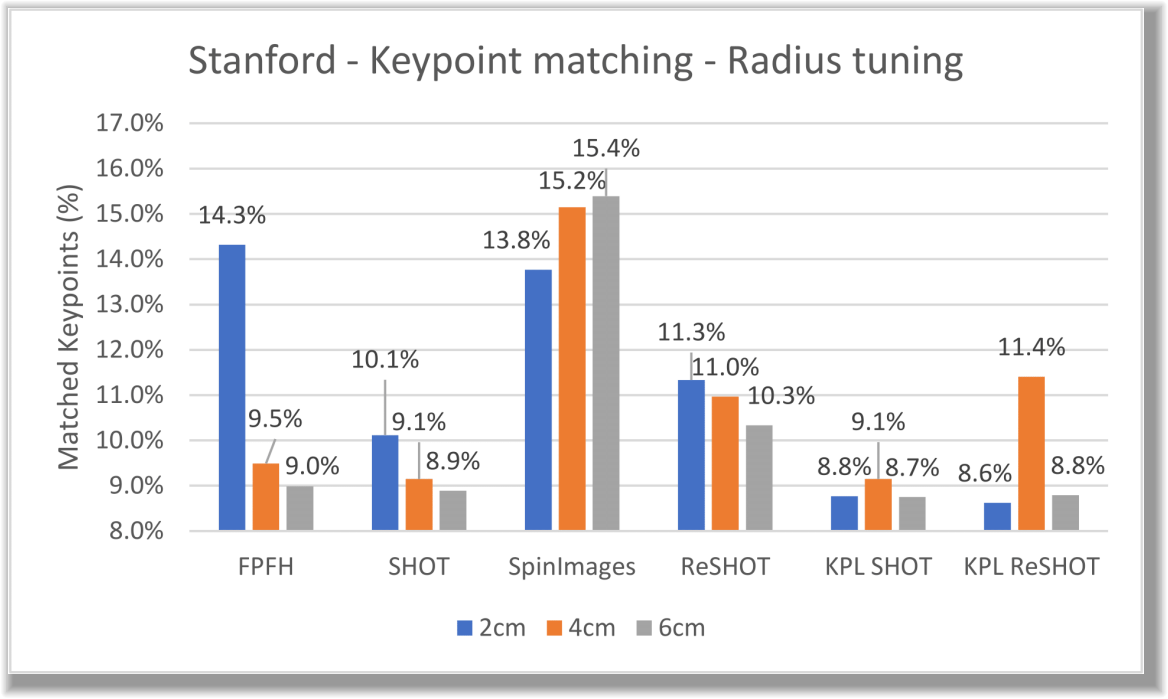

Figura 5.1: Percentuale di corrispondenze esatte per i 6 descrittori al variare del raggio del supporto, 2cm, 4cm e 6cm, Dataset Stanford View

5.3.2 Tempi di esecuzione per punto

Questo test è stato effettuato eliminando le ottimizzazioni per il calcolo parallelo, il grafico in figura 5.2 evidenzia come KPL-Descriptor, nelle due sue varianti, abbia un tempo di esecuzione decisamente inferiore agli altri descrittori.

I raggi di supporto sono stati scelti tra quelli che hanno avuto una performance migliore nel test precedente il che fornisce un valore più legato alla prestazione che al descrittore in sè.

SpinImages ed FPFH, invece sono i descrittori computazionalmente più onerosi, il primo, in particolare, benché mostri una performance migliore degli altri in prima battuta, risente notevolmente della maggior grandezza del raggio.

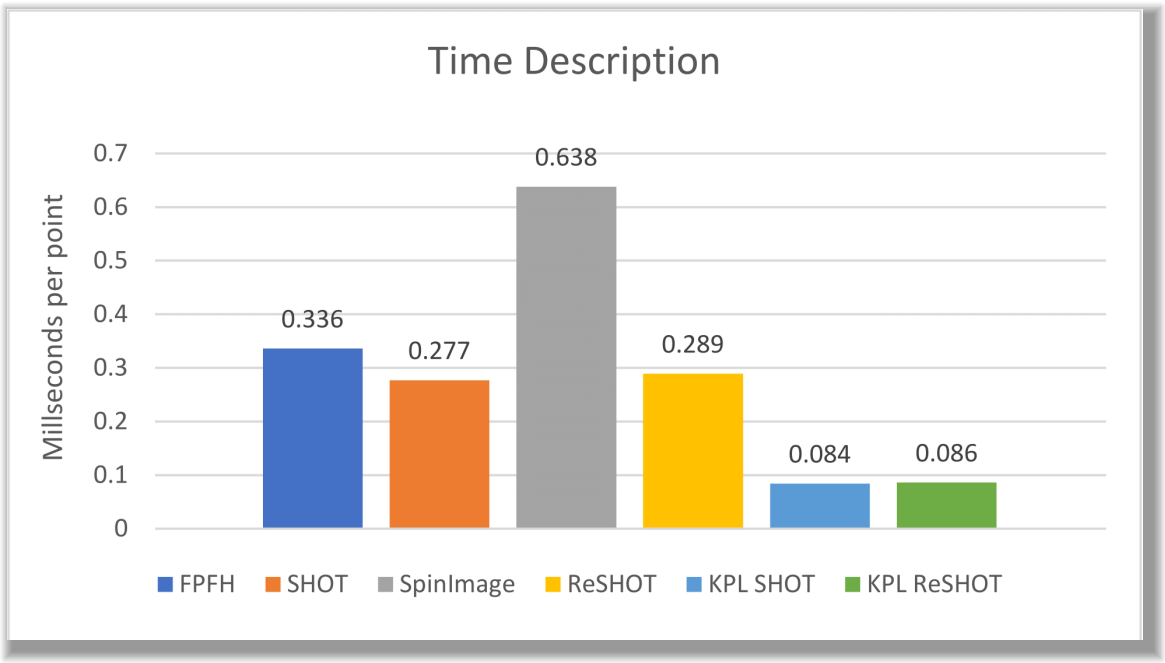

Figura 5.2: Tempi di calcolo per punti di ciascun descrittore espressi in millisecondi, il raggio scelto è quello della miglior performance del punto precedente

5.3.3 Confronto tra sistemi di riferimento

Questo test, effettuato sul dataset Stanford, ci ha permesso di comparare l'influenza del sistema di riferimento locale sui descrittori SHOT e ReSHOT.

Dal grafico in figura 5.3 vediamo come entrambi beneficiano significativamente dell'utilizzo di Flare che si rivela, dunque, più stabile rispetto a quello built-in di SHOT.

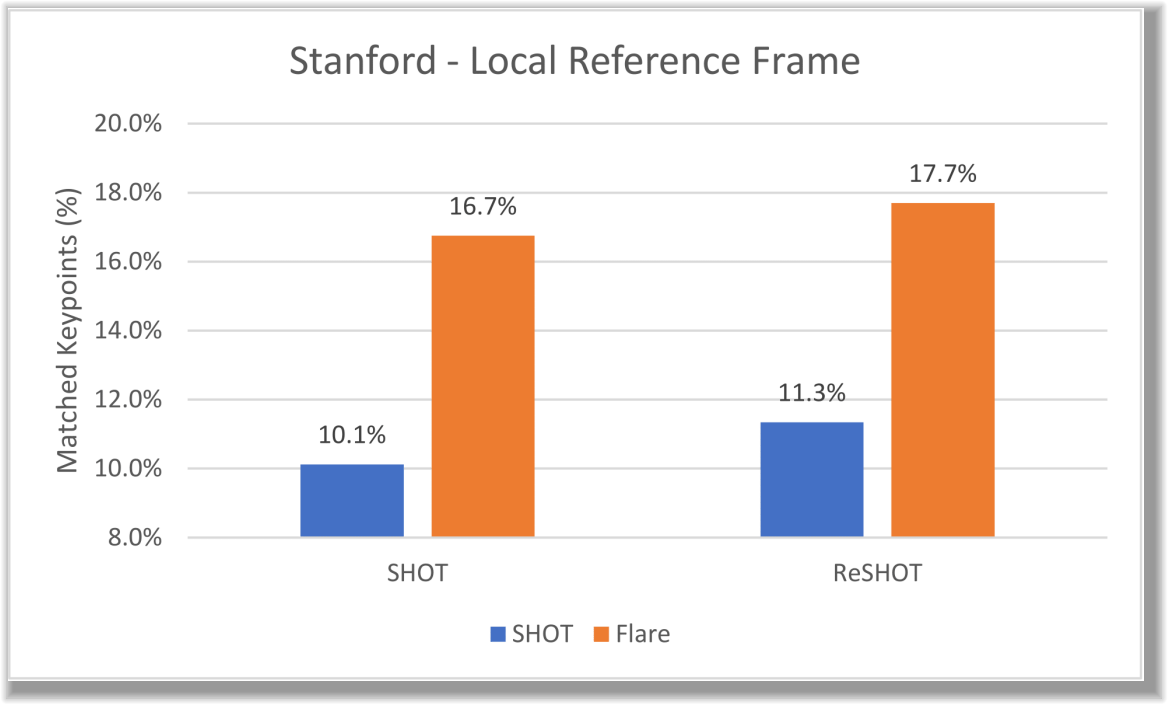

Figura 5.3: Percentuale di corrispondenze esatte per SHOT e ReSHOT con sistema di riferimento SHOT e Flare

5.3.4 Tuning dei support angle

Sono stati testati sia su dataset Stanford che 3DMatch due valori di support angle secondo il metodo proposto nel capitolo 3.1.1. Al di sotto di tale valore, i punti del supporto vengono ignorati nella formazione dell'istogramma.

Per quanto riguarda Stanford View, il grafico in figura 5.4 mostra che i punti del supporto che apportano un maggiore contributo in termini di performance dei descrittori sono quelli la cui normale ha bassa variazione angolare rispetto al keypoint. In particolare, risulta più conveniente generare l'istogramma con i soli punti orientati con angoli inferiori a $\frac{\pi}{3}$.

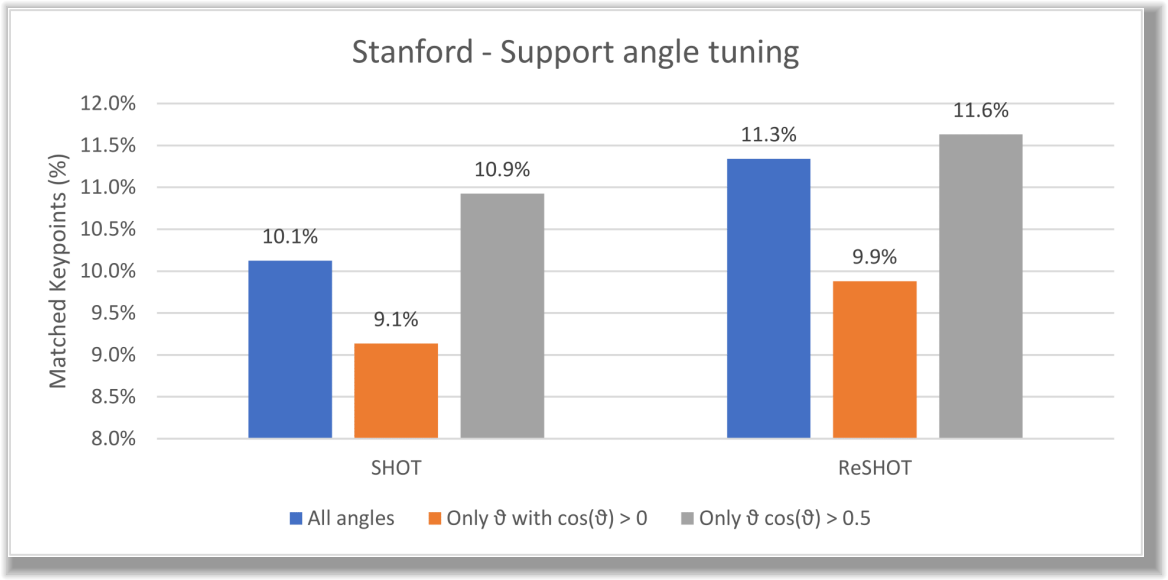

Figura 5.4: Percentuale di corrispondenze esatte nel dataset Stanford View per SHOT e ReSHOT senza filtro e filtrando il supporto dai punti con angoli maggiori di 90°e 66,6°

Il grafico a figura 5.5, mostra che il dataset 3DMatch, invece, non trova alcun beneficio dall'applicazione di tali soglie. Il motivo è probabilmente dovuto sia alla diversa conformazione dei due dataset che alla scelta di un supporto più grande in quest'ultimo.

Infatti, mentre Stanford View ritrae singoli oggetti, 3DMatch contiene delle scene molto ampie e con superfici eterogenee. Questo fatto, unito alla scelta di un maggiore raggio del supporto (che ingloba al suo interno, quindi, un maggior numero di irregolarità), induce a pensare che non conviene sacrificare il contenuto informativo apportato da tali irregolarità.

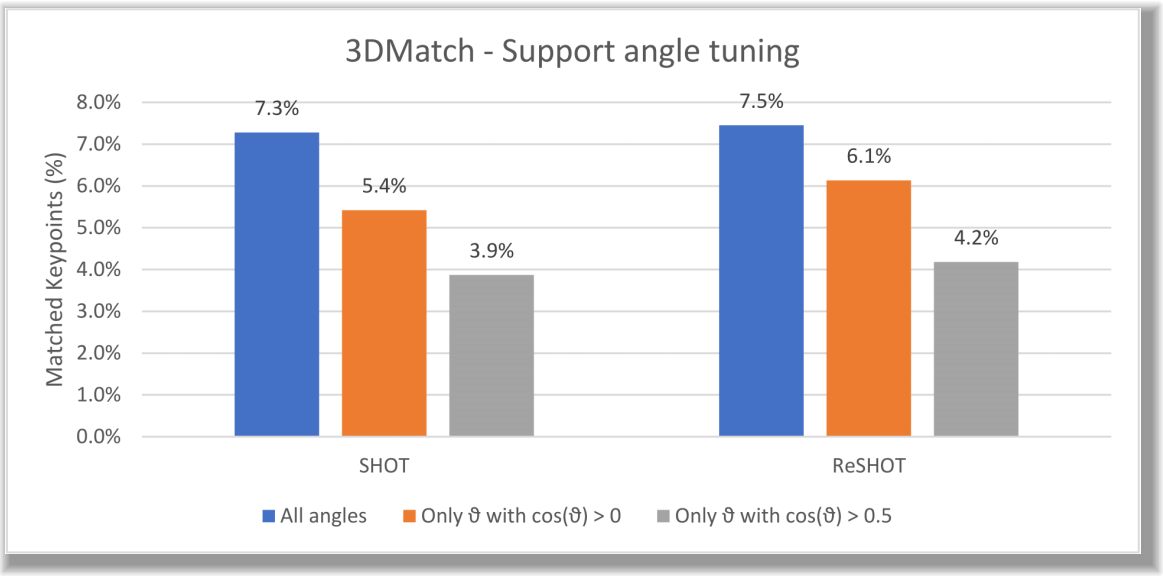

Figura 5.5: Percentuale di corrispondenze esatte nel dataset 3DMatch per SHOT e ReSHOT senza filtro e filtrando il supporto dai punti con angoli maggiori di 90°e 66,6°

5.3.5 Confronto tra distanze L2 e Chi quadrato

L'ultimo test prima del confronto di tutti i risultati ottenuti riguarda la sola fase di matching ed in particolare la distanza tra descrittori discussa nel capitolo 3.3. Questo test è stato effettuato dapprima solo su SHOT e ReSHOT dato che l'uso della distanza χ^2 è stata suggerita dall'autore di PPF [12], che ha ispirato ReSHOT.

Come vedremo a breve, questo non migliora soltanto la performance dei due descrittori i cui risultati sono riportati nel grafico di figura 5.6

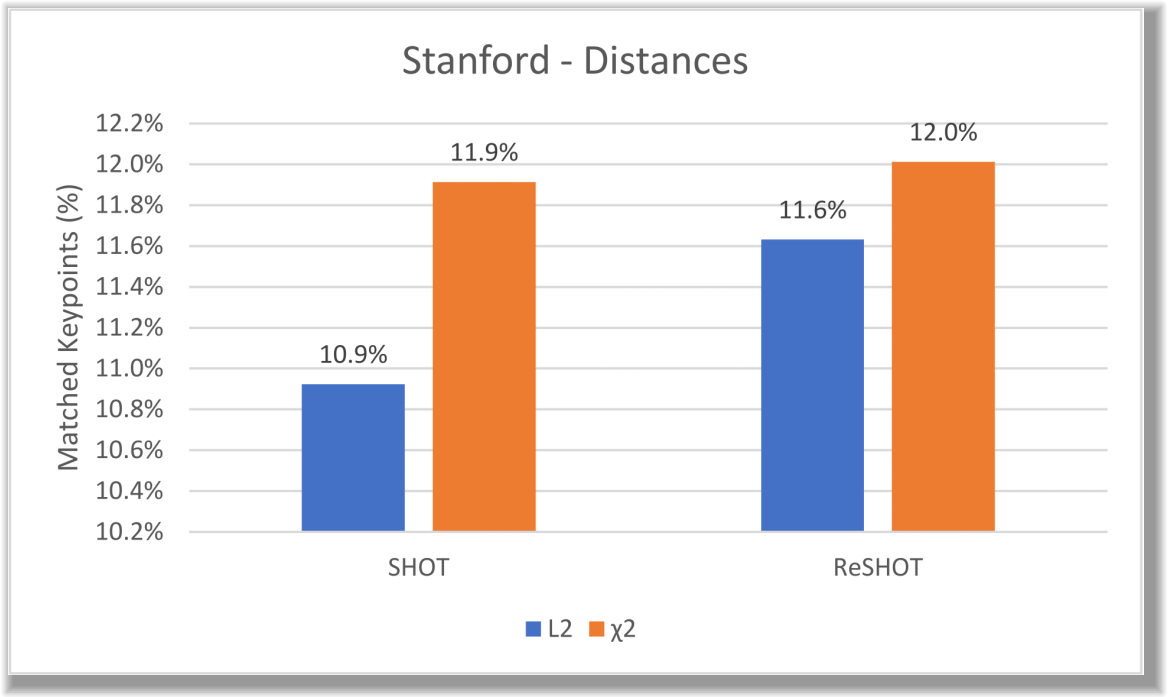

Figura 5.6: Percentuale di corrispondenze esatte per SHOT e ReSHOT utilizzando come minima distanza di match prima L_2 e poi χ^2, Dataset Stanford View

5.3.6 Confronti finali

In quest'ultima parte riportiamo, per ciascuno dei descrittori discussi, i confronti tra la versione originale dell'algoritmo e l'apporto di tutti i metodi proposti, al fine di riscontrarne il complessivo aumento delle performance.

Per quanto riguarda i descrittori FPFH e SpinImages, KPL-Descriptor SHOT e KPL-Descriptor ReSHOT (gli ultimi due sono considerati descrittori separati) l'unico metodo proposto compatibile è stato quello della distanza χ^2 in fase di matching del descrittore.

La versione base di SHOT e ReSHOT (anch'essi considerati come descrittori separati), è stata confrontata una versione in cui oltre alla distanza χ^2 è stato sommato l'apporto del cambio di sistema di riferimento locale Flare e il tuning del support angle.

In tutti i casi osserviamo un miglioramento delle performance all'applicazione del metodo proposto:

I grafici presenti nelle figure 5.7, 5.8 e 5.9 ci mostrano le percentuali di match esatti e la Recall sul dataset Stanford. Il secondo, in particolare, evidenzia l'aumento di match considerati positivi all'aumentare della distanza d_M.

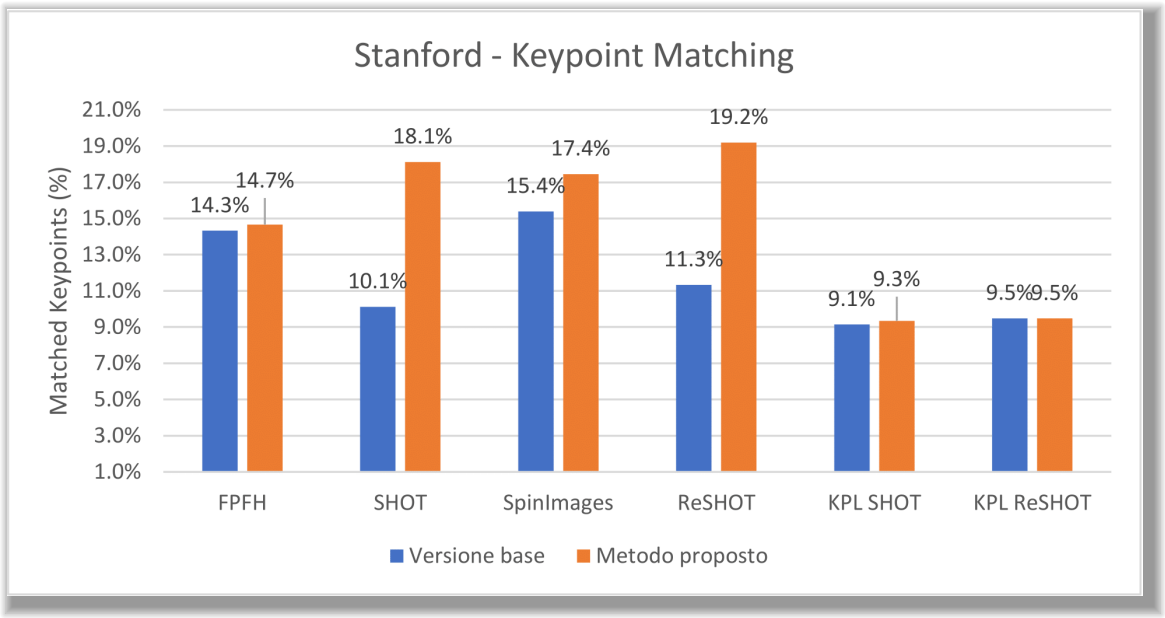

Figura 5.7: Percentuale di corrispondenze correttamente individuate su dataset Stanford prima e dopo l'applicazione dei metodi proposti

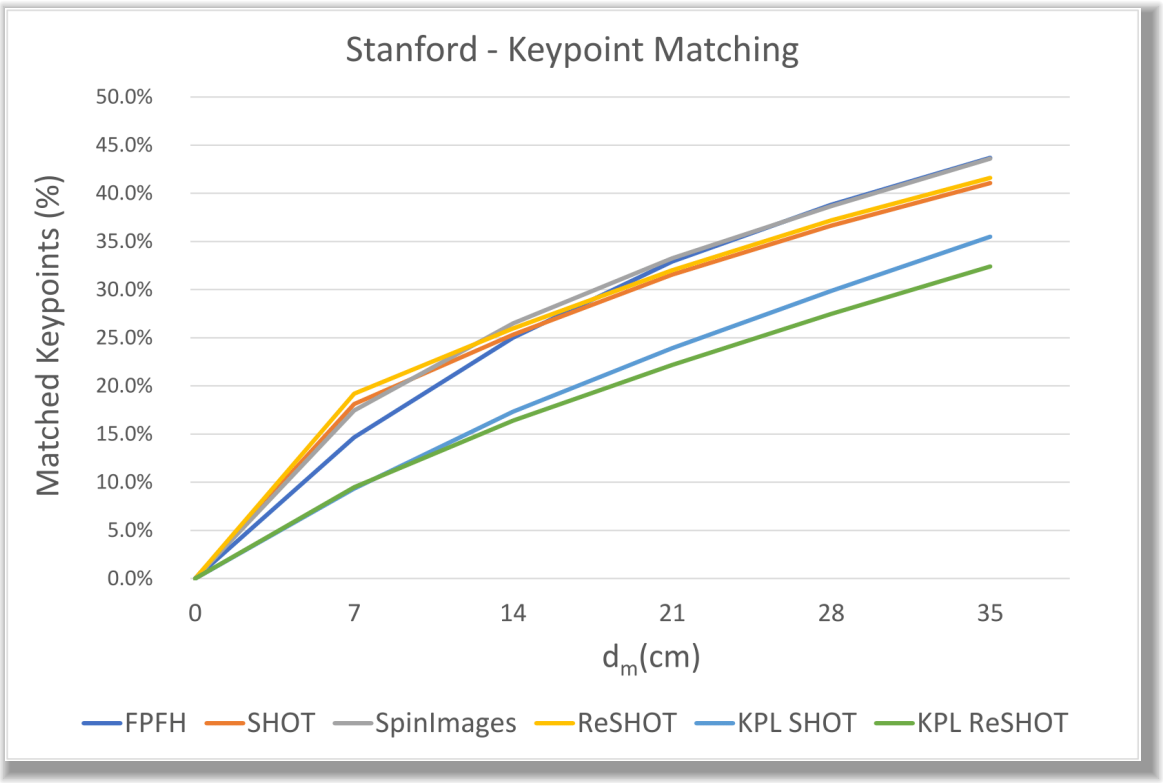

Figura 5.8: Percentuale di corrispondenze correttamente individuate su dataset Stanford dopo l'applicazione dei metodi proposti in funzione del parametro d_m

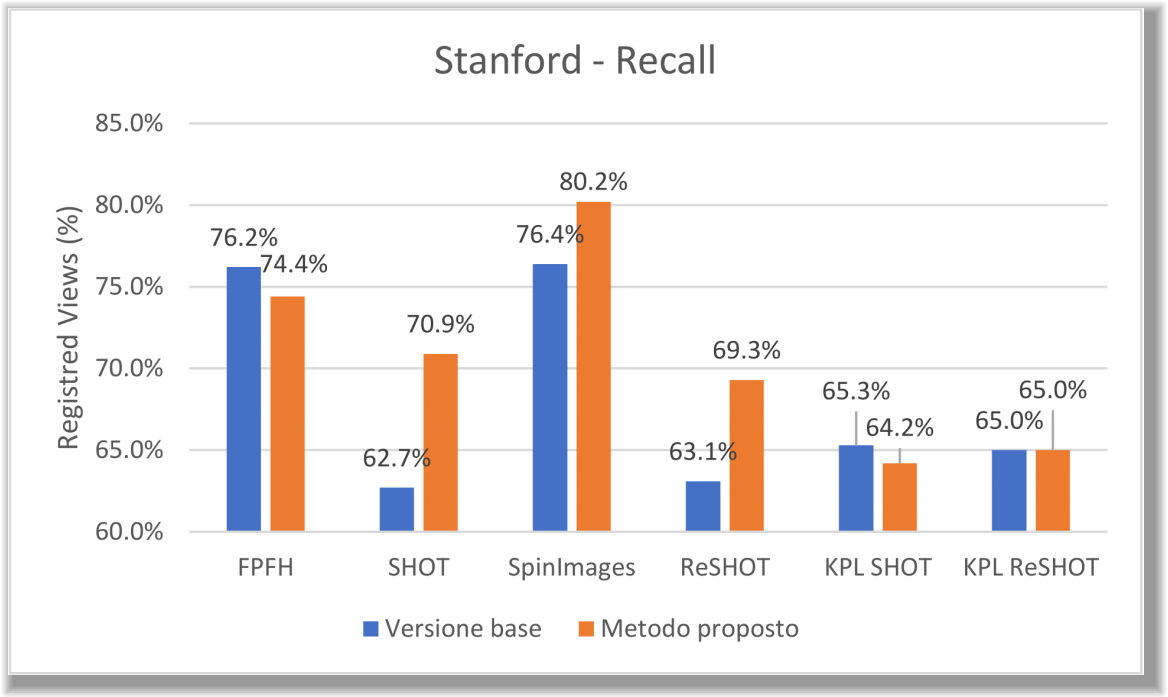

Figura 5.9: Percentuale di viste correttamente registrate su dataset Stanford prima e dopo l'applicazione dei metodi proposti

Per quanto riguarda i dati di 3DMatch presenti nei grafici di figura 5.10, 5.11 e 5.12, osserviamo delle percentuali più basse di match positivi, in particolare per quanto riguarda il descrittore SpinImages.

Il calo complessivo della performance è dovuta al minor numero di keypoint presi in considerazione in questa seconda fase di test, questa scelta è stata fatta per snellire i tempi di calcolo del test, un maggior numero di keypoint avrebbe migliorato complessivamente la performance, lasciando inalterata la differenza tra versione originale dell'algoritmo e metodo proposto.

L'ulteriore calo riscontrato nel solo descrittore SpinImages è probabilmente dovuto ad un support angle, implementato di default nel suo algoritmo, del tutto analogo a quello testato in questo lavoro di tesi su SHOT e ReSHOT che, nel caso del dataset Stanford View si è rivelato efficace, ma non in 3DMatch.

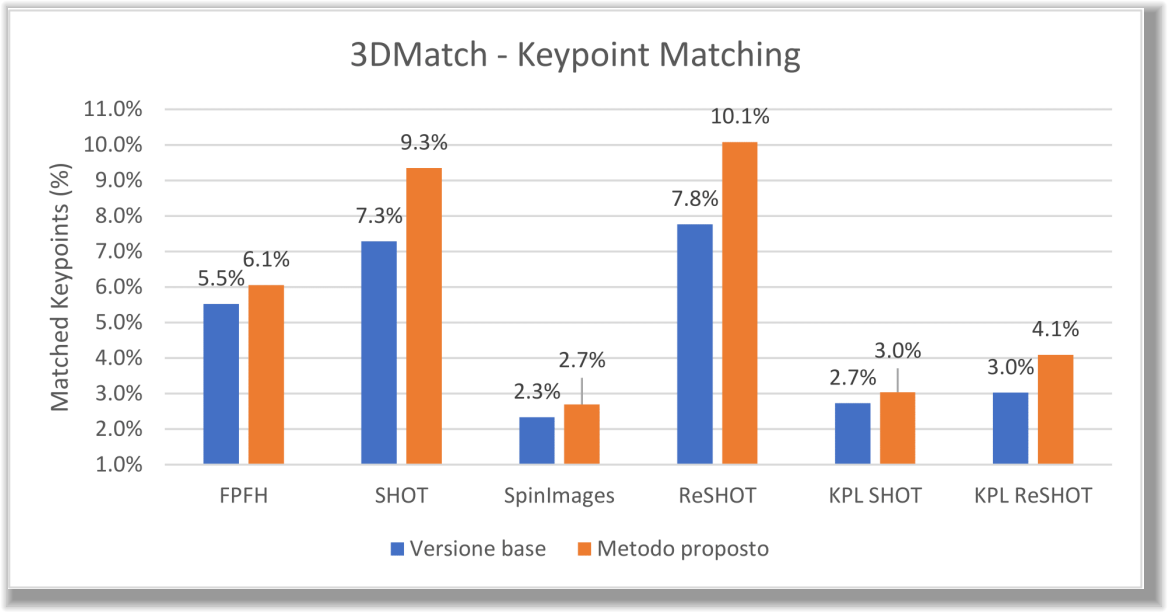

Figura 5.10: Percentuale di corrispondenze correttamente individuate su dataset 3DMatch prima e dopo l'applicazione dei metodi proposti

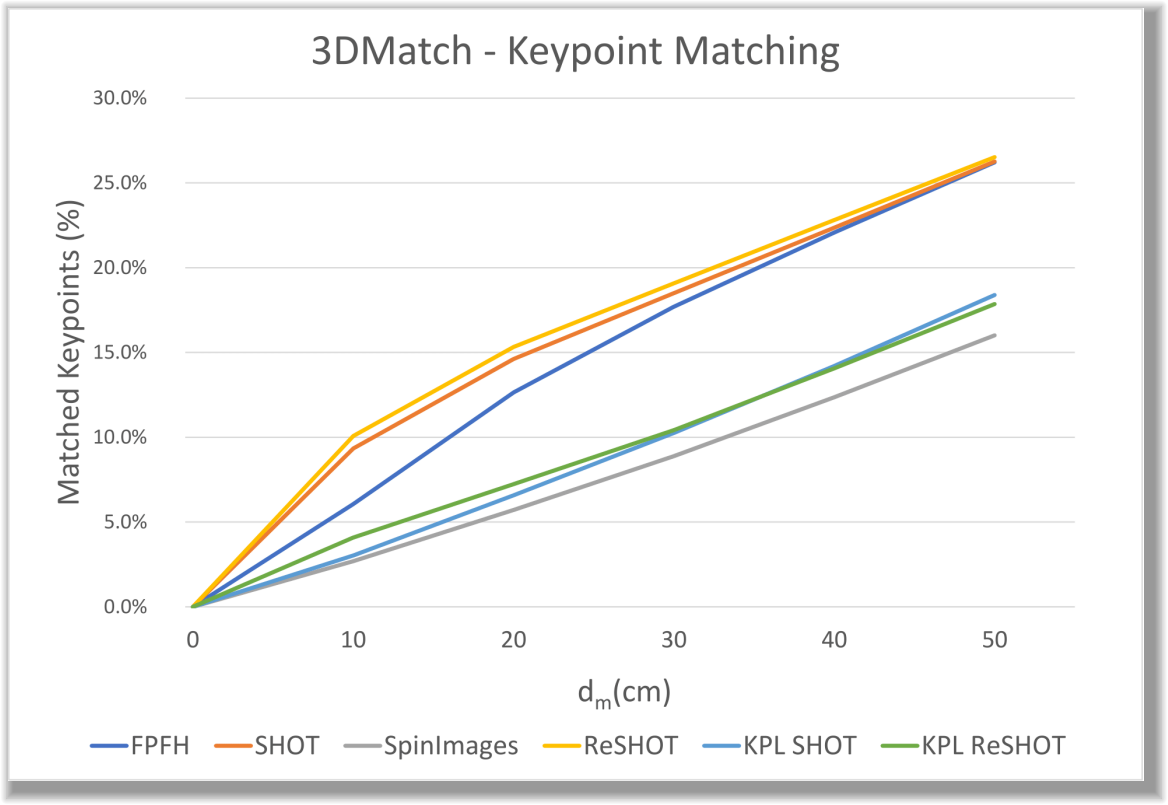

Figura 5.11: Percentuale di corrispondenze correttamente individuate su dataset 3DMatch dopo l'applicazione dei metodi proposti in funzione del parametro d_m

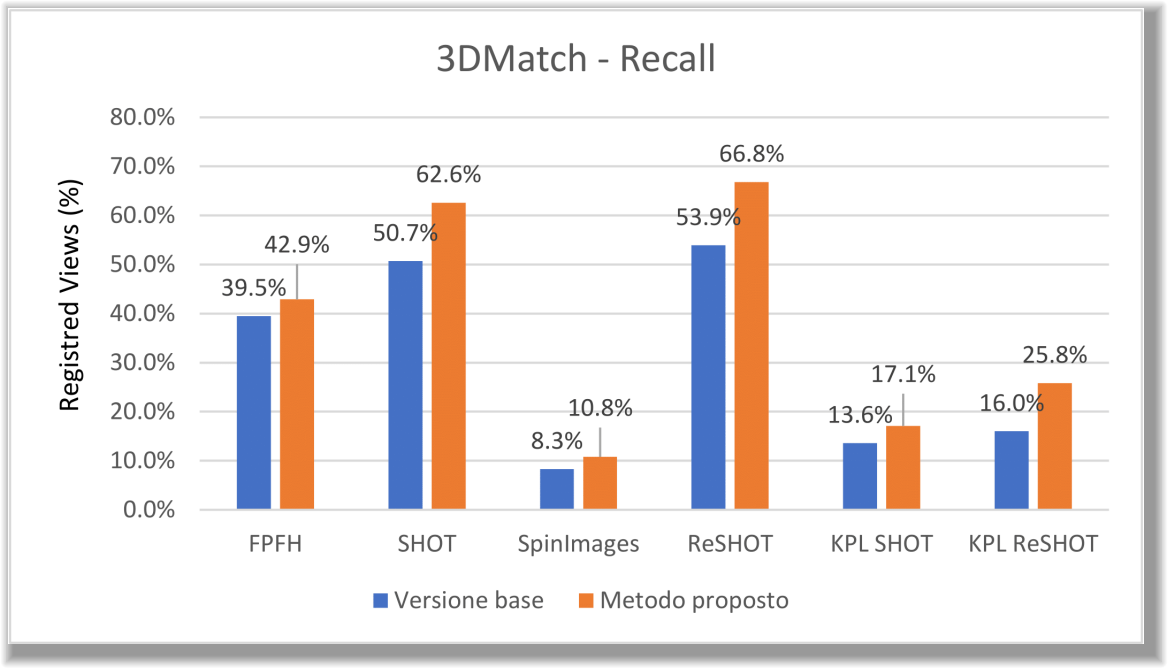

Figura 5.12: Percentuale di viste correttamente registrate su dataset 3DMatch prima e dopo l'applicazione dei metodi proposti

Infine notiamo che i dati dei test del dataset 3DMatch presenti nei grafici delle figure 5.13, 5.14 e 5.15, a cui sono state apportate rotazioni casuali, sono praticamente sovrapponibili a quelli della versione base del dataset, il che è perfettamente giustificato dal fatto che i descrittori presi in considerazione sono rotation invariant.

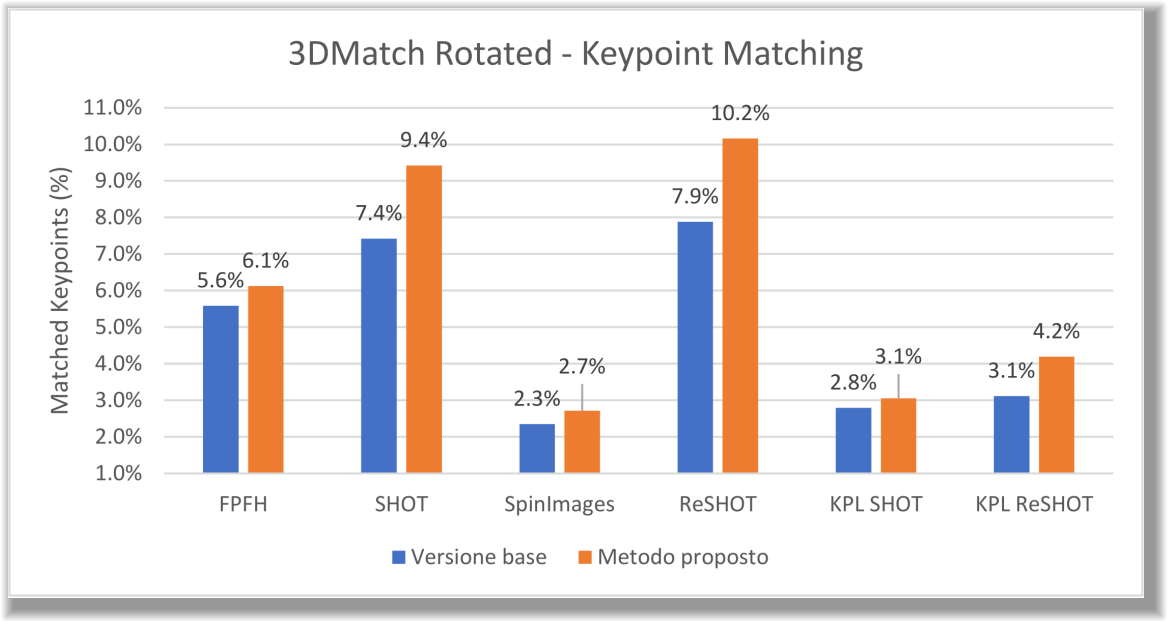

Figura 5.13: Percentuale di corrispondenze correttamente individuate su dataset 3DMatch (con aggiunta di rotazioni casuali) prima e dopo l'applicazione dei metodi proposti

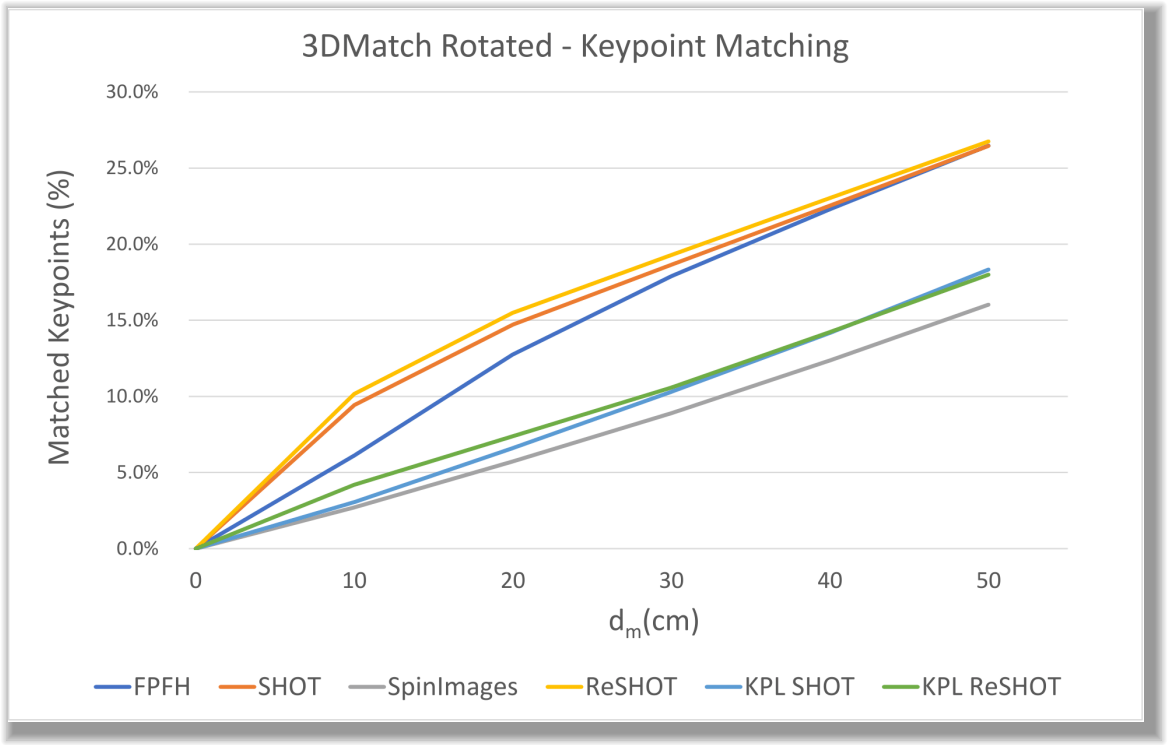

Figura 5.14: Percentuale di corrispondenze correttamente individuate su dataset 3DMatch (con aggiunta di rotazioni casuali) dopo l'applicazione dei metodi proposti in funzione del parametro d_m

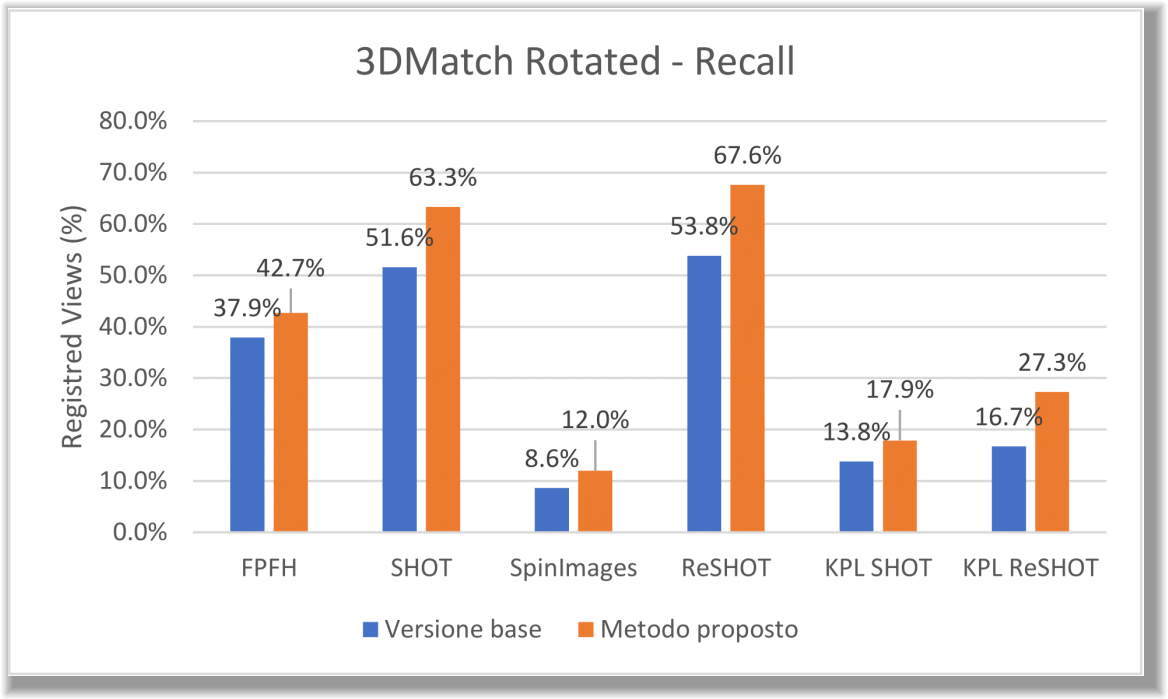

Figura 5.15: Percentuale di viste correttamente registrate su dataset 3DMatch (con aggiunta di rotazioni casuali) prima e dopo l'applicazione dei metodi proposti

Capitolo 6

Conclusioni e sviluppi futuri

In questo lavoro di tesi abbiamo affrontato la tematica dei descrittori 3D per point cloud. In particolare abbiamo valutato dei descrittori preesistenti quali SHOT, SpinImages ed FPFH e li abbiamo confrontati con dei metodi nuovi: ReSHOT, KPL-Descriptor SHOT e KPL-Descriptor ReSHOT.

Abbiamo effettuato un tuning dei raggi del supporto e dei support angle ed in seguito effettuato un ablation test per feature, Local Reference Frame e distanza nello spazio dei descrittori per capire quali singoli step hanno migliorato l'intera pipeline.

Ognuno dei test ha verificato che l'applicazione dei metodi proposti ha avuto un effetto migliorativo, riscontrando un aumento delle performance in termini di percentuali di match esatti e viste correttamente registrate.

Un possibile sviluppo di questo lavoro potrebbe essere il test sul support angle che, diversamente da quello a singola soglia testato in questo lavoro di tesi, escluda dei punti in base ad algoritmi euristici o comunque basati sulla correlazione tra partizione dell'istogramma e percentuali di match errati.

Si potrebbero differenziare le soglie o gli algoritmi in base ad un analisi preliminare del dataset. Questo escluderebbe l'apporto negativo di alcuni punti presenti a causa del rumore.

Si propone, inoltre, nella fase finale della pipeline, il test di altri operatori di distanza, peraltro già implementati nella libreria FLANN, utilizzata per la ricerca di un vicino in spazi multidimensionali.

Sarebbe anche interessante, verificare come le feature testate con successo in questo lavoro di tesi siano in grado di addestrare delle pipeline di feature matching basate su deep learning, facendo riferimento ad alcuni dei lavori qui citati che già sono orientati in questa direzione come [2] e [13].

Un'ultima considerazione è rivolta agli studi sul Local Reference Frame che, come si nota anche dai risultati dei nostri test, hanno un impatto molto forte sulle performance. Una algoritmo che permetta di stimare in modo affidabile e ripetibile la rotazione di un keypoint relativamente ad una superficie è un problema di difficile soluzione su cui è necessario concentrare ancora molte risorse.

Bibliografia

[1] I. Sipiran e B. Bustos, «Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes», *The Visual Computer*, vol. 27, n. 11, p. 963, 2011.

[2] S. Salti, F. Tombari, R. Spezialetti e L. Di Stefano, «Learning a descriptor-specific 3d keypoint detector», in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2318–2326.

[3] S. Salti, F. Tombari e L. Di Stefano, «A performance evaluation of 3d keypoint detectors», in *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, IEEE, 2011, pp. 236–243.

[4] X.-F. Hana, J. S. Jin, J. Xie, M.-J. Wang e W. Jiang, «A comprehensive review of 3D point cloud descriptors», 2018.

[5] D. G. Lowe, «Distinctive image features from scale-invariant keypoints», *International journal of computer vision*, vol. 60, n. 2, pp. 91–110, 2004.

[6] H. Bay, T. Tuytelaars e L. Van Gool, «Surf: Speeded up robust features», in *European conference on computer vision*, Springer, 2006, pp. 404–417.

[7] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan e N. M. Kwok, «A comprehensive performance evaluation of 3D local feature descriptors», *International Journal of Computer Vision*, vol. 116, n. 1, pp. 66–89, 2016.

[8] R. B. Rusu, N. Blodow e M. Beetz, «Fast point feature histograms (FPFH) for 3D registration», in *2009 IEEE International Conference on Robotics and Automation*, IEEE, 2009, pp. 3212–3217.

[9] R. Bro, E. Acar e T. G. Kolda, «Resolving the sign ambiguity in the singular value decomposition», 2, vol. 22, Wiley Online Library, 2008, pp. 135–140.

[10] A. E. Johnson e M. Hebert, «Using spin images for efficient object recognition in cluttered 3D scenes», 5, vol. 21, IEEE, 1999, pp. 433–449.

[11] J. H'roura, M. Roy, A. Mansouri, D. Mammass, P. Juillion, A. Bouzit e P. Méniel, «Salient spin images: A descriptor for 3D object recognition», in *International conference on image and signal processing*, Springer, 2018, pp. 233–242.

[12] A. G. Buch, D. Kraft e D. Odense, «Local point pair feature histogram for accurate 3D matching», BMVC, 2018.

[13] B. Drost, M. Ulrich, N. Navab e S. Ilic, «Model globally, match locally: Efficient and robust 3D object recognition», in *2010 IEEE computer society conference on computer vision and pattern recognition*, Ieee, 2010, pp. 998–1005.

[14] A. Petrelli e L. Di Stefano, «On the repeatability of the local reference frame for partial shape matching», in *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 2244–2251.

[15] ——, «A repeatable and efficient canonical reference for surface matching», in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, IEEE, 2012, pp. 403–410.

[16] S. Cousins e R. B. Rusu, «3D is here: point cloud library (pcl)», in *IEEE International Conference on Robotics and Automation, Shanghai (China)*, 2011.

[17] G. Turk e M. Levoy, «Zippered polygon meshes from range images», in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, 1994, pp. 311–318.

[18] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao e T. Funkhouser, «3dmatch: Learning local geometric descriptors from rgb-d reconstructions», in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1802–1811.

[19] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi e A. Fitzgibbon, «Scene coordinate regression forests for camera relocalization in RGB-D images», in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2930–2937.

[20] H. Deng, T. Birdal e S. Ilic, «Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors», in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 602–618.