UNIVERSITÀ DI PISA

Facoltà di Scienze
Matematiche Fisiche e Naturali

Corso di Laurea Specialistica
in Tecnologie Informatiche

Tesi di Laurea

Allineamento Automatico di Scansioni 3D

Candidato:

Francesco Tonarelli

Relatori:

Dr. Paolo Cignoni

Dr. Fabio Ganovelli

Controrelatore:

Prof.ssa Ornella Menchi

Anno Accademico 2008/09

Indice

Capitolo 1

Introduzione

Negli ultimi anni, i continui sviluppi e la crescente disponibilità dei dispositivi di scansione ha determinato un sempre più frequente impiego delle tecnologie di *3D scanning*. Queste consistono nell'acquisire la forma e il colore di oggetti reali per costruirne una rappresentazione digitale tridimensionale estremamente fedele ed accurata.

Il *3D scanning* è molto utilizzato in diversi ambiti. Ad esempio queste tecnologie trovano molte applicazioni nel contesto dei *beni culturali* (*cultural heritage*); le opere vengono acquisite per svariati motivi, dalla semplice archiviazione digitale alla riproduzione di copie, dall'impiego in musei digitali allo studio della loro conservazione o restauro. In archeologia l'acquisizione 3D dei reperti, spesso rinvenuti in diversi frammenti, è sfruttata per poi ricomporli in una rappresentazione digitale dell'oggetto originale. Altri ambiti in cui viene adoperato il *3D scanning* sono le classiche applicazioni industriali, come *reverse engineering*, *quality control* e *architecture design*.

A causa del crescente impiego delle tecnologie di *3D scanning*, si è fatta sempre più pressante l'esigenza di automatizzare il processo di acquisizione e digitalizzazione degli oggetti, noto come *acquisition pipeline*. Infatti una parte degli strumenti *software* e *hardware* usati in questo processo sono ancora semi-automatici e richiedono l'intervento dell'uomo, fatto che contribuisce a un sostanziale aumento dei tempi di sviluppo.

In particolare uno dei principali colli di bottiglia dell'intero processo di acquisizione sembra essere rappresentato dalla fase di allineamento (*alignment* o *registration*) delle singole scansioni. Il problema emerge poiché la superficie dell'oggetto, durante il processo di scansione, viene acquisita da più punti di vista e successivamente tutte le singole scansioni devono essere allineate, ossia portate in un unico coerente sistema di coordinate, per riprodurre l'oggetto completo.

La tesi concerne la realizzazione di un *framework* per l'allineamento automatico di scansioni 3D. Per affrontare il problema dell'allineamento si è scelto di utilizzare la classe di algoritmi che sfrutta *feature* della superficie per individuare i punti più rappresentativi e corrispondenti delle varie scansioni; successivamente viene eseguita una ricerca per trovare i punti corrispondenti sull'altra scansione. Dopo aver individuato due insiemi di punti corrispondenti sulle due superfici è possibile calcolare la trasformazione rigida che allinea le due scansioni.

Sebbene questo approccio sia concettualmente semplice, la realizzazione e il confronto pratico di algoritmi basati su differenti *feature* comporta delle difficoltà dovute principalmente alla presenza di molti parametri da impostare e gestire. In letteratura esistono molte proposte di procedure di allineamento che sfruttano l'approccio presentato, mancano tuttavia degli studi sugli effetti che la variazione dei parametri comporta in termini di prestazioni, ossia del numero di allineamenti corretti e della loro bontà. Allo stesso modo mancano degli studi comparativi sull'efficacia delle diverse *feature* proposte in letteratura; questo perchè gran parte delle implementazioni esistenti sono state concepite per funzionare con una *feature* specifica.

Un *framework* per l'allineamento automatico, generale e del tutto indipendente dalle *feature*, come quello proposto in questa tesi, permette di utilizzare *feature* diverse e condurre un'analisi approfondita sull'efficacia delle svariate *feature* presentate in letteratura. Inoltre il *framework* consente di

valutare in modo semplice e veloce gli effetti della variazione dei parametri.

Il *framework* è stato sottoposto a svariate verifiche per certificarne la correttezza e la consistenza sperimentale; dopodiché sono state implementate due tipi di *feature* basate sul calcolo della curvatura e sono state valutate sperimentalmente le loro prestazioni tramite il *framework* proposto. Infine sono stati condotti degli esperimenti per osservare l'impatto sulle prestazioni di alcune modifiche apportate al *framework*.

La tesi è organizzata come segue. Nel capitolo 2 si descrive il processo di acquisizione e le problematiche ad esso inerenti, soffermandosi in particolare sulla parte che riguarda l'allineamento di *range map* e illustrando i metodi di allineamento automatico proposti in letteratura. Nel capitolo 3 vengono descritte nel dettaglio il *framework* e le *feature* implementate. Il capitolo 4 presenta gli aspetti architetturali e il design del software, mentre nel capitolo 5 sono riportati i risultati dei test condotti per verificare la correttezza sperimentale del *framework* e la valutazione dell'efficacia delle *feature* implementate.

Capitolo 2

Stato dell'arte

2.1 Introduzione

In questo capitolo viene descritto lo stato dell'arte per quanto riguarda l'allineamento di range map. Inizieremo ripercorrendo tutte le fasi del *pipeline di acquisizione* di oggetti reali (Sezione 2.2), per poi soffermarci sulla fase di *coarse registration* (Sezione 2.3), che è quella di maggior interesse per gli scopi che la tesi si propone. Nella sezione 2.3.2 viene introdotto il concetto di *feature* e nella sezione 2.3.3 e seguenti, vengono presentate tutte le principali procedure di allineamento proposte in letteratura. La sezione 2.4 è interamente dedicata all'algoritmo RANSAC che verrà ampiamente utilizzato nel seguito del testo. Infine viene illustrato ICP, l'algoritmo utilizzato per la fase di *fine registration* (Sezione 2.5).

2.2 Pipeline di acquisizione

Per comprendere pienamente le tematiche e le difficoltà inerenti all'allineamento di range map è bene ripercorrere in modo dettagliato l'intera filiera di operazioni che consente di creare la rappresentazione digitale per la forma e il colore di un oggetto reale [Fig. 2.1]. Questo ci permetterà di introdur-

re la nomenclatura che verrà usata nel seguito del testo e anche di dare al lettore una migliore conoscenza delle entità e delle grandezze in gioco. L'intera sequenza di operazioni viene generalmente detta *pipeline di acquisizione* (*acquisition pipeline*) [Cur99].

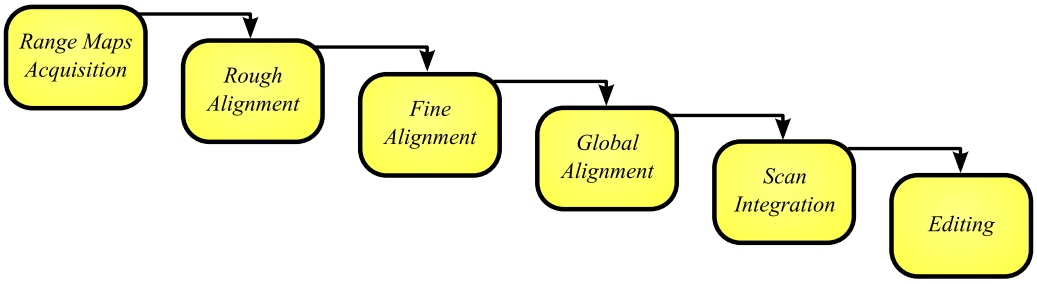


Figura 2.1: *Struttura del pipeline di acquisizione. Si effettuano molte scansioni (e fotografie) dell'oggetto; queste vengono allineate prima in modo approssimato e poi raffinate distribuendo l'errore su tutto il modello. Le scansioni allineate vengono fuse in un modello unico, viene aggiunto l'attributo colore e preparato ad hoc per il contesto applicativo.*

In questa esposizione vengono riportate le fasi più comuni; tuttavia in base al contesto in cui il modello deve essere usato, alcune parti del pipeline possono essere saltate o essere affrontate in altro ordine. Risulta altresì chiaro che alcune fasi sono invece indispensabili e inamovibili. Per informazioni dettagliate su tutti gli stadi del processo di acquisizione si consiglia la lettura di [BR02].

2.2.1 Range Map Acquisition

La prima fase, che potremmo definire *Range Map Acquisition* [Sco03], consiste nell'acquisire, mediante dispositivi di scansione, molte porzioni dell'oggetto, dette *range map*, che verranno poi ricomposte per formare il modello 3D. Una *range map* è definita da una griglia di punti nello spazio [Fig. 2.2a] e

rappresenta quella parte di superficie di un oggetto che è 'visibile' dal dispositivo di scansione. Al posto di *range map* si usano anche i termini *scansione* (*scan*), *vista* (*view*) e il termine *nuvola di punti* (*point cloud*), quest'ultimo facendo riferimento alla mancanza di informazione topologica. Poiché la maggior parte degli scanner forniscono le nuvole di punti strutturate in array bidimensionali, è possibile aggiungere la topologia per costituire una mesh di triangoli e ricavare la normale per ogni punto [Fig. 2.2b].

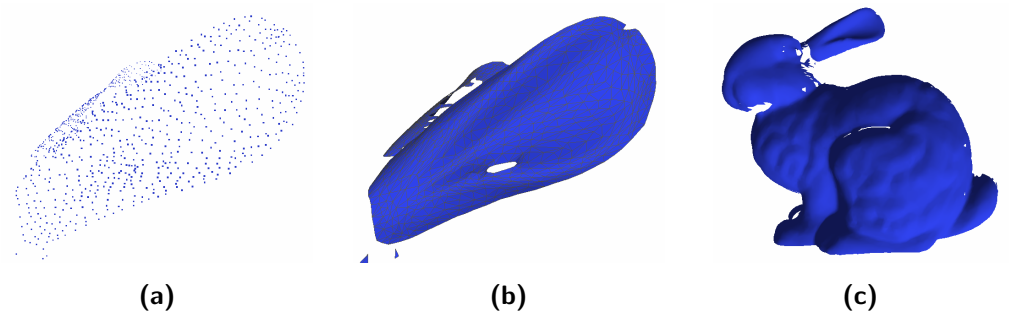


Figura 2.2: *Una range map. (a) dettaglio della point cloud. (b) Dettaglio della mesh di triangoli ricavata dalla point cloud. (c) La range map completa.*

Obiettivo di questo stadio del pipeline è riuscire ad acquisire il minor numero di range map necessarie per poter ricomporre integralmente l'oggetto scansionato. Questa fase è cruciale e deve essere condotta con metodo per avere il minor numero possibile di 'buchi' nel modello; per questo la stessa parte dell'oggetto viene ripresa più volte da differenti punti di vista e il numero di range map da gestire raggiunge facilmente l'ordine delle centinaia, con il conseguente aumento anche del tempo necessario al compimento dell'operazione [Fig. 2.3].

Non sempre è possibile riuscire a ricostruire un modello completo dalle scansioni raccolte, principalmente a causa della presenza nell'oggetto di parti che si auto-occludono e per l'impossibilità di accedere a taluni punti di vista.

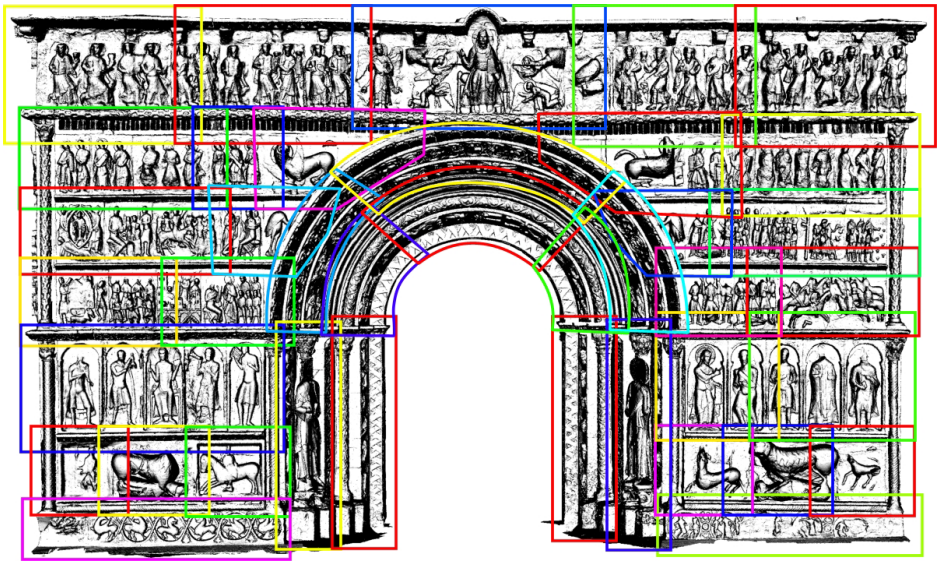


Figura 2.3: *Il piano della campagna di acquisizione della Portalada di Ripoll. Ogni rettangolo indica una sezione ripresa dalle 20 alle 90 volte, in base alle dimensioni e alla complessità del rilievo. In totale sono state acquisite 2212 range map che hanno richiesto 5 giorni di lavoro di un team di 4 persone.*

Fondamentale è che ogni range map si sovrapponga in parte[1] alle altre, così da consentire il passo succesivo del pipeline: l'allineamento.

2.2.2 Rough alignment

Scopo della fase di allineamento (*alignment* o *registration*) è quello portare tutte le range maps in un unico sistema di riferimento, così da poter ricomporre un unico modello 3D. Infatti, ogni range map è riferita a un sistema di coordinate locale al dispositivo di scansione; questo significa che, prima del-

[1]Perché sia possibile allineare le scansioni è necessario *almeno* una sovrapposizione del 20%.

l'allineamento, tutte le range maps appaiono disordinatamente sovrapposte una alle altre [Fig. 2.4a].

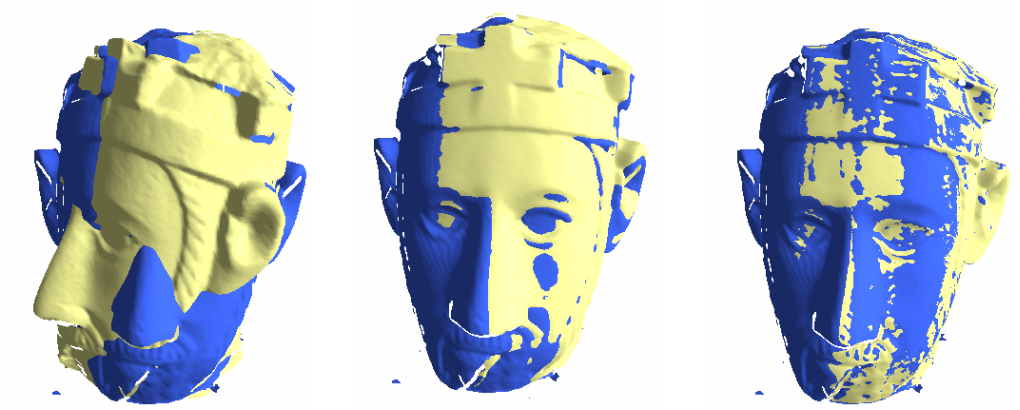


Figura 2.4: *Allineamento di due range maps: (a) range maps prima dell'allineamento, ognuna nel proprio sistema di coordinate locale; (b) range maps dopo la fase di Coarse Registration; (c) range maps dopo la fase di Fine Registration (ICP). Il maggior interscambio fra i colori è indice della maggior precisione dell'allineamento.*

Un possibile approccio al problema consiste nel dotare l'installazione di scansione di un dispositivo di tracking, così da tenere una precisa traccia di tutti gli spostamenti dello scanner e poter risolvere a priori il problema dell'allineamentro delle diverse viste. Il principale svantaggio di questo metodo è il costo dell'installazione. Spesso questa fase avviene con la partecipazione di un utente che indica due quadruple di punti corrispondenti sulle due range maps da allineare. Da queste due quadruple di punti il software può calcolare la trasformazione rigida che allinea le scansioni. Al momento della scrittura di questo testo, la fase di allineamento rappresenta ancora il collo di bottiglia dell'intero procedimento di acquisizione 3D, poiché qui viene spesa la maggiore parte del tempo e sono necessari, peraltro, molti interventi umani per

ottenere risultati davvero affidabili. Il problema, nella sua formulazione più generale, quella che non permette assunzioni sull'esistenza e sulla quantità di sovrapposizione (*overlap*) tra le scansioni, risulta essere di difficile risoluzione e la ricerca di procedure per l'allineamento automatico è un campo molto attivo. Le tecniche per l'allineamento automatico saranno descritte debitamente nella sezione 2.3; qui ci limiteremo ad anticipare che, qualunque sia la tecnica adottata, nessun procedimento automatico raggiunge un grado di precisione paragonabile a quella dei dispositivi di acquisizione. Per questo motivo, dopo che due range maps sono state allineate grossolanamente (*coarse/rough registration*) [Fig. 2.4b], c'è bisogno di un ulteriore passo di raffinamento (*fine registration*) [Fig. 2.4c].

2.2.3 Fine alignment

Per trovare un allineamento fine tra due viste che sono quasi allineate tra loro, il procedimento che è diventato lo standard de facto e che è stato più volte migliorato nel tempo è l'algoritmo *ICP* (*Iterative Closest Point*) [BM92, CM92]. ICP consiste di due fasi principali; nella prima si trovano delle coppie di punti corrispondenti nella zona di sovrapposizione delle due superfici, nella seconda si calcola la trasformazione che minimizza la distanza tra i punti. Le due fasi vengono reiterate fino a che non viene raggiunta una desiderata soglia di errore. Data l'importanza di questo procedimento, una spiegazione più dettagliata viene fornita successivamente nella sezione 2.5.

2.2.4 Global alignment

Generalmente il processo di allineamento dell'intero data set di range maps procede in maniera sequenziale. Le scansioni vengono allineate due alla volta (*pairwise alignment*); dopo ogni allineamento una nuova range map viene registrata con una scansione tra quelle precedentemente allineate e parzial-

mente sovrapponibile. Con questo modo di procedere l'errore di ciascun passo di pairwise alignment, si ripercuote accumulandosi sull'intero modello.

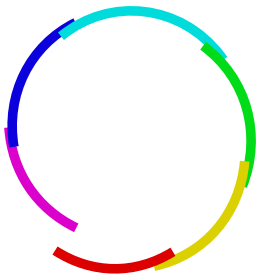


Figura 2.5: *L'immagine mostra come si distribuiscono le range maps di un vaso procedendo con pairwise alignments successivi. L'errore si accumula impedendo al vaso di "chiudersi"*

Per comprendere questo fenomeno è sufficiente pensare a quello che accade registrando con questo procedimento le diverse parti di un vaso [Fig. 2.5]; la prima e l'ultima scansione, che dovrebbero coincidere per 'chiudere' il perimetro del vaso, sono invece distanti tra loro a causa dell'errore che si è accumulato precedentemente. La fase di *allineamento globale* (*global alignment* o *multiview alignment*) del pipeline serve a risolvere questo problema. Sono stati proposti algoritmi che distribuiscono l'errore su tutte le diverse viste [Pul99, BSGL96], portando così a un allineamento di buona qualità complessiva. Tipicamente ci si avvale di un grafo in cui i nodi rappresentano le scansioni e gli archi uniscono ogni scansione con tutti i suoi vicini, vale a dire con tutte le viste per cui esiste una parziale sovrapposizione. Ad ogni arco è associata la matrice di trasformazione ricavata dal pairwise alignment delle due viste in questione; poiché gli allineamenti forniti dal pairwise alignment possono essere molto diversi per ciascun vicino [Fig. 2.6a, 2.6b], l'idea del multiview alignment è quella di ri-registrare ciascuna vista con tutti i suoi vicini simultaneamente [Fig. 2.6c].

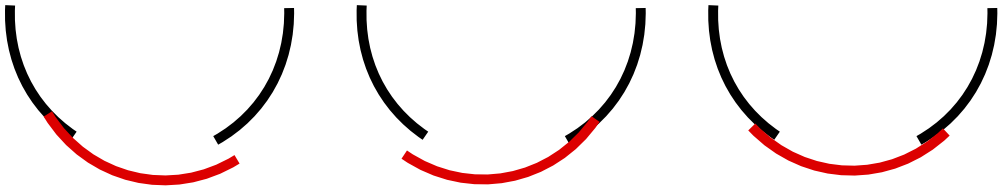


Figura 2.6: *Allineamento globale: l'immagine mostra come il pairwise alignment individui trasformazioni diverse per diversi 'vicini' e come l'errore diminuisca considerando tutti i vicini contemporaneamente. (a) Allineamento con la range map a sinistra; (b) allineamento con la range map a destra; (c) allineamento con entrambe le mesh contemporaneamente.*

Il passo che allinea una scansione con tutti i suoi vicini viene iterato più volte per ciascuna range map, fino al raggiungimento di una determinata soglia di errore. Per aumentare la velocità convergenza dell'algoritmo, il grafo viene costruito incrementalmente procedendo come descritto di seguito. Si parte dal grafo minimale di due viste che vengono allineate tra loro; si aggiunge una nuova range map al grafo e si procede ad allineare tra loro con il procedimento descritto tutte e tre le viste; si aggiunge un'altra range map al grafo etc...fino all'esaurimento dell'intero data set.

Naturalmente questo procedimento viene iterato più volte fino al raggiungimento di una determinata soglia di errore; inoltre, proprio per aumentare la convergenza dell'algoritmo, il grafo viene costruito incrementalmente, in altre parole ogni nuova vista viene registrata con una parte di modello già allineato in modo globale.

2.2.5 Scan integration

A questo punto tutte le scansioni devono essere 'fuse' in un unico modello 3D che non contenga facce e vertici ridondanti. L'obbiettivo di questa fase, chiamata *scan integration*, è quello di ricostruire la topologia e la geometria dell'oggetto scansionato a partire dai dati acquisiti tramite le scansioni. Il problema è complesso perché i dati acquisiti possono contenere rumore, alcune parti possono non essere state raggiunte durante la scansione e, nella formulazione più ampia del problema, non è dato sapere se la densità dei punti permette una corretta ricostruzione del modello. Su questo argomento esiste una vasta letteratura; qui verra data solo una descrizione sommaria dei pricipali metodi. I procedimenti *surface based* ricostruiscono la superficie basandosi su criteri locali per unire un punto con i suoi vicini; a volte possono essere utilizzate anche le informazioni di connettività fornite dalle range maps. Esempi di questa famiglia sono gli algoritmi di *stitching* o *zippering* [TL94, SL92] che tentano di 'cucire' tra loro le scansioni. Un altro algoritmo proposto in [BMR+99] crea una superficie interpolando le nuvole di punti delle diverse scansioni. I procedimenti volumetrici [WSI98, HSIW96] partizionano lo spazio intorno alle range maps in piccoli voxels, e ad ogni voxel associano una distanza, positiva o negativa, dalla superficie; dopodichè ricostruiscono la superficie definita dai voxel con distanza zero, tramite l'algoritmo *marching cube* [LC87] [Fig. 2.7].

2.2.6 Editing

Finalmente il nostro modello 3D è completo, ma ancora lontano dal poter essere utilizzato per una applicazione reale. Infatti il nostro modello è probabilmente composto da milioni di vertici, presenta diversi buchi e non contiene nessuna informazione sul colore, sul materiale, o texture. Qui inizia il lavoro della fase di *editing*; la mesh viene modificata e aggiustata ad hoc in base al

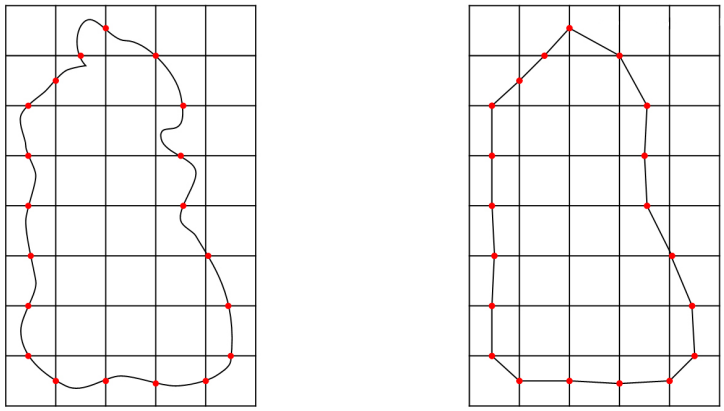
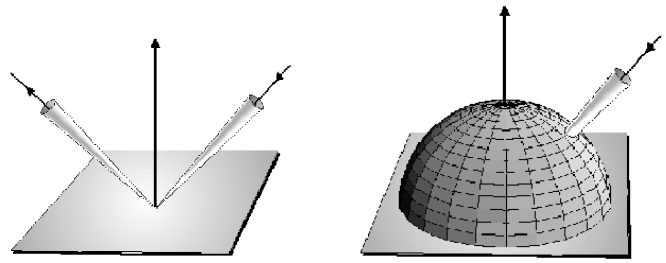


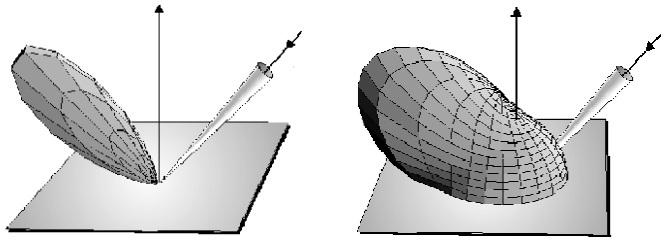
Figura 2.7: *Schema di funzionamento semplificato dell'algoritmo di ricostruzione Marching Cube. (a) La superficie viene inserita in una voxel grid. (b) Superficie ricostruita dall'algoritmo utilizzando i punti che intersecano la voxel grid.*

contesto applicativo a cui è destinata. Operazioni comuni in questa fase sono: *mesh semplification* [Gar99], per ridurre il numero di vertici e facce così da permettere il rendering in applicazioni interattive; *holes filling*, per chiudere i buchi della mesh, e l'operazione di *smoothing*, per modelli derivanti da scansioni con molto rumore. Un'altra grande area di interesse è quella relativa al *color recovery*. Le possibilità per applicare l'attributo colore a una mesh variano dall'applicare una texture, a definire per ogni punto della superficie la *Bidirectional Reflectance Distribution Function* (*BRDF*) [Fig. 2.8].

La prima soluzione applica alla mesh il colore così come catturato dalla camera; il risultato finale risente quindi delle condizioni di illuminazione dell'ambiente al momento dell'acquisizione delle texture. La seconda soluzione cerca invece di registrare quelle che sono le proprietà della superficie dell'oggetto ripreso, così da poterne poi effettuare un rendering che risulti realistico e accurato sotto qualsiasi condizione di illuminazione. Così come avviene per



(a) Caso speculare ideale **(b)** Caso diffusivo ideale



(c) Caso speculare reale **(d)** Caso diffusivo reale

Figura 2.8: *La BRDF è la funzione che descrive l'interazione tra luce e materia e che contribuisce a determinare il colore del modello. La riga superiore mostra le BRDF nei casi ideali. Sotto sono mostrati i comportamenti reali: riflessione con scattering e diffusione con highlight.*

le point clouds, anche le immagini delle diverse viste devono essere registrate sul modello; questo può essere fatto prima della fase di allineamento, se le immagini sono prese coerentemente con le range maps[2], oppure dopo la ricostruzione del modello, se le immagini sono state acquisite separatamente [Fig. 2.9]. Per tutti questi modi di operare è disponibile un'ampia bibliografia nel contesto della Computer Graphics e più precisamente per quanto riguarda la *image calibration*. Un testo molto completo a proposito è [RZ03].

[2]In questo caso l'informazione sul colore può anche essere usata nella fase di allineamento per risolvere alcune situazioni in cui le sole point clouds porterebbero ad allineamenti erronei.

(a) Fotografia **(b)** Modello 3D

Figura 2.9: *Color recovery: Modello 3D dove il colore è stato recuperato completamente tramite la costruzione di una texture map da più immagini fotografiche.*

2.3 Automatic coarse registration

Come già menzionato, l'obbiettivo della procedura di allineamento di range maps è portare tutte le scansioni, che sono state acquisite in un sistema di riferimento locale, in un sistema di coordinate globale unico per il modello. Questo avviene prendendo in considerazione due range maps alla volta e procedendo in due passi distinti; prima si trova un allineamento grezzo tra le due scansioni e successivamente questo viene raffinato fino a produrre un allineamento accurato. In altre parole, cercare un allineamento rozzo significa cercare di capire quali parti delle due viste si sovrappongono tra loro, mentre cercare un allineamento preciso significa minimizzare il divario tra queste parti. Alcune delle difficoltà più comuni che rendono la registrazione, in generale, un problema complicato da risolvere sono: la mancanza di informazioni circa le posizioni relative delle scansioni; l'impossibilità di fare assunzioni riguardo alla presenza di parti coincidenti (*overlap*), ed eventual-

mente la presenza di una superficie di sovrapposizione troppo esigua tra le scansioni, e infine ma non meno importante, la presenza di *rumore* (*noise*) nelle range maps acquisite.

2.3.1 Procedimento di base

Tutti gli algoritmi di *coarse registration* si basano sulla seguente considerazione: per risolvere il problema dell'allineamento è sufficente trovare due insiemi di punti corrispondenti sulle due viste. Nello spazio 3D sono sufficienti tre coppie di punti per determinare la trasformazione rigida cercata. Invece di tentare soluzioni brutali del problema che risulterebbe computazionalmente intrattabile, ossia cercare esaustivamente su tutti i punti di una mesh le corrispondenze con tutti i punti dell'altra mesh, tutti gli sforzi dei diversi approcci si concentrano nel estrarre da una mesh pochi punti 'rappresentativi', che sia facile individuare anche sull'altra mesh. Quest'ultima operazione che ricerca i punti che combaciano viene usualmente detta *matching*. Si capisce facilmente che questo modo di affrontare il problema comporta una forte riduzione della complessità. Le differenze tra i diversi approcci presenti in letteratura riguardano il modo in cui vengono individuati 'piccoli' insiemi di possibili punti candidati; come questi insiemi vengono esplorati per cercare i matching tra i punti, e come viene appurato se queste corrispondenze portano a una registrazione corretta.

2.3.2 Il concetto di Feature

La quasi totalità degli algoritmi presenti in letteratura utilizza un approccio *feature based* per ottenere un insieme significativo di punti su cui fare matching. Una *feature* (viene anche usato spesso il termine *descriptor*) è un valore che descrive, caratterizza, la porzione di superficie nell'intorno di un punto. Una feature ideale:

1. dovrebbe essere invariante rispetto alle trasformazioni rigide
2. dovrebbe avere un alto potere discriminatorio
3. dovrebbe essere robusta al rumore
4. dovrebbe essere efficiente da calcolare

La proprietà di invarianza alle trasformazioni rigide è fondamentale; ci garantisce che il valore della feature per lo stesso punto non cambia muovendo la mesh e, poiché non possiamo fare assunzioni sugli orientamenti reciproci delle viste, questa peculiarità risulta essere irrinunciabile. Il potere discriminatorio di una feature dovrebbe essere alto per poterci permettere di selezionare un numero piccolo di punti su cui eseguire il matching. Quando si parla di potere discriminatorio si intende che, su un vasto data set di range maps, restringendosi a considerare i valori della feature in un piccolo intervallo (ad esempio intorno allo zero) si ottengono in media solo pochi punti. La robustezza di una feature al rumore è anch'essa una proprietà importante, perché due acquisizioni della stessa superficie non sono mai identiche tra loro a causa dell'errore prodotto dallo specifico sistema di scansione e/o per le caratteristiche del materiale dell'oggetto acquisito. In contesti reali può capitare di acquisire con sistemi ottici degli oggetti di materiali particolarmente lucidi o traslucidi che forniscono range maps con molto rumore. Affinché sia possibile allineare con successo range maps rumorose è necessario, per due zone di superficie corrispondenti, ottenere due valori della feature molto vicini, anche in presenza di difformità dovute al rumore. La semplicità di calcolo è in sè per sè un attributo sempre desiderabile, tuttavia è bene precisare che il costo computazionale dell'algoritmo di allineamento è indipendente dalla complessità della feature utilizzata; il calcolo delle features per tutto il data set può essere visto come una fase di pre-processing che genera i dati di input dell'algoritmo di allineamento *feature based*.

Questo tipo di algoritmi utilizzano il valore delle features per svolgere la fase di matching, ossia per stabilire le corrispondenze tra i punti delle due range maps. Questo è possibile grazie alla proprietà di invarianza alle trasformazioni rigide della feature, da cui consegue che lo stesso punto possiede un valore del descrittore simile su entrambe le range maps. La fase di matching sfrutta proprio questa considerazione: per un punto p sulla mesh $\mathcal{M}_{fix}$ con valore del descrittore $\mathcal{F}(p)$, si cercano sulla mesh $\mathcal{M}_{mov}$ i k punti $p_1, .., p_k$ con valori dei descrittori $\mathcal{F}(p_1), .., \mathcal{F}(p_k)$ più vicini a $\mathcal{F}(p)$. Questo tipo di query, noto come *k-nearest neighbors search*, consente di ottenere delle corrispondenze ragionevoli tra i punti delle range maps con una buona probabilità.

Una caratteristica di tutte features, che determina un modo trasversale per classificarle, è la quantità di informazioni che servono a rappresentare la feature stessa. In questo senso possiamo discernere tra:

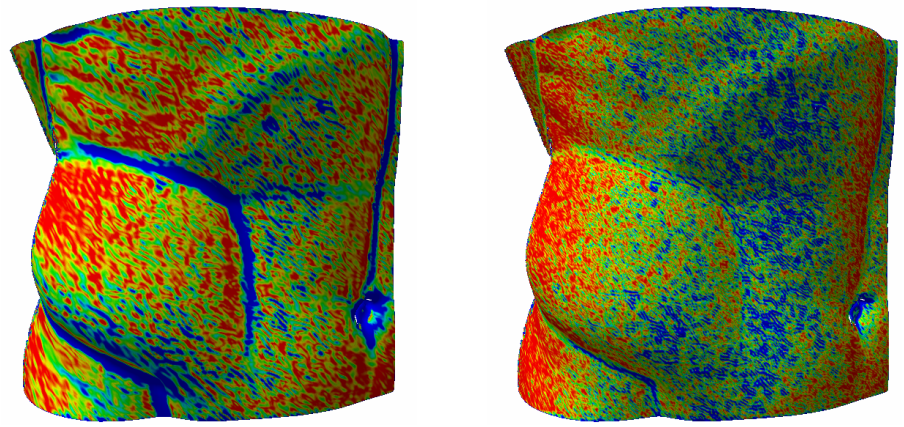
- ▷ *features mono-dimensionali*, che sono descritte da un valore scalare;
- ▷ *features multi-scala*, descritte da un vettore di scalari;
- ▷ *high-dimensional features*, descritte da strutture più complesse come array a 2/3 dimensioni.

Di seguito verrà usata questa classificazione per passare in rassegna le diverse features proposte in letteratura.

2.3.3 Features mono-dimensionali

La *curvatura*, in tutte le sue varianti, è senz'altro uno dei descrittori più usati [SLW00, LGB]. Il valore della curvatura indica quanto un oggetto si discosta dall'essere piatto in un determinato punto. Alcuni tipi di curvatura comunemente usati in questo contesto sono la *principale*, *gaussiana* e *media*. La curvatura è invariante rispetto alle trasformazioni geometriche ed è

calcolabile in modo efficienete ma, essendo una quantità legata a proprietà differenziali, è sensibile al rumore [Fig. 2.10]. Alcuni metodi per il calcolo della curvatura sono descritti in [BJ86].



(a) Poco noise **(b)** Molto noise

Figura 2.10: *Curvature assoluta visualizzata come colore su una mesh con/senza noise. Il blu corrisponde alle alte frequenze (pieghe, spigoli etc.), il rosso alle basse frequenze (parti pianegianti). Le figure evidenziano quanto la curvatura risenta del rumore.*

Oztireli e Basdogan [OB08] usano come descrittore la distanza d tra il punto sulla superficie q e il centro di massa dei suoi vicini, dove i vicini sono tutti i punti della superficie contenuti in una sfera di raggio r e centro q. Anche questa feature è molto semplice da calcolare ed è invariante alle trasformazioni rigide ma non è robusta al rumore e, cosa più importante, ha un potere discriminatorio molto basso. Gli autori hanno scelto appositamente un descrittore poco discriminante per mostrare come, applicando a un vasto insieme di valori un procedimento in più fasi (*histogram analysis*, *features clustering*, *outlier removal*, *center of mass extraction*), si possa comunque pervenire a un piccolo insieme di punti significativi.

Nel loro articolo, Pingi, Cignoni*et al.* [PFC+05], considerano la range map come una *mappa di altezze* (*height fields*) e definiscono per ogni punto p una quantità che dipende dai vicini contenuti in un kernel quadrato di $n \times n$. Il valore della feature non è altro che la varianza della distanza tra le normali dei punti del kernel; la selezione è effettuata scartando i valori a bassa e a alta varianza. Per ogni punto estratto viene eseguito il *matching* assumendo come punto coincidente quello con valore della feature più vicino; poiché la cardinalità dell'insieme dei punti estratti è ragionevolmente piccola, viene eseguita una ricerca esaustiva nello spazio delle triple di punti che appartengono all'insime, per ogni tripla viene calcolata e stimata la trasformazione associata. Questa feature è qualcosa di strettamente correlato alla curvatura; le zone con valori di varianza prossimi allo zero corrispondono a zone pianeggianti mentre i valori con alta varianza corrispondono ai 'picchi'. Zone con alta/bassa varianza possono anche essere generate sui bordi della range map o in aree che corrispondono al profilo di una parte occlusa; queste situazioni possono falsare la corrispondenza tra i punti e per questo è stato scelto di eliminare tali intervalli e considerare solo punti con varianza intermedia.

2.3.4 Features multi-scala

Di seguito sono riportati due algoritmi di allineamento che fanno uso di features rappresentate da un vettore di scalari.

Guskov *et al.* [LGB] selezionano due insiemi di punti dalle due range maps e effettuano una ricerca esaustiva in questo spazio delle soluzioni, verificando tutte le possibili trasformazioni e memorizzando via via quella che minizza la distanza tra le due viste. Dato che per calcolare la trasformazione rigida vengono utilizzate le normali e le direzioni principali di curvatura nei punti considerati[3], una singola coppia di punti è sufficiente a determinare

[3]Per un punto p le direzioni principali di curvatura k_1 e k_2, definiscono due vettori

la trasformazione cercata. L'articolo mostra come una selezione del set di punti basata su criteri *multi-scala* sia più efficace di una selezione casuale. Si dice rappresentazione su un'altra scala di una range map, una riduzione della range map a un minor numero di vertici oppure con i vertici *smoothed*. Un descrittore multi-scala dunque, è composto da un vettore n-dimensionale di valori, dove ogni dimensione corrisponde a una scala del modello. Nell'articolo si procede a costruire la rappresentazioni su più scale delle scansioni tramite un operatore di proiezione [AK04] che associa a ogni punto sulla superficie $\mathcal{S}$ il suo corrispettivo nella versione *smoothed* di $\mathcal{S}$. In seguito per ogni punto i e scala $j \geq 1$ viene calcolata la distanza tra normali di livelli adiacenti $d^j(i) = n_i^j \cdot (p_i^j - p_i^{j-1})$; di fatto questi valori costituiscono il descrittore multi-scala. Vengono selezionati come punti significativi solo quelli che risultano essere *neighborhood maximum*: un punto i di scala j è *neighborhood maximum* se il valore $d^j(i)$ è il più grande in un intorno di i, sia per la scala corrente j che per le scale adiacenti $j+1$,$j-1$. Intuitivamente questo significa che un punto viene selezionato se il suo descrittore 'spicca' sia rispetto ai vicini che alle scale vicine.

Si deve sottolineare il fatto che il modo di procedere dell'artricolo appena illustrato si scosta da quello più tipico degli algoritmi feature based descritto nella sezione 2.3.2, in quanto non viene effettuata nessuna fase di matching; invece ci si affida completamente alla bontà della selezione dei punti.

Un'altra procedura di allineamento che utilizza i *neighborhood maximum* e *minimum* è stato proposto in [LG05]. Da una scansione si estrae un sottoinsieme piuttosto contenuto di punti ricercando i *neighborhood maximum/-minimum*; per ognuno di qesti punti viene calcolato un valore invariante alle

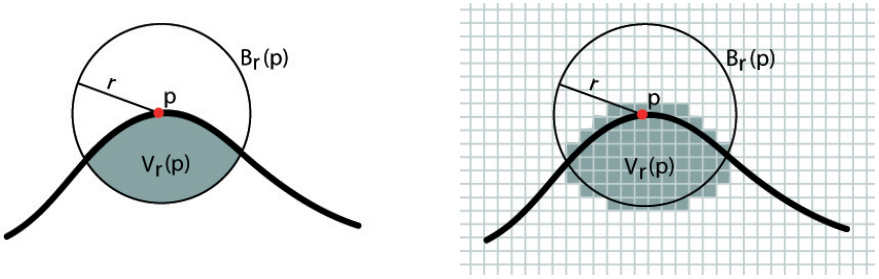
perpendicolari tra loro e giacenti sul piano ortogonale alla direzione della normale. Assieme alla normale al punto costiuiscono una terna di assi utilizzabile per calcolare la trasformazione rigida. Per maggiori informazioni sul calcolo di k_1 e k_2 si rimanda a [Rus04, Tau95a].

trasformazioni rigide, detto *signature*. Con ciascuno di questi valori viene effettuata una query del tipo *k-nearest neighbors* per ottenenere le plausibili corrispondenze sull'altra scansione. Anche in questo caso vengono provate esaustivamente tutte le trasformazioni possibili[4] e la migliore è restituita come risultato del pairwise alignment assime a una percentuale stimata di sovrapposizione. Gli autori continuano descrivendo anche una procedura di allineamento *multi-view*, ossia che permetta di registrare tutte le viste del data-set tra loro. Per un data-set di $\mathcal{N}$ viste vengono condotti $\mathcal{N}(\mathcal{N}-1)$ pairwise alignments come sopra descritto; le coppie di scansioni allineate vengono ordinate in base alla percentuale di overlap. A questo punto le coppie di viste vengono fuse (*merged*) con ICP seguendo l'ordine stabilito, ma vengono 'saltate' quelle coppie che già fanno parte di una componente *merged*. In altre parole si esegue una costruzione *greedy* dell'albero di copertura dell'insieme di scansioni.

Gelfand, Mitra *et al.* [GMGP05], utilizzano una feature multi-scala basata su proprietà integrali, come il volume, piuttosto che differenziali, come la curvatura. Features integrali garantiscono le proprietà di invarianza alle trasformazioni rigide e semplicità di calcolo ma sono meno sensibili al rumore rispetto alle feature differenziali. L'*integral volume descriptor* utilizzato misura, per ogni punto p di una mesh, il volume racchiuso tra una sfera centrata in p di raggio r e la parte interna della mesh [Fig. 2.11].

Calcolando la feature per diversi valori del raggio, si ottiene una feature multi-scala. Selezionando soltanto i punti della range map che hanno dei descrittori *'rari'* su tutti i livelli di scala è possibile ottenere un piccolo insieme di punti significativo, sparsi su tutta la superficie [Fig. 2.12]; per questi punti vengono travati i corrispondenti sull'altra vista effettuando una query dei k punti con descrittori piu vicini. Lo spazio delle triple di punti corrisponden-

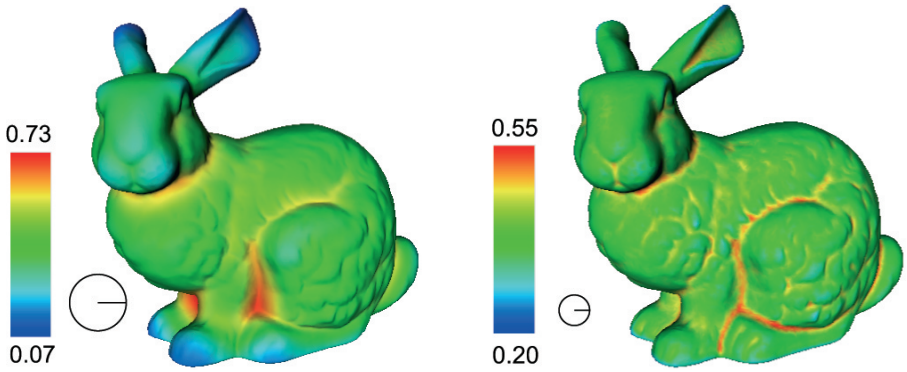
[4] In questo caso la trasformazione rigida è determinata usando una coppia di punti su ogni scansione e le relative normali.



(a) Valore ideale del descrittore **(b)** Valore del descrittore calcolato

Figura 2.11: *L'integral volume descriptor per il punto p misura il volume racchiuso tra una sfera centrata in p di raggio r e la parte interna della mesh. La zona in verde indica il valore del descrittore per il punto p. Il valore calcolato è un'approssimazione del valore ideale.*

ti è scandito esaustivamente utilizzando l'algoritmo *branch and bound*; solo le triple che soddisfano alcuni requisiti geometrici risultano accettabili e la trasformazione associata viene stimata.



(a) Valori del descrittore per un raggio d'integrazione grande **(b)** Valori del descrittore per un raggio d'integrazione piccolo

Figura 2.12: *Integral volume descriptor per due scale diverse. La scala più piccola ha zone rosse più ristrette; i punti colorati in rosso su entrambe le scale sono punti significativi poiché consistenti su tutte le scale.*

Huang, Flory *et al.* [HFG+] utilizzano un procedimento simile nel loro articolo sulla ricostruzione di oggetti fratturati, un problema molto usuale nel settore archeologico di cui la registrazione di superfici 3D è un sottoproblema. A partire dalla stesso *integral volume descriptor* citato in precedenza, vengono definite due ulteriori features, *sharpness* e *roughness*, che vengono usate per capire se una superficie appartiene a un lato frammentato dell'oggetto oppure no.

2.3.5 High-Dimensional Features

Tutte le features finora esaminate possono essere classificate come *low-dimensional features*, ossia sfruttano pochi valori scalari per rappresentare informazione. Johnson e Hebert [JH99] definiscono una *high-dimensional feature* che associa a un punto della superficie una particolare immagine 2D, chiamata *spin image*. Per eseguire il *matching* tra i punti delle range maps si vuole sfruttare la considerazione di fondo che punti corrispondenti avranno *spin images* simili. Il matching viene eseguito come illustrato di seguito. Le due meshes $\mathcal{M}_{fix}$ e $\mathcal{M}_{mov}$ vengono campionate in modo uniforme e per ogni vertice campionato su $\mathcal{M}_{fix}$ viene calcolata la *spin image* associata. Da $\mathcal{M}_{mov}$ vengono selezionati casualmente un centinaio di vertici e per ognuno di questi si procede al calcolo della spin image. Usando come metrica di similarità il *coefficiente di correlazione lineare*, tutte le spin images di $\mathcal{M}_{mov}$ vengono confrontate con tutte quelle di $\mathcal{M}_{fix}$, e ognuna di esse viene abbinata alla spin image che risulta più simile. Una volta ottenuti due insiemi di punti corrispondenti questi vengono divisi in piccoli gruppi, filtrati mediante la verifica di alcuni vincoli geometrici e infine usati per ottenere la trasformazione rigida. La *spin image* associata a un punto si ottiene a partire da una base locale al punto stesso, caratteristica che rende la spin image invariante alle trasformazioni rigide. La posizione di un qualunque punto della superficie

rispetto a un punto p di riferimento e alla sua normale n, può essere rappresentata con due coordinate cilindriche: la coordinata α stima la distanza tra il punto e la linea che passa per la normale n; la coordinata β è la distanza (con segno) perpendicolare al piano tangente definito da p e n. La porzione di spazio di forma cilindrica intorno al punto p è discretizzato in un array bidimensionale di valori; incrementando il valore associato alle coordinate (α, β) di un punto si ricava un istogramma che può facilmente essere letto come un'immagine 2D [Fig. 2.13].

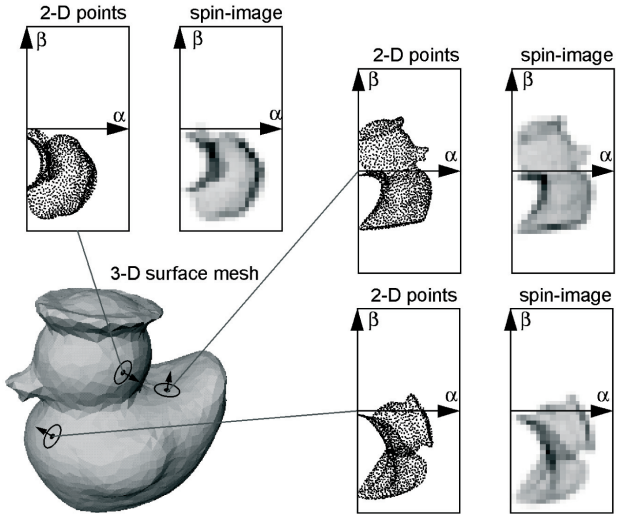


Figura 2.13: *Spin images calcolate per tre punti sul modello 3D. Per ogni punto sono mostrati in coordinate cilindriche i vertici che si trovano nella regione di supporto, oltre alla la spin image associata.*

Un'altra *high-dimensional feature* utilizzata nel contesto del*3D object recognition* è la feature *3D shape context* introdotta da Frome, Huber *et al.* [FHK+] come estensione a tre dimensioni della precedente *2D shape context* usata da Belongie *et al.* [BMP02]. Per calcolare la feature *3D shape context* relativa a un punto p con normale n, si procede in modo del tutto analogo al calcolo di una spin image. Le differenza sta nel fatto che viene costruito

un istogramma 3D, e non 2D; inoltre la regione intorno al punto p considerato è di forma sferica e non cilindrica [Fig. 2.14]. L'articolo utilizza le due features, spin images e 3D shape context, per eseguire il matching su scene 3D disordinate e con rumore; i risultati confermano che la feature 3D shape context si comporta meglio delle spin images in presenza di rumore.

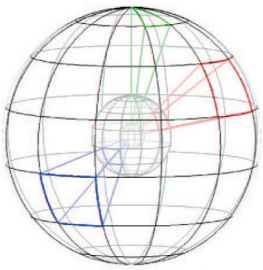


Figura 2.14: *La regione di supporto della feature 3DShapeContext. La regione di supporto è una sfera che viene discretizzata in bins; il numero di punti che si trovano all'interno dei bins determinano l'istogramma 3D che identifica la feature 3DShapeContext.*

In generale si può dire che le *high-dimensional features* codificano un'informazione più ricca rispetto alle *low-dimensional features* e per questo hanno un maggior potere discriminatorio; tuttavia, come visto in precedenza, le *low-dimensional features* possono essere efficacemente combinate ad approcci multi-scala che permettono di migliorarne decisamente l'efficacia discriminatoria riducendo il divario tra le due classi di features. D'altro canto le *high-dimensional features* hanno un costo computazionale molto più elevato rispetto alle *low-dimensional features* e questo ne limita fortemente l'uso in letteratura.

2.3.6 Feature Based vs Random Sampling Based

Tutti gli algoritmi di allineamento sinora esaminati possono essere classificati come *feature based*, in quanto sfruttano il valore di un descrittore per tentare di stabilire una corretta corrispondenza tra i punti delle range maps. Altri algoritmi, classificati come *random sampling based*, tentano di stabilire la corrispondenza tra punti delle range maps sorteggiando ripetutamente in modo casuale degli insiemi di punti, e affidandosi a considerazioni statistiche. Aiger, Mitra *et al.* [AMCO] descrivono uno di questi procedimenti che si basa sulla selezione di quattro punti co-planari e sull'algoritmo *RANSAC* (*RANdom SAmple Consensus*). Date due range maps da allineare P e Q con una percentuale di sovrapposizione f, vengono casualmente estratti tre punti da P e ne viene scelto un quarto tale da ottenere una base B di quattro punti (quasi) co-planari. Più i punti della base sono distanti tra loro più i risultati dell'allineamento sono stabili [GMO94] ma, per allineamenti parziali, aumentà la probabilità di scegliere punti fuori dalla regione di overlap e produrre quindi allineamenti erronei; per questo motivo la distanza tra i punti della base è una funzione di f. A questo punto si estrae da Q l'insieme $U \equiv \{U_1, U_2, .., U_s\}$ di tutti i sottoinsiemi di quattro punti che sono potenzialmente congruenti a B con una soglia di precisione δ. I punti dei sottoinsiemi in U, non solo devono essere co-planari, ma devono anche soddisfare alcune proprietà geometriche invarianti alle trasformazioni rigide (*ratios*) per avere la stessa configurazione spaziale della base B. Per ogni sottoinsieme $U_i \in U$ viene calcolata la trasformazione T_i associata con il metodo proposto da Horn [Hor87] e viene eseguita una fase di consenso, ossia viene applicata a P la trasformazione T_i e contati quanti punti di Q risultano più vicini della soglia δ a $T_i(P)$. Il punteggio ottenuto in questo modo viene associato alla trasformazione T_i. L'intero procedimento viene reiterato fino all'estrazione di L differenti basi da P e alla fine l'algoritmo restituisce la trasformazione

T_i che ha ottenuto il miglior consenso. Più alto è numero di iterazioni L e più alta è la probabilità di ottenere un buon risultato; tuttavia il numero di iterazioni richieste per raggiungere una certa probabilità di successo aumenta al diminuire della percentuale di overlap f. Lo schema generale del procedimento rispecchia quello dell'algoritmo *RANSAC*; data l'importanza di questo algoritmo, sia in letteratura che per il metodo da noi proposto nel seguito della tesi, viene illustrato in dettaglio nella prossima sezione 2.4.

2.4 RANdom SAmple Consensus

RANSAC, proposto nel 1981 da Fischler e Bolles [FB81], è un metodo iterativo per la stima dei parametri di un modello matematico a partire da un'insieme di dati osservati che contengono *noise* e *outliers*. Si tratta di un algoritmo non deterministico nel senso che non si ha la garanzia di ottenere *sempre* il risultato corretto, ma solo di ottenere un buon risultato *con una certa probabilità*. La probabilità di successo del RANSAC aumenta con il numero di iterazioni eseguite, tuttavia con esse aumenta anche il tempo di completamento della procedura. Per essere sicuri di ottenere un risultato ragionevole è possibile continuare a iterare fino che non viene trovata una soluzione che soddisfi una certa soglia di precisione; lo svantaggio di questo scenario è quello di non poter determinare a priori quanto tempo l'algoritmo impiegherà per fornire il risultato desiderato. Più comunemente viene fissato un limite al numero di iterazioni consentite, e quindi anche al tempo d'esecuzione, accontentandosi di ottenere un risultato significativo solo con una certa probabilità. Un vantaggio derivante dall'uso di questo metodo è che i risultati ottenuti sono buoni anche in presenza di numerosi *outliers* e di una grande quantità di *noise*.

2.4.1 L'algoritmo RANSAC

L'idea alla base del RANSAC è quella di selezionare casualmente un piccolo sottoinsieme di dati e utilizzare solo questi per ricavare una prima stima del modello. Se i dati selezionati sono tutti *inliers*, cioé aderiscono pienamente al modello, allora verrà stimato un modello ragionevolmete buono, altrimenti se anche solo un dato risulta appartenere agli *outliers*, la stima del modello sarà errata. A questo punto si passa verificare la bontà del modello rispetto a tutti i dati; se il numero di dati che si adatta al modello è inferiore a una certa soglia, la stima viene considerata errata e scartata, altrimenti il modello viene considerato accettabile e viene ricalcolato tenendo in considerazione tutti i dati in consenso. L'algoritmo termina quando viene trovato un modello con errore[5] inferiore a un valore fissato, oppure quando si raggiunge il limite massimo di iterazioni; in questo caso viene restituito il miglior modello trovato. Si possono applicare alcune variazioni a questo schema di base; una in particolare che ricorre spesso non ricalcola il modello per adattarlo a tutti i dati in consenso, ma si accontenta della stima iniziale. Questo permette di risparmiare tempo di calcolo a scapito di un risultato meno robusto al rumore. Alcuni parametri dell'algoritmo devono essere impostati in base alle caratteristiche specifiche del problema e dei dati, talvolta anche dopo delle analisi sperimentali; la necessità di risolvere parametri dipendenti dal contesto rappresenta un altro svantaggio dell'approccio RANSAC. A pagina 32 viene mostrato lo pseudo-codice di una versione piuttosto generica dell'algoritmo RANSAC.

[5]Invece di ragionare in termini di errore può essere più intuitivo ragionare in termini di numero di consensi ottenuti. Infatti maggiore è il consenso ottenuto dal modello e minore è l'errore commesso.

2.4.2 RANSAC per l'allineamento di range maps

RANSAC è stato spesso usato in computer grafica e in particolare per il problema dell'allineamento. In questo contesto i dati osservati da cui ricavare il modello sono le due range maps da allineare; il modello è la trasformazione rigida che registra correttamente le range maps, mentre gli *outliers* sono i punti che si trovano fuori dalla zona di sovrapposizione delle due scansioni. Passiamo ora a spiegare come le fasi precedentemente descritte si adattano al nostro contesto.

- ▷ All'iterazione i-esima viene scelta casualmente dalla mesh $\mathcal{M}_{fix}$ una base di n punti (almeno tre)
- ▷ Attraverso la procedura di matching si ottengono n punti corrispondenti sulla mesh $\mathcal{M}_{mov}$
- ▷ Viene stimata la trasformazione rigida T_i che meglio allinea i punti di $\mathcal{M}_{fix}$ con quelli di $\mathcal{M}_{mov}$
- ▷ La trasformazione viene verificata rispetto a tutti i punti di $\mathcal{M}_{fix}$ attraverso la procedura di consenso; si contano i punti di $\mathcal{M}_{fix}$ che si trovano entro una distanza δ da $\mathcal{M}_{mov}$.
- ▷ Se il risultato del consenso non supera una soglia Δ la trasformazione viene scartata perchè considerata errata e si procede con l'iterazione successiva.
- ▷ Se il risultato del consenso supera la soglia Δ la trasformazione è considerata corretta e, se il consenso ottenuto è più alto di tutti i precedenti, viene memorizzata come la migliore trasformazione possibile.
- ▷ Al raggiungimento del limite di iterazioni, oppure quando si trova una soluzione che superi un determinato limite di consenso, l'algorit-

mo termina e viene restituita la miglior trasformazione calcolata (o la trasformazione identica se l'algoritmo fallisce).

In letteratura sono state proposte alcune variazioni a questo schema con l'obiettivo di migliorarne le prestazioni; infatti la fase di consenso può richiedere non poco tempo se le scansioni hanno molti punti. Irani e Raghavan [IR96] hanno proposto una diversa procedura di consenso organizzata in due fasi: prima si testa solamente un numero costante di punti selezionati casualmente da $\mathcal{M}_{fix}$ e solo se una frazione significativa di questi risulta essere in consenso si passa a testare anche tutti i punti rimanenti. Lo schema del RANSAC sopra illustrato costituirà lo scheletro dell'algoritmo da noi implementato; è stato scelto non solo per la sua robustezza al rumore, ma soprattutto per la sua flessibilità ed estendibilità. Infatti la struttura descritta è totalmente indipendente dall'implementazione delle sue parti, ossia il *matching*, il *consenso*, il *calcolo della trasformazione* e la *selezione delle basi di punti*. Questa importante caratteristica ci offre la possibilità di potenziare e raffinare ogni aspetto dell'algoritmo di allineamento così da ottenere sia migliori risultati che migliori performance.

2.5 Fine registration

Come già detto in precedenza le range maps allineate in modo approssimato vengono portate a combaciare esattamente applicando l'algoritmo *Iterative Closest Point* (*ICP*).

L'algoritmo è stato proposto originariamente da Besl e McKay nel 1992 [BM92] e richiede che le due range maps siano tra loro quasi allineate per produrre un risultato corretto. Lo scopo di ICP è quello di trovare la trasformazione rigida T che meglio allinea una nuvola di punti $\mathcal{S}$ a un modello geometrico $\mathcal{M}$. Il processo di allineamento cerca di minimizzare la *media*

Algorithm 1 RANSAC in pseudo codice

data - a set of observed points

n - the minimum number of data required to fit the model

k - the maximum number of iterations allowed in the algorithm

δ - a threshold value for determining when a datum fits a model

Δ - number of close data required to assert that model fits well

$bestModel \leftarrow nil$, $bestConsensusSet \leftarrow \emptyset$, $bestError \leftarrow \infty$

for $iterations = 0$ to k **do**

  $maybeInliers \leftarrow n$ randomly selected values from $data$

  $maybeModel \leftarrow$ model parameters fitted to $maybeInliers$

  $consensusSet \leftarrow maybeInliers$

  for all $p \subseteq data \wedge p \nsubseteq maybeInliers$ **do**

    if p fits $maybeModel$ with an error smaller than δ **then**

      add p to $consensusSet$

    end if

  end for

  if $\mid consensusSet \mid > \Delta$ **then**

    $betterModel \leftarrow$ model fitted to all points in $consensusSet$

    $\epsilon \leftarrow$ a measure of how well $betterModel$ fits these points

    if $\epsilon < bestError$ **then**

      $bestModel \leftarrow betterModel$

      $bestConsensusSet \leftarrow consensusSet$

      $bestError \leftarrow \epsilon$

    end if

  end if

end for

return $bestModel$, $bestConsensusSet$, $bestError$

dei quadrati delle distanze (*mean square error*) tra i punti di $\mathcal{S}$ e quelli a loro più vicini sul modello $\mathcal{M}$. All'iterazione i-esima l'algoritmo seleziona opportunamente n punti da $\mathcal{S}$ e cerca i punti a questi più vicini sul modello $\mathcal{M}$, assumendoli come punti corrispondenti [Fig. 2.15]; dopodiché si ricava la trasformazione T_i che minimizza il *mean square error* tra i punti di $\mathcal{S}$ e quelli di $\mathcal{M}$; si applica T_i a $\mathcal{S}$ e si itera il procedimento fino al raggiungimento di un criterio di convergenza. Besl e McKay hanno dimostrato che il metodo converge monotonicamente a un minimo locale; tuttavia non c'è nessuna garanzia che ICP converga sempre al minimo globale, ossia alla soluzione corretta, ed è per evitare questa sfortunata circostanza che viene richiesto un buon allineamento di partenza [JNHL04]. Dal punto di vista computazionale ICP risulta essere un algoritmo efficiente; la sua complessità nel caso medio è spesso di $O(n \log n)$, dove n è il numero di punti scelti su $\mathcal{S}$. Una descrizione dell'algoritmo viene fornita a pagina 35.

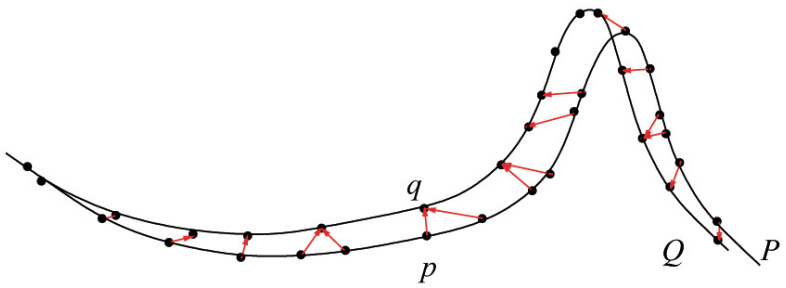


Figura 2.15: *Matching dell'algoritmo ICP: i punti sulla superficie P sono messi in relazione con i punti più vicini sulla superficie Q. Dato che le range maps sono già allineate in modo approssimato, l'algoritmo assume che questi punti siano effettivamente corrispondenti.*

2.5.1 Varianti di ICP

Diverse modifiche e miglioramenti sono state apportate a ICP nel tempo. Zang [Zha94] ha esteso ICP introducendo statistiche robuste e un meccanismo di soglie adattive che permettono di gestire *outliers* e occlusioni. Masuda e Yokoya [MY95] usano ICP con una diversa metrica per l'errore che risulta essere più robusta quando i modelli si sovrappongono solo parzialmente. Chen e Medioni [CM92] hanno perfezionato la fase di matching utilizzando un diverso criterio che porta a una selezione di punti coincidenti più significativa e conseguentemente a una più rapida convergenza dell'algoritmo. Le corrispondenze sono stabilite proiettando i punti di $\mathcal{S}$ su $\mathcal{M}$ secondo le direzioni delle normali, piuttosto che selezionando semplicemente i punti più vicini. Dorai *et al.* [DWJ97] hanno esteso il metodo di Chen e Medioni introducendo un sistema di pesi. Sharp *et al.* [SLW00] hanno proposto una variazione di ICP che utilizza l'informazione di un certo numero k di features invarianti rispetto le trasformazioni rigide per trovare migliori corrispondenze tra i punti di $\mathcal{S}$ e quelli di $\mathcal{M}$. I risultati ottenuti mostrano che questa tecnica porta a una più veloce convergenza dell'algoritmo e, soprattutto, porta l'algoritmo a convergere verso il minimo globale da un più vasto insieme di posizioni iniziali. A ogni punto estratto da $\mathcal{S}$ viene associato il punto di $\mathcal{M}$ che risulta essere più vicino in uno spazio a $3 + k$ dimensioni. Le prime tre dimensioni sono date dalla posizione del punto, le altre k dimensioni sono date dalle features; la distanza si misura in accordo alla norma L_2 e il contributo delle features può essere opportunamente pesato rispetto al contributo della posizione.

Algorithm 2 ICP (*Iterative Closest Point*)

Sia $\mathcal{S}$ la scansione da allineare e sia $\mathcal{M}$ il modello a cui allineare.
Sia $\|s - m\|$ la distanza tra il punto $s \in \mathcal{S}$ e $m \in \mathcal{M}$.
Sia $CP(s_i, \mathcal{M})$ il punto di $\mathcal{M}$ più vicino al punto $s_i \in \mathcal{S}$.
Sia T_0 la trasformazione rigida di partenza.

- ▷ Ripeti per $k = 1 \dots k_{max}$ o finché non viene soddisfatto un criterio di convergenza:
  - ▷ Seleziona da $\mathcal{S}$ un insieme di punti $N_s = \{s_1, \dots, s_n\}$
  - ▷ Costruisci l'insieme di punti corrispondenti:

$$\mathcal{C} = \cup_{i=1}^{n} \{(T_{k-1}(s_i), CP(T_{k-1}(s_i), \mathcal{M}))\}$$

  - ▷ Calcola la nuova trasformazione T_k che minimizza la media dei quadrati delle distanze tra le coppie di punti in $\mathcal{C}$

Capitolo 3

Il framework di allineamento

3.1 Introduzione

In questo capitolo viene descritto nel dettaglio il framework di allineamento automatico di range map da noi realizzato. Il framework in questione rientra nella famiglia dei procedimenti *feature based* descritti nella sezione 2.3.2, tuttavia il framework non è legato ad una feature particolare; al contrario, la sua struttura astrae completamente dalle caratteristiche delle molteplici feature presentate nelle sezioni 2.3.3 e seguenti. Il framework di allineamento è pertanto in grado di funzionare con diverse feature; la scelta e la realizzazione della specifica feature da utilizzare è lasciata all'utente, che fornirà queste informazioni alla procedura come *input* assieme alle range map.

Il capitolo inizia con una descrizione ad alto livello della struttura del framework e delle parti che lo compongono (sezione 3.2). Viene discussa in dettaglio la rappresentazione del concetto di feature (sezione 3.3) e, in particolare, vengono analizzate le feature implementate (sezione 3.4). Successivamente viene presentato il funzionamento dettagliato delle diverse parti del framework di allineamento: la selezione delle basi di punti (sezione 3.5), la procedura di matching (sezione 3.6), il calcolo delle trasformazioni rigide e il loro ordinamento (sezione 3.7), e infine la procedura di consenso (sezione

3.8).

3.2 Organizzazione ad alto livello

Ad alto livello il framework è strutturato come RANSAC (Sezione 2.4). Si è già illustrato nella sezione 2.4.2 come RANSAC possa essere utilizzato per realizzare un algoritmo di allineamento automatico. Usando RANSAC come schema di base del nostro metodo di allineamento facciamo nostre alcune importanti proprietà, come la robustezza al rumore e l'opportunità di variare la probabilità di successo con le iterazioni, ma soprattutto otteniamo un'infrastruttura del tutto indipendente dalla nozione di feature e abbastanza generica da permetterci di definire separatamente tutte le parti più rilevanti del processo di allineamento. Un'importante conseguenza di questa organizzazione del framework consiste nell'avere a disposizione un solido metodo di allineamento con cui analizzare l'efficacia delle molte feature proposte in letteratura (Sezioni 2.3.3, 2.3.4, 2.3.5); infatti benchè esista un'ampia proposta di feature tra gli articoli di settore, non sono molti quelli che confrontano l'efficacia delle diverse feature utilizzate.

3.2.1 Parti costituenti

La figura 3.1 a pagina 38 mostra uno schema della struttura del framework. L'organizzazione è conforme a quanto descritto nella sezione 2.4.2 a proposito dell'algoritmo RANSAC. Lo schema mette in risalto come il corpo del procedimento di allineamento sia costituito da quattro parti, racchiuse in forme di colore celeste, realizzabili ognuna separatamente dal resto della struttura. Lo schema mette anche in evidenza un ciclo interno al RANSAC con cui vengono verificati tutti i possibili allineamenti candidati per ogni iterazione.

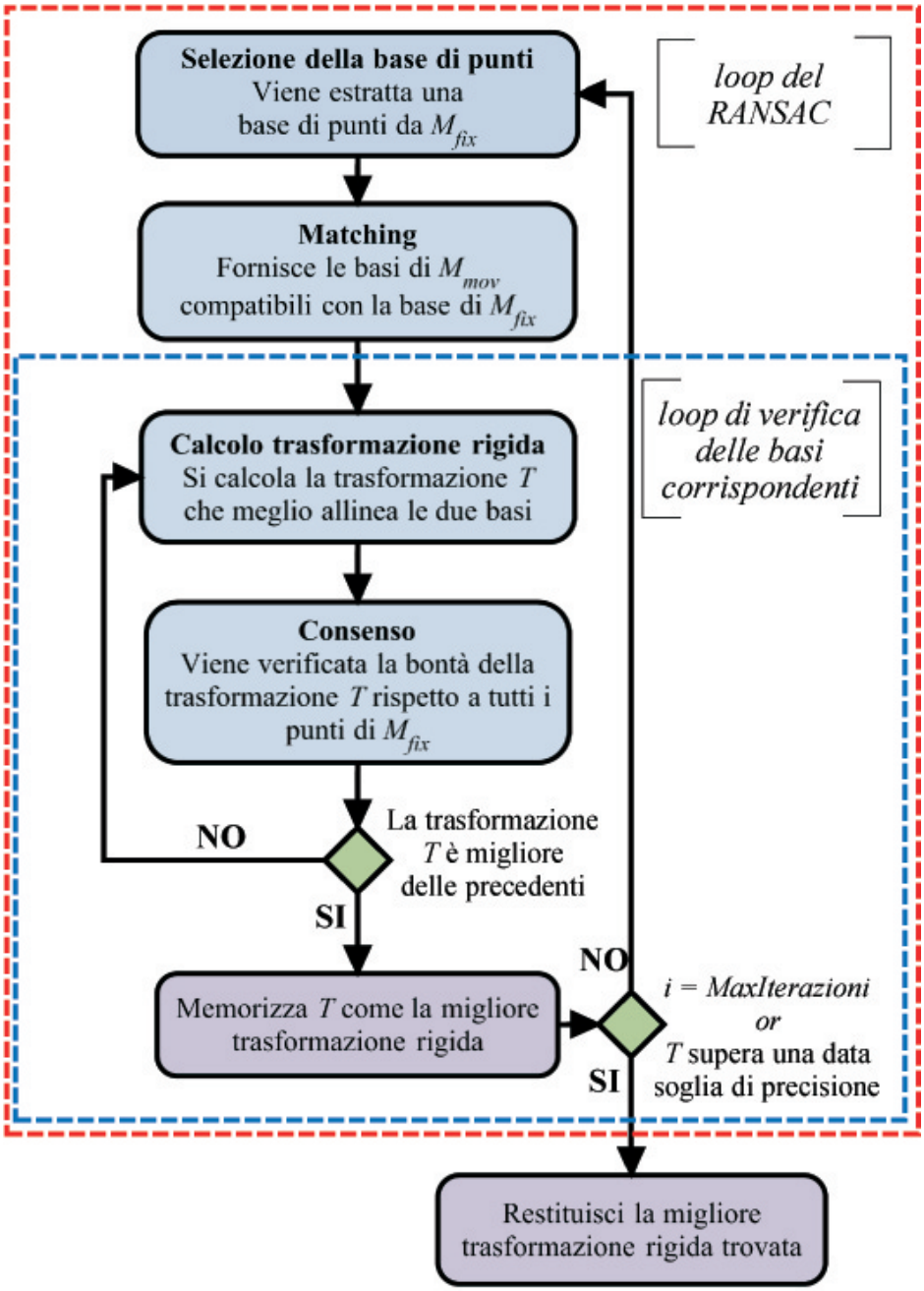


Figura 3.1: *Schema del framework. La struttura è conforme al RANSAC; i riquadri di colore celeste mostrano le funzioni realizzabili separatamente dal resto. Il riquadro rosso tratteggiato contiene le azioni svolte a ogni iterazione del RANSAC, mentre il riquadro blu tratteggiato le azioni svolte per verificare tutti i plausibili allineamenti di una iterazione del RANSAC.*

Le parti da definire per realizzare il framework di allineamento sono le seguenti:

- ▷ *Selezione delle basi di punti* dalla mesh $\mathcal{M}_{fix}$ (sezione 3.5).
- ▷ *Matching*, per ottenere le plausibili basi coincidenti sulla mesh $\mathcal{M}_{mov}$ (sezione 3.6).
- ▷ *Calcolo della trasformazione rigida* che allinea le due basi di punti (sezione 3.7).
- ▷ *Consenso*, per verificare la bontà della trasformazione considerata (sezione 3.8).

Oltre a questi elementi, che verrano discussi in dettaglio nelle sezioni indicate, è importante dare una rappresentazione del concetto di feature. Questa deve possedere alcune caratteristiche fondamentali che ne permettano l'utilizzo da parte del framework, ma deve astrarre dai dettagli, così da permettere l'implementazione di molteplici feature esistenti in letteratura. Poiché la feature da impiegare per la registrazione è un'informazione basilare da fornire alla procedura di allineamento, inizieremo proprio dalla rappresentazione della feature la discussione degli elementi che costituiscono il framework. Contestualmente alla descrizione di ogni parte del framework saranno talvolta menzionati i valori di default assegnati ai diversi parametri; per un elenco più compatto e completo si rimanda alla apposita sezione 5.2.

3.3 Rappresentazione del concetto di feature

Tutte le feature devono rispettare una specifica interfaccia che consente al framework di utilizzarle indipendentemente dalla loro specifica natura. Possiamo pensare che una feature sia composta da un insieme di *attributi* e da un'insieme di *funzionalità*, elencati di seguito.

- ▷ *Attributi*
  - ▷ *Posizione*, che individua il punto nello spazio 3D a cui la feature si riferisce.
  - ▷ *Normale*, che indica la normale al punto a cui la feature si riferisce.
  - ▷ *Descrizione*, un vettore di n scalari (uno o più) che contiene il valore del descrittore.
- ▷ *Funzionalità*
  - ▷ Calcolare il valore del descrittore per ogni vertice della mesh.
  - ▷ Selezionare n descrittori tra quelli calcolati che risultano i migliori in accordo a una specifica strategia.
  - ▷ Fornire una lista di requisiti che la mesh deve possedere affinché il descrittore possa essere calcolato correttamente.
  - ▷ Fornire una nozione di *descrittore non attendibile (null value)*.
  - ▷ Fornire una nozione di *persistenza* per l'attributo descrizione.

3.3.1 Attributi

L'attributo *descrizione* è quello che contiene il valore del descrittore vero e proprio, come ad esempio il valore della *curvatura* o dell'*integral volume descriptor* (Sezione 2.3.3). L'impiego di un vettore per rappresentare questo attributo ci consente di realizzare sia *mono-dimensional feature* che *multi-dimensional feature* e *multi-scale feature*. Si deve porre attenzione al fatto che feature multi dimensionali e feature multi scala hanno la stessa rappresentazione; è solo l'uso che viene fatto dell'attributo descrizione, da parte della feature stessa, che ne determina la natura. Il valore della feature viene calcolato per un dato punto p sulla superficie di una mesh; le coordinate del

punto sono memorizzate nell'attributo *posizione*, mentre la direzione della normale al punto p viene memorizzata nell'attributo *normale*. L'informazione sulla normale al punto p è usata in più parti del framework per migliorarne l'efficacia; la necessità di fornire questa ulteriore informazione tuttavia non costituisce una perdita di generalità in quanto da una nuvola di punti è possibile calcolare una ragionevole normale per ogni punto, come già menzionato nella sezione 2.2.1.

3.3.2 Funzionalità

Le funzionalità principali riguardano il calcolo dei descrittori e la loro selezione. Quando viene realizzata una specifica feature deve essere definita una funzione che calcola il valore del descrittore per tutti i vertici della mesh; il descrittore calcolato viene assegnato all'attributo *descrizione*. I dettagli della computazione da eseguire riguardano la natura della specifica feature implementata.

Può capitare spesso di avere a che fare con descrittori il cui valore non sia definito con certezza per tutti i punti della range map; per molte feature una situazione di questo genere si verifica in prossimità dei bordi o di buchi. Per questa evenienza ogni feature viene fornita di un valore speciale, detto *null value*, con il significato di *'descrittore non attendibile'*. Il *null value* deve essere usato per gestire situazioni come quella sopra descritta, così da poter garantire che ogni punto della scansione abbia un descrittore associato.

Data la varietà delle tipologie di feature esistenti è comprensibile che ogni feature necessiti di un diverso insieme di dati per calcolare il proprio descrittore. Alcune feature, ad esempio, possono essere calcolate a partire dalla sola nuvola di punti, altre possono avere bisogno anche delle direzioni normali ai punti o della topologia della mesh. Per ogni feature creata è dunque indispensabile fornire una lista di requisiti che la range map deve

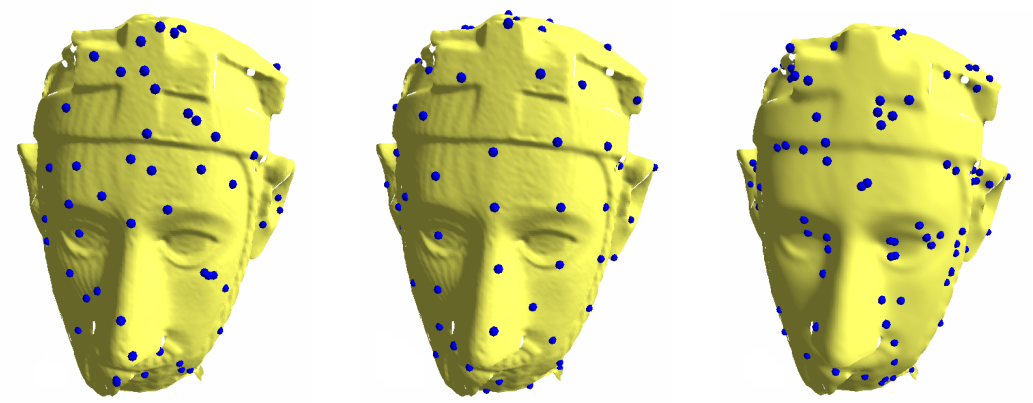
possedere affinché si possa procedere al calcolo dei descrittori; tale lista verrà richiesta dal framework di allineamento che la userà per arricchire la range map con le informazioni necessarie.

Dopo aver calcolato il descrittore per tutti i punti della range map è necessario poter selezionare un sottoinsieme di punti in base al valore del descrittore associato. Infatti, come spiegato nella sezione 2.3.1, per poter ottenere un allineamento è necessario individuare un ristretto numero di punti coincidenti sulla range map da registrare. I descrittori vengono proprio impiegati per discernere una ridotta quantità di punti che sia facile riconoscere sull'altra mesh; questo viene fatto esaminando separatamente l'istogramma di ogni dimensione del vettore dei descrittori e isolandone le aree più promettenti.

Definire delle soglie che identifichino un'area contenente valori affidabili si è rivelato un compito molto complicato, in quanto gli istogrammi dei descrittori mutano significativamente il loro aspetto al variare dei tipi di feature, del metodo di computazione usato, della quantità di noise e delle range map considerate. Per questo motivo si è scelto di vagliare separatamente più istogrammi, ciascuno relativo a una dimensione dell'attributo descrizione. I punti scartati vengono contrassegnati tramite il *null value*.

Da questo sottoinsieme vengono estratti gli n punti con descrittori 'migliori' in accordo a un certo criterio. Ad esempio, un criterio può essere quello di selezionare n punti in modo casuale [Fig. 3.2a], un altro criterio può selezionare i punti in modo che risultino tra loro più distanti di una distanza d [Fig. 3.2b], un altro ancora può selezionare i punti in modo equalizzato rispetto all'istogramma dei descrittori [Fig. 3.2c]. Per ogni feature esiste una apposita funzione che deve definire tutti questi dettagli che sono specifici della feature da realizzare.

Per non perdere generalità tutte le feature sono anche provviste di una nozione di *persistenza*. Come visto nella sezione 2.3.4, nel caso di feature multi scala o multi dimensionali, è una pratica comune selezionare solo



(a) Criterio casuale **(b)** Criterio spaziale **(c)** Equalizzazione

Figura 3.2: *Le immagini mostrano tre diverse strategie per selezionare i punti sulla mesh. (a) Punti selezionati casualmente; (b) Punti selezionati in modo da essere tra loro più distanti di una quantità d; (c) I punti vengono campionati in modo equalizzato rispetto all'istogramma dei descrittori che indicano la curvatura assoluta.*

valori dei descrittori che siano persistenti su più scale. Anche la nozione di persistenza è strettamente legata al genere di feature che si intende realizzare e per questo deve essere opportunamente definita. Per fare un esempio molto usato in letteratura, una feature viene considerata persistente se il valore del descrittore è definito, ossia non è un *null value*, per tutte le scale.

Lo schema in figura 3.3 illustra l'intero procedimento effettuato per ottenere un sottoinsieme di punti significativi a partire dal calcolo dei descrittori di una feature multi-scala.

3.4 Feature implementate

Parte integrante del lavoro svolto è stata l'implementazione di alcune feature con cui testare e verificare il framework realizzato. Sono state realizzate delle

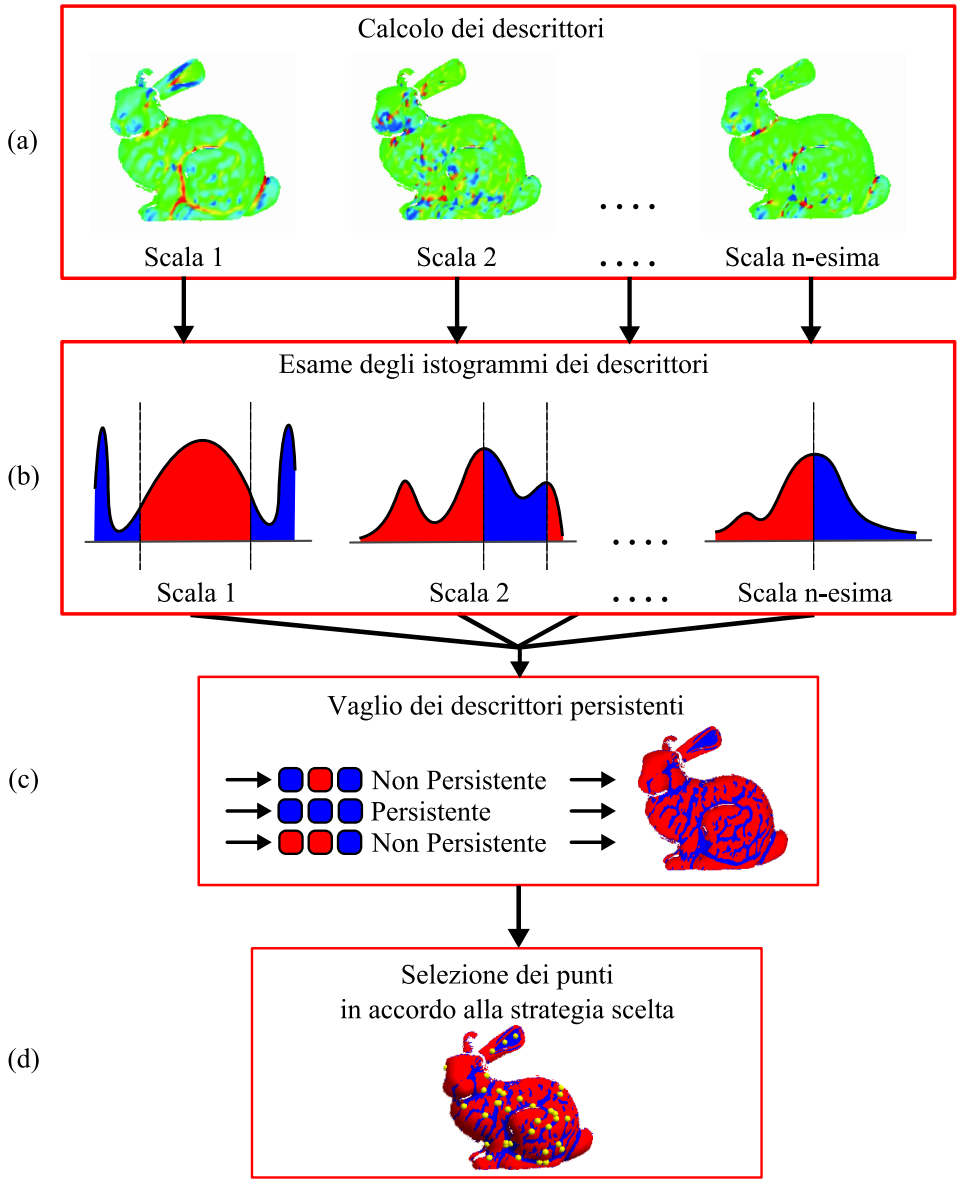


Figura 3.3: *Lo schema mostra la sequenza di operazioni da compiere per selezionare un ristretto sottoinsieme di punti. (a) Per ogni scala vengono calcolati i descrittori (b) Per ogni scala viene isolata una parte dell'istogramma dei descrittori (c) Vengono scartati tutti i descrittori non persistenti (d) Si estrae un sottoinsieme di punti in accordo a una precisa strategia.*

feature multi dimensionali e delle *feature multi scala*. Il motivo di questa scelta sta nel fatto che queste classi di feature offrono un buon compromesso tra complessità di calcolo, efficacia e potere discriminatorio. Un discorso a parte va fatto per la feature *ground truth*: quest'ultima è stata pensata col solo scopo di verificare il corretto funzionamento del framework e quindi non è stata considerata per valutare l'efficacia o le prestazioni del procedimento di allineamento. Nel seguito si descrivono in dettaglio le feature implementate.

3.4.1 GaussianMeanAbsolute Curvature

Questa feature multi dimensionale rappresenta il valore del descrittore con un vettore di tre scalari; il primo scalare contiene il valore della curvatura gaussiana, il secondo contiene il valore della curvatura media, il terzo contiene il valore della curvatura assoluta.

La curvatura è calcolata con il metodo di Taubin [Tau95a] e non viene considerata sui bordi della range map, zona dove solitamente assume valori non rilevanti. Per la selezione dei punti vengono considerati solo quelli con valore del descrittore persistente: una feature è persistente se il vettore descrizione non contiene nessun *null value*.

Per ogni tipo di curvatura si cerca di scartare i valori che sono evidentemente non rilevanti. Questo viene fatto costruendo l'istogramma per ogni dimensione dei descrittori calcolati e tagliando le fasce esterne dell'istogramma, che corrispondono ai valori di curvatura alti in valore assoluto [Fig. 3.4]. Gli istogrammi vengono tagliati al 15° e al 90° percentile.

Per estrarre n punti viene eseguito un *sampling uniforme*; ciò significa che qualunque punto ha la stessa probabilità di essere estratto.

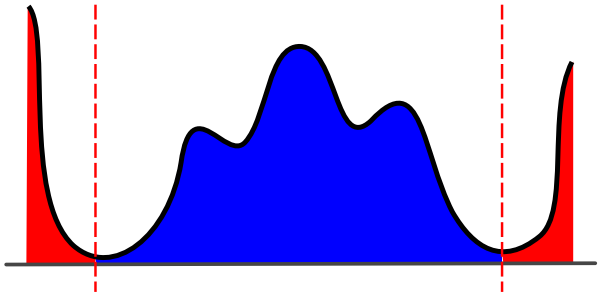


Figura 3.4: *La figura illustra in modo schematico il risultato che si cerca di ottenere eliminando i valori dei descrittori alti in valore assoluto. Gli estremi dell'istogramma sono chiaramente valori non affidabili e sono rimossi (in rosso). Si cerca invece di preservare interamente la parte di valori affidabili (in blu), tra cui si andrà a estrarre un insieme di punti rappresentativi per l'allineamento.*

3.4.2 MeanSmooth Curvature

Questa feature multi scala rappresenta il valore del descrittore con un vettore di tre scalari, ognuno dei quali indica un valore della curvatura media calcolato per successivi livelli di *smoothness* della range map. La prima dimensione del vettore descrizione si riferisce alla range map originale; la seconda dimensione si riferisce alla range map dopo sei iterazioni di *Laplacian smooth*, mentre la terza dimensione si riferisce alla range map dopo altre sei iterazioni di *Laplacian smooth*.

Sia il calcolo della curvatura che del Laplacian smooth seguono il procedimento di Taubin [Tau95a, Tau95b]; come di consueto la curvatura non è calcolata sui bordi della mesh. La feature è persistente se il vettore descrizione non contiene nessun *null value*.

Per tutti i livelli di *smoothness* si cerca di eliminare i valori che sono evidentemente non rilevanti. In questo caso si è scelto di mantenere l'informazione contenuta nelle zone esterne dell'istogramma dei descrittori, che

corrispondono ai valori di curvatura alti in valore assoluto [Fig. 3.5]. Sebbene questo tipo di informazione possa essere inaffidabile, poiché talvolta generata in corrispondenza artefatti visivi, se opportunamente depurata rappresenta invece un'ottima fonte di zone 'salienti' della scansione. In questo contesto il *Laplacian smooth* ha l'effetto di attenuare la presenza di valori della curvatura non affidabili; ne deriva che il calcolo della curvatura compiuto per diversi livelli di smoothness della range map ci garantisce una sufficiente stabilità dei dati. Ciascun livello di *smoothness* viene tagliato al 40° e al 90° percentile.

Anche in questo caso come per la feature precedente, per estrarre n punti viene eseguito un sampling uniforme.

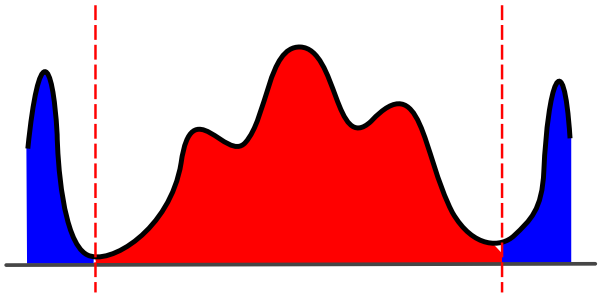
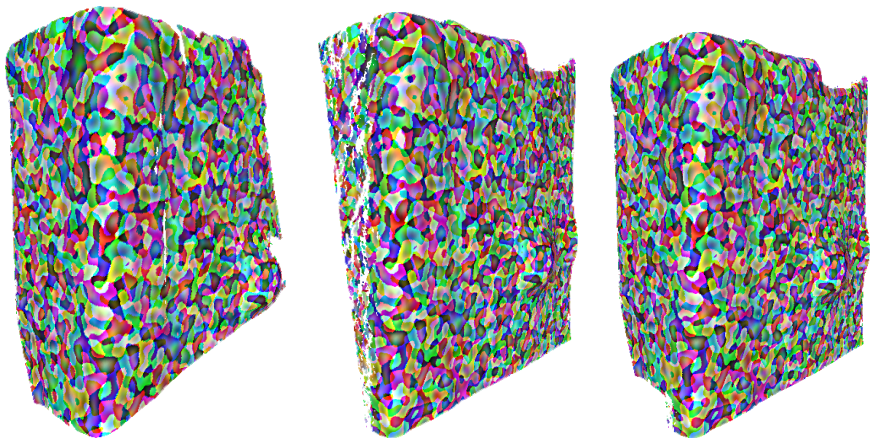


Figura 3.5: *La figura illustra in modo schematico il risultato che si vuole ottenere mantenendo i descrittori grandi in valore assoluto. La parte centrale dell'istogramma è computazionalmente affidabile ma poco rappresentativa della superficie; perciò viene rimossa (in rosso). Vengono mantenute le zone esterne (in blu), più descrittive della superficie e rese attendibili dalle iterazioni di Laplacian smooth.*

3.4.3 Ground truth feature

La feature ground truth è stata ideata contestualmente a un particolare test per verificare la correttezza del framework implementato; pertanto prima di illustrare in dettaglio la feature è necessario spiegare il test utilizzato. Ai fini

del test è necessario che i descrittori nelle zone di sovrapposizione delle due range map risultino simili; per garantire questa proprietà le due range map sono state allineate approssimativamente a mano e raffinate tramite ICP (sezione 2.5), dopodiché sono state colorate utilizzando la funzione *perlin noise* che dipende dalla posizione. Questo significa che per due punti nella stessa posizione viene calcolato lo stesso valore, mentre per due punti vicini vengono calcolati valori vicini. Con queste operazioni abbiamo la certezza che il colore delle due range map combaci nella zona di overlap [Fig 3.6].



(a) Allineate **(b)** Range map 1 **(c)** Range map 2

Figura 3.6: *Le figure mostrano due scansioni colorate tramite la funzione perlin noise dopo essere state allineate. Le zone che si sovrappongono hanno colori quasi identici.*

Come feature ground truth è stato utilizzato l'attributo colore. Il descrittore è rappresentato tramite un vettore di tre scalari che contiene le componenti *rgb*. Il calcolo del descrittore consiste nel trasferire nell'attributo descrizione della feature il valore del colore per vertice. Questa particolare feature non ha bisogno dell'analisi dell'istogramma dei valori, né della nozione di persistenza, poiché tutti i valori sono validi e utilizzabili per estrarre un sottoinsieme di punti; a questo scopo viene eseguito un sampling uniforme.

3.5 Selezione delle basi di punti

Ogni iterazione dell'algoritmo RANSAC inizia estraendo una base di punti dalla mesh $\mathcal{M}_{fix}$. In generale, una *base* è formata da un numero di punti n_{base} maggiore o uguale a tre; nel nostro caso ne saranno usati quattro per rendere più stabile il successivo calcolo della trasformazione rigida. Tramite la funzionalità di selezione fornita dalla specifica feature utilizzata (sezione 3.3.2), viene ricavato un sottoinsieme di $n_{subset} = 250$ punti; tra questi viene estratta casualmente una base B di quattro punti distinti.

Una volta scelta una base, si verifica che tutti i punti che la compongono si trovino tra loro a una distanza superiore a una determinata soglia δ_{base}. Se un punto viola questo vincolo la base viene rifiutata, l'iterazione del RANSAC viene conclusa prematuramente, e si procede con l'iterazione successiva che tenterà di estrarre una nuova base di punti. Questa verifica spaziale viene fatta in quanto è noto che basi di punti tra loro distanti producono allineamenti di migliore qualità [GMO94], in particolare si andrebbe incontro ad allineamenti erronei se i punti si trovassero a una distanza inferiore al doppio di quella utilizzata per asserire la validità dell'allineamento stesso. Per questo si è impostato come valore di δ_{base} il doppio della *distanza di consenso* (sezione 3.8.1).

3.6 Matching

La funzionalità di *matching* fa parte del framework di allineamento e sfrutta i dati forniti dalla feature per ricavare un insieme di basi corrispondenti e ammissibili. Data una base B di punti su $\mathcal{M}_{fix}$, la procedura di *matching* cerca per ogni punto di B un insieme di punti corrispondenti, cioè con valore del descrittore simile, su $\mathcal{M}_{mov}$. A partire da questi insiemi di punti candidati si esplorano esaustivamente tutte le quadruple di punti che formano una

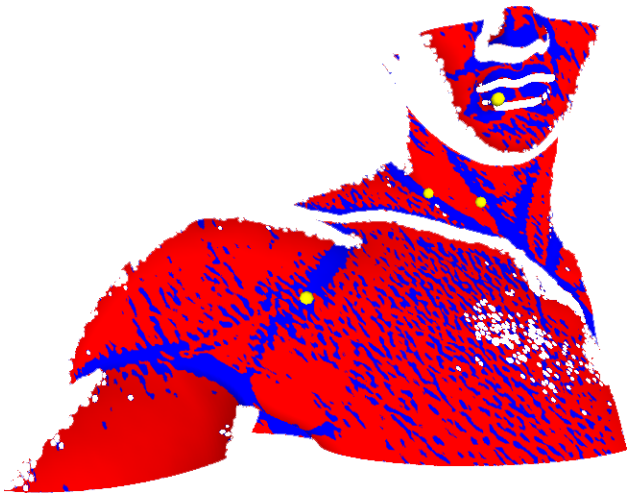


Figura 3.7: *La figura mostra una base di quattro punti (sfere gialle) selezionate dall'algoritmo di allineamento. Si è usata la feature MeanSmooth Curvature: questa filtra i descrittori affidabili e persistenti (zone blu), e ne fornisce casualmente un sottoinsieme di 250 punti. A ogni iterazione la procedura di allineamento estrae casualmente tra questi una base di 4 punti e verifica il vincolo di spazialità.*

possibile base coincidente. La procedura di matching restituisce un insieme di basi verosimilmente coincidenti. La procedura avviene dunque in due passi; prima si individuano per ogni punto della base i punti corrispondenti sulla mesh $\mathcal{M}_{mov}$, in seguito vengono vagliate tutte le basi possibili e isolate quelle ammissibili.

3.6.1 Individuare i punti corrispondenti

Dato un punto $p \in B$ con valore del descrittore f_p, si esegue una *k-nearest neighbor search* dei k punti con valore del descrittore più vicino a f_p. Il parametro k è stato impostato a 150. La ricerca è implementata in maniera efficiente tramite l'utilizzo di *kd-tree* [AMS+98], una particolare struttura

per il partizionamento spaziale. Il *kd-tree* indicizza tutti i punti di $\mathcal{M}_{mov}$ con valori dei descrittori affidabili.

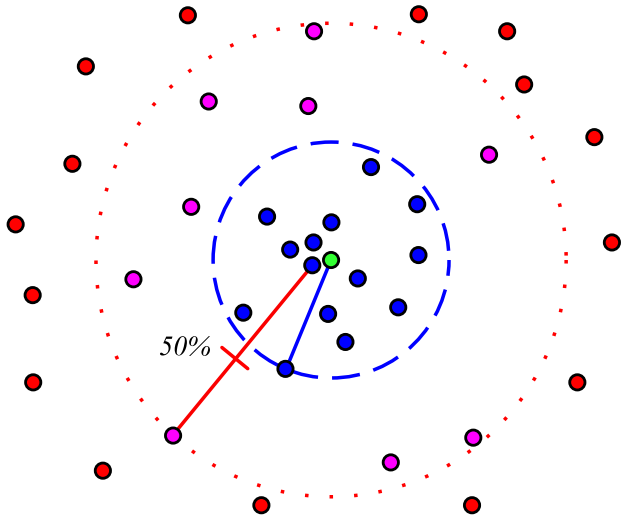


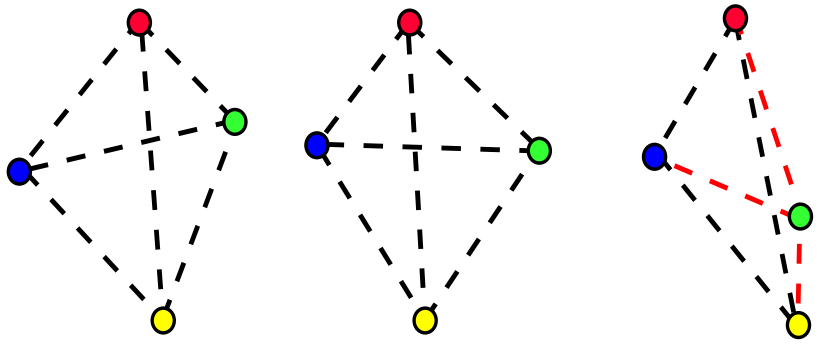
Figura 3.8: *Selezione dei punti coincidenti in uno spazio dei descrittori bidimensionale. Il punto verde al centro rappresenta la query; all'interno della circonferenza rossa si trovano tutti i k punti restituiti dalla k-nearest neighbor search; i punti blu sono quelli che ricadono nella circonferenza che ha raggio pari alla metà della distanza tra il descrittore più vicino e quello più distante.*

I 150 punti ottenuti tramite *k-nearest neighbor search* vengono ulteriormente esaminati per conservarne solo la parte più significativa. In questo caso ci si avvale di un criterio sulla distanza nello spazio dei descrittori; tutti i punti con valori dei descrittori oltre una certa soglia di distanza vengono scartati [Fig 3.8]. La soglia λ viene impostata come una percentuale della distanza tra il descrittore più vicino e quello più lontano. Si è stabilita una soglia del 50%.

3.6.2 Ricerca delle basi ammissibili

A questo punto la procedura di *matching* dispone di quattro insiemi di punti, ognuno dei quali ne contiene al più 150. Scegliendo un punto da ciascun insieme si ottiene una *base coincidente* (*matched base*), ossia una quadrupla di punti corrispondenti ai punti della base B di $\mathcal{M}_{fix}$. Lo spazio delle basi possibili contiene al più 150^4 elementi, ossia poco più di 500 milioni di basi.

Lo spazio delle basi può essere esaminato per intero in modo efficiente con l'algoritmo *Branch & Bound*, un algoritmo di enumerazione implicita. L'algoritmo enumera tutte le possibili basi, senza però considerare alcuni sottoinsiemi che possono essere preventivamente contraddistinti come non ammissibili ai fini dell'allineamento.



(a) Base **(b)** Ammissibile **(c)** Non ammissibile

Figura 3.9: *Gli schemi rappresentano delle basi e le loro interdistanze. (a) La base B di riferimento (b) Una base ammissibile; la disposizione spaziale è simile a quella della base B, le differenze non si ripercuotono troppo sulle interdistanze tra i punti. (c) Una base non ammissibile. In rosso sono tracciate le interdistanze che risultano significativamente diverse dalle originali, alterando la configurazione spaziale della base.*

Il criterio usato per stabilire l'ammissibilità di una base è l'interdistanza tra i suoi punti; in altre parole una base B' è ammissibile rispetto a una base

B se le distanze tra tutti i punti di B' sono uguali alle distanze tra tutti i punti di B, a meno di una certa soglia di errore ϵ_{base}. Intuitivamente questo significa che, affinché due basi possano essere allineate correttamente, le due quadruple di punti devono avere una configurazione spaziale simile [Fig 3.9].

Per comprendere il funzionamento della procedura di enumerazione implicita possiamo immaginare che quest'ultimo lavori su un'albero di quattro livelli: il primo livello contiene i punti candidati corrispondenti al primo punto di B, il secondo livello contiene i punti candidati corrispondenti al secondo punto di B, etc...[Fig 3.10].

L'algoritmo considera un punto del primo livello e uno del secondo; valuta la distanza e la confronta con la distanza dei relativi punti di B. Se il criterio non è soddisfatto, si passa a considerare una nuova coppia formata con un'altro punto del secondo livello. Questo equivale a dire che l'intero sotto-albero della prima coppia è stato visitato implicitamente, con un risparmio di circa 150^2 visite. Se invece il criterio di interdistanza è soddisfatto, si passa a esaminare una tripla di punti aggiungendo alla coppia precedente un punto del terzo livello. Le quadruple di punti che verificano i vincoli spaziali vengono annotate e costituiscono il risultato della procedura di *matching*.

3.7 Calcolo della trasformazione rigida

La trasformazione rigida che allinea due range map viene calcolata con il metodo proposto da Besl e McKay [BM92] a partire da due insiemi di punti. Dato che nello spazio 3D due terne di punti sono sufficienti per determinare una matrice di roto-traslazione, è possibile ottenere la matrice cercata da due basi corrispondenti. Considerando che le quadruple di punti coincidono in maniera approssimata, e non esatta, la matrice calcolata è quella che meglio sovrappone le due basi di punti; tuttavia l'allineamento fornito rimane un'approssimazione [Fig 3.11].

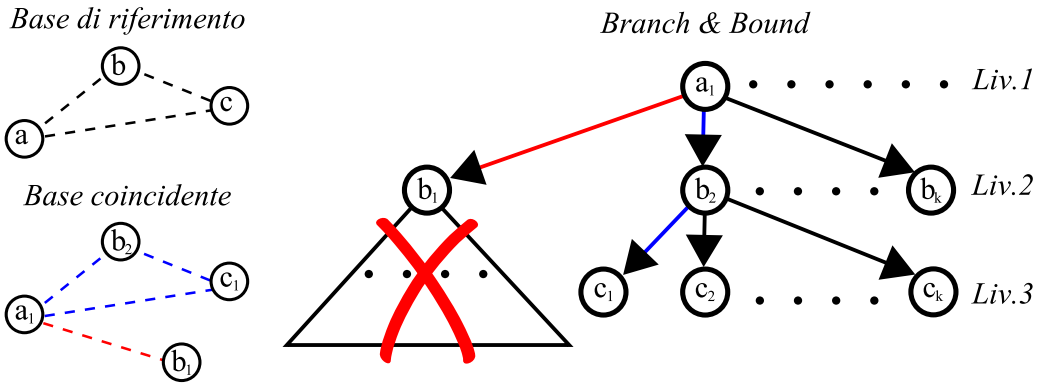
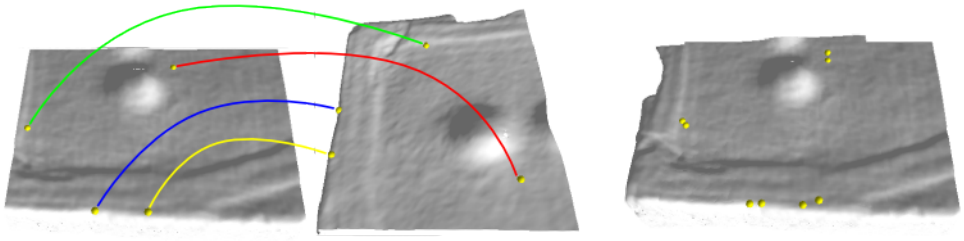


Figura 3.10: *Rappresentazione grafica dell'algoritmo Branch & Bound per una base di 3 punti. Viene visitato un albero di 3 livelli, ogni livello contiene i candidati per un punto della base. Poiché la coppia a_1b_1 non verifica il vincolo sulle interdistanze, ossia $\overline{a_1b_1} \neq \overline{ab}$, tutto il sottoalbero relativo alle triple a_1b_1* viene considerato visitato implicitamente. Si passa a verificare la coppia a_1b_2 e poi la tripla $a_1b_2c_1$, che risulta ammissibile.*

La matrice di trasformazione rigida deve essere calcolata per tutte le basi ammissibili fornite dalla fase di *matching*; questo è necessario affinché possa essere stimata la bontà di tutti gli allineamenti e venga scelto il migliore come soluzione del problema di registrazione.

3.7.1 Ordinamento delle trasformazioni rigide

La procedura di allineamento termina quando viene raggiunto il limite massimo di iterazioni o quando viene trovata una trasformazione rigida che possa essere considerata ottimale, vale a dire che superi una certa soglia di consenso $\Delta_{success}$ (sezione 3.8.2). Per migliorare le prestazioni della procedura ha dunque senso cercare di individuare le trasformazioni rigide più promettenti tra tutte quelle fornite dalla fase di *matching*.



(a) Prima dell'allineamento **(b)** Dopo l'allineamento

Figura 3.11: *Le immagini mostrano come la trasformazione rigida cerchi di allineare nel miglior modo possibile le due basi di punti coincidenti. (a) Le basi di punti sulle due range map non allineate. Gli archi indicano le corrispondenze tra i punti (b) Le basi dopo l'allineamento. Si noti come la trasformazione cerchi di far coincidere i punti.*

Questo viene fatto ordinando le trasformazioni rigide in base a una quantità q che ne misuri, sebbene in modo approssimativo, la bontà. Una stima più precisa sarà ovviamente fornita dalla fase di consenso. Il criterio utilizzato a questo scopo è la somma della distanza tra i punti corrispondenti delle basi dopo l'allineamento, pesata con il prodotto scalare delle loro normali. Il criterio è precisato dalla formula seguente, dove p_i^{fix} e p_i^{mov} indicano la posizione di punti corrispondenti delle basi, mentre N_i^{fix} e N_i^{mov} indicano le loro normali.

$$q = \sum_{i=0}^{3} \parallel p_i^{fix}, p_i^{mov} \parallel \cdot \left(1 - \left(N_i^{fix} \cdot N_i^{mov}\right)\right)$$

Bisogna porre attenzione al fatto che la quantità q può essere calcolata solo dopo avere allineato le due basi di punti. Per questo si procede come descritto di seguito [Fig 3.12].

Subito dopo aver calcolato la trasformazione rigida, questa viene applicata alle basi di punti, viene calcolata la quantità q e memorizzata assieme alla

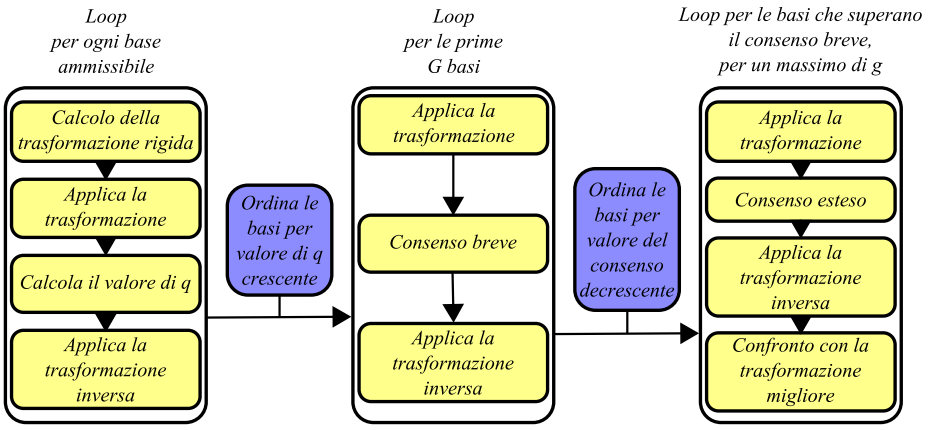


Figura 3.12: *Lo schema mostra in dettaglio le operazioni da compiere prima e durante la fase di consenso, al fine di diminuire il tempo di calcolo dell'algoritmo. Prima della fase di consenso, le trasformazioni rigide vengono ordinate in base al valore di q; dopo la fase di short consensus (sezione 3.8.1) vengono ordinate in base al punteggio ottenuto.*

matrice di trasformazione in un apposito record. Una volta considerate tutte le basi candidate, i record vengono ordinati in base al valore della qualità q. Le trasformazioni rigide così ordinate forniscono l'input della fase di consenso.

3.8 Consenso

Prima di descrivere i dettagli della procedura, è bene ricordare che lo scopo della fase di consenso è quello di dare una valutazione della trasformazione rigida, così da stabilire se l'allineamento prodotto sia da ritenersi corretto oppure no. Il tipico svolgimento della procedura di *consenso* è gia stato illustrato nella sezione 2.4.2. La versione da noi implementata è una variazione di quella usata da Irani e Raghavan [IR96] (sezione 2.4.2); la procedura di consenso può essere suddivisa in due parti; *consenso breve* (*short consensus*)

e *consenso esteso* (*full consensus*), che verranno descritte nel seguito (sezioni 3.8.1 e 3.8.2).

La procedura di consenso viene ripetuta per tutte le trasformazioni rigide 'candidate' fornite dagli stadi precedenti della procedura di allineamento. Per questo motivo prima di eseguire la procedura di consenso vera e propria viene calcolata la trasformazione rigida e le due scansioni vengono espresse in un unico spazio di coordinate; viceversa, subito dopo la procedura di consenso deve essere calcolata la trasformazione inversa per riportare le scansioni nella posizione originale.

3.8.1 Short consensus

La fase di consenso breve consiste nel selezionare $n_{short} = 200$ punti dalla mesh $\mathcal{M}_{mov}$ e per ognuno di essi verificare se il punto più vicino sulla mesh $\mathcal{M}_{fix}$ si trova entro la *distanza di consenso* δ. Inoltre si controlla anche se l'angolo formato dalle due normali dei punti sia minore di un angolo α. Se entrambe le condizioni sono soddisfatte viene incrementato il punteggio del consenso, poiché i punti sono da considerarsi allineati correttamente. Nel caso contrario, in cui anche una sola delle condizioni non sia verificata, il punteggio non viene incrementato [Fig 3.13, 3.14]. La *distanza di consenso* δ è stata scelta pari al 2% della diagonale del bounding box della mesh $\mathcal{M}_{mov}$, mentre l'angolo α è stato impostato a 15°.

I 200 punti vengono selezionati in modo equalizzato rispetto alle direzioni delle normali dei vertici di $\mathcal{M}_{mov}$. Rispetto a una selezione casuale, questa strategia ci garantisce di andare a testare dei punti su tutte le angolazioni della mesh; considerando il ristretto numero di punti, questo comporta un considerevole incremento della affidabilità del consenso. L'efficacia di questa tecnica è stata studiata in modo approfondito da Rusinkiewicz e Levoy [RL].

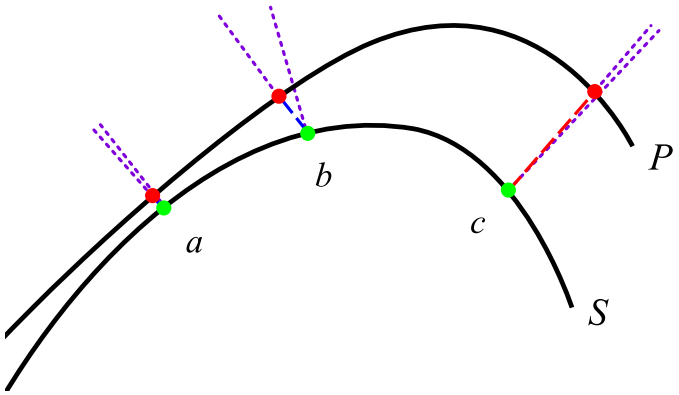
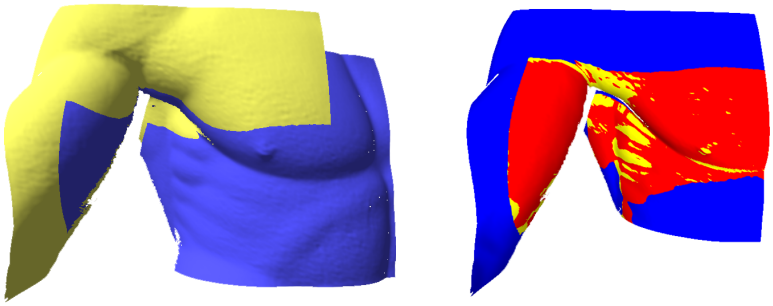


Figura 3.13: *Tre tipiche situazioni che si possono verificare durante il consenso. (a) I due punti sono in consenso. Le distanze tra i punti sono minori di δ, le normali (tratteggio viola) sono in un cono di α gradi. (b) I punti non sono in consenso. I punti sono abbastanza vicini (tratteggio blu) ma le normali sono troppo distanti tra loro. (c) I punti non sono in consenso. Le normali coincidono ma i punti sono troppo distanti (tratteggio rosso).*

Il consenso breve viene eseguito solo per le prime G trasformazioni rigide della lista, ossia per le G trasformazioni più promettenti. Il parametro G è stato oggetto di verifiche sperimentali ed è stato impostato pari a 50. Il punteggio di ogni consenso viene memorizzato in un opportuno record. Dopo aver compiuto (al più) G consensi brevi, le trasformazioni rigide relative vengono nuovamente ordinate, questa volta con punteggio del consenso decrescente [Fig 3.12]. La lista di trasformazioni rigide così ordinata viene fornita in input al *full consensus*.

3.8.2 Full consensus

Il consenso esteso è del tutto analogo al consenso breve, salvo per il fatto che il numero di punti selezionati dalla mesh $\mathcal{M}_{mov}$ sale a $n_{full} = 2500$. Affinché il tempo di completamento del RANSAC rimanga entro limiti accettabili, è



(a) Posizioni iniziali **(b)** Risultato consenso

Figura 3.14: *Le immagini mostrano l'esito di una procedura di consenso condotta per tutti i punti di $\mathcal{M}_{mov}$. (a) L'allineamento da stimare: $\mathcal{M}_{mov}$ in giallo e $\mathcal{M}_{fix}$ in blu (b) I punti di $\mathcal{M}_{mov}$ colorati in base al consenso. Blu significa non in consenso perché troppo distante; giallo significa non in consenso perché le normali sono troppo distanti; rosso significa in consenso.*

indispensabile limitare il numero di *full consensus* eseguiti; questa infatti è la parte della procedura di allineamento che più incide sulla performance.

Il consenso esteso viene eseguito solo per le trasformazioni rigide che hanno realizzato un punteggio di *short consensus* sopra una soglia Δ_{short}, e comunque non sono mai considerate più di $g = 8$ roto-traslazioni [Fig 3.12]. Ogni *full consensus* effettuato restituisce un punteggio, calcolato allo stesso modo del consenso breve (sezione 3.8.1). Se il punteggio del consenso esteso supera una soglia Δ_{full} la trasformazione è da considerarsi 'buona'. Il punteggio corrente viene confrontato col miglior punteggio ottenuto fino a quel momento; se il punteggio corrente risulta essere più alto sia il punteggio che la trasformazione rigida vengono memorizzati come le migliori calcolate. Se invece il punteggio non supera la soglia Δ_{full} oppure non supera il miglior punteggio ottenuto, la trasformazione rigida viene definitivamente scartata, in quanto non sufficientemente 'buona'. Dopo aver valutato tutte

le trasformazioni rigide, la procedura restituisce la migliore trovata.

Le soglie Δ_{short}, Δ_{full} e $\Delta_{success}$, dipendono dalla percentuale di sovrapposizione stimata per le due range map, che costituisce un parametro del nostro algoritmo di allineamento. Sperimentalmente si è verificato che buoni compromessi tra qualità degli allineamenti e performance si ottengono con i seguenti valori:

- $\Delta_{success}$: 90% della percentuale di sovrapposizione stimata.
- Δ_{full}: 60% della percentuale di sovrapposizione stimata.
- Δ_{short}: 40% della percentuale di sovrapposizione stimata.

Infine è bene sottolineare l'importanza di una corretta stima del parametro *percentuale di sovrapposizione*. Una sovrastima del parametro porterebbe a non trovare allineamenti, o meglio a ridurre drasticamente le probabilità di successo del RANSAC. Una sottostima al contrario porterebbe ad allineamenti scorretti o eccessivamente approssimati.

Capitolo 4

Implementazione e design

4.1 Introduzione

Sino a questo momento si è discusso del procedimento di allineamento astraendo dagli aspetti implementativi e di design del software. Tali aspetti sono importanti sia per realizzare una effettiva indipendenza del procedimento di allineamento dai descrittori utilizzati, sia per garantire la portabilità dell'algoritmo all'interno di applicazioni software diverse. In questo capitolo vengono affrontati questi aspetti, con particolare riferimento alle tecnologie impiegate per la realizzazione dell'algoritmo (sezione 4.2), come il linguaggio di programmazione utilizzato, e alle librerie e strutture dati adoperate (sezione 4.4). Viene inoltre illustrato il design del software e le motivazioni che hanno portato a preferire tale soluzione (sezione 4.3).

4.2 Tecnologie impiegate

L'algoritmo realizzato deve essere considerato una funzionalità a sé, integrabile e utilizzabile all'interno di qualunque strumento software ne avesse bisogno. Il codice, scritto completamente in *C++*, è stato incluso all'interno

della *Visual Computer Graphic Library*, in breve *VCG Lib* (sezione 4.2.1), di cui l'algoritmo stesso adopera molte componenti.

Una versione funzionante dell'algoritmo è stata aggiunta a *Meshlab* (sezione 4.2.2), un software di elaborazione di mesh, anch'esso basato su *VCG Lib*.

4.2.1 VCG Lib

VCG Lib è una libreria, scritta in *C++*, per la gestione e l'elaborazione di mesh di triangoli. La libreria è il risultato del lavoro compiuto dal *Visual Computing Lab* dell' *ISTI*, un istituto del *National Research Council* (*CNR*) italiano. *VCG Lib* è disponibile sotto licenza GPL.

La libreria è piuttosto ampia e fornisce molte funzionalità per il trattamento di mesh; di seguito ne vengono riportate solo alcune che sono state utilizzate per la realizzazione dell'algoritmo proposto:

- ▷ Calcolo della curvatura
- ▷ Algoritmi avanzati di smoothing
- ▷ Algoritmi avanzati di semplificazione delle mesh
- ▷ Strutture avanzate per l'indicizzazione spaziale e per l'esecuzione efficiente di query

VCG Lib fornisce supporto per la creazione e la gestione delle componenti, delle strutture dati e degli algoritmi più comuni nel contesto della computer graphics; tuttavia non fornisce nessuno strumento per il rendering o per la creazione di GUI, in quanto non è stata pensata con l'intento di costituire una specifica applicazione. Tutti gli aspetti inerenti alla presentazione degli elementi di grafica, così come gli aspetti relativi alle interazioni tra l'utente e questi elementi, sono lasciati alla specifica applicazione che fà uso di *VCG*

Lib. Nel caso in questione, come software applicativo che ci consenta di impiegare l'algoritmo RANSAC, è stato scelto *Meshlab*.

4.2.2 Meshlab

Meshlab è una applicazione open source estendibile per l'editing e il processing di mesh di triangoli. Il software è stato creato per aiutare l'utente durante il trattamento di modelli 3D derivanti da scansioni di oggetti reali; a questo scopo fornisce un insieme di strumenti per la pulizia, la riparazione, l'ispezione, la visualizzazione e la conversione di mesh. *Meshlab* adopera VCG Lib per la rappresentazione interna delle mesh e delle altre strutture; per questo la libreria costituisce di fatto il nucleo delle funzionalità offerte. *Meshlab* è disponibile per le piattaforme Windows, Linux e MacOSX.

Meshlab è strutturato al suo interno secondo una architettura a plugin che lo rende estendibile e permette di aggiungere facilmente diverse funzionalià che operano sulle mesh. Questa caratteristica ha fatto sì che il nostro algoritmo di allineamento trovasse la sua naturale collocazione tra i plugin di *Meshlab*. Non solo l'algoritmo di allineamento è stato implementato come plugin di *Meshlab*, bensì, durante la fase di sviluppo, tutte le sue componenti, descritte nelle sezioni 3.2.1 e seguenti, sono state realizzate separatamente come plugin e testate singolarmente, prima di essere assemblate nel RANSAC.

4.3 Design

La principale prerogativa che ha guidato la progettazione delle classi è l'indipendenza del procedimento di allineamento dalle feature impiegate. Questa caratteristica consente di ottenere un'infrastruttura di base, efficiente ed estendibile per valutare la qualità e la quantità degli allineamenti ottenuti

per mezzo di feature diverse. Uno schema dell'architettura del software è mostrato in figura 4.1 a pagina 65.

Per ottenere sia l'indipendenza che la genericità delle parti di software realizzate, si è fatto un largo impiego del meccanismo dei *template*, fornito dal linguaggio *C++*. Le diverse feature sono state disegnate come classi templatate parametriche rispetto al tipo della mesh. Ogni feature espone un'interfaccia attraverso cui può essere correttamente utilizzata dalla procedura di allineamento. L'interfaccia della feature è stata implementata tramite metodi statici.

L'algoritmo di allineamento e tutte le sue parti sono state progettate come funzioni templatate di una classe statica. L'algoritmo è una funzione parametrica rispetto al tipo della mesh e della feature. La classe statica che racchiude la procedura di allineamento fa uso anche di altre classi e funzioni templatate definite all'interno della libreria *VCG Lib*.

Come si può vedere dallo schema in figura 4.1, l'intera infrastruttura del procedimento di allineamento è interamente contenuta in questi pochi files. La procedura di allineamento è invocata da una classe che realizza un apposito plugin di *Meshlab*, semplicemente implementando l'interfaccia *MeshFilterInterface*. La redefinizione di questi pochi metodi dell'interfaccia è tutto quello di cui c'è bisogno per integrare il procedimento di allineamento all'interno di *Meshlab*.

4.4 Strutture dati

Durante la realizzazione dell'algoritmo proposto è stato necessario impiegare due diverse strutture per il *partizionamento spaziale* (*spatial indexing*): un *kd-Tree* [Fig. 4.2], usato nella fase di *matching*, e una *griglia* (*grid*), adoperata durante la fase di *consenso*.

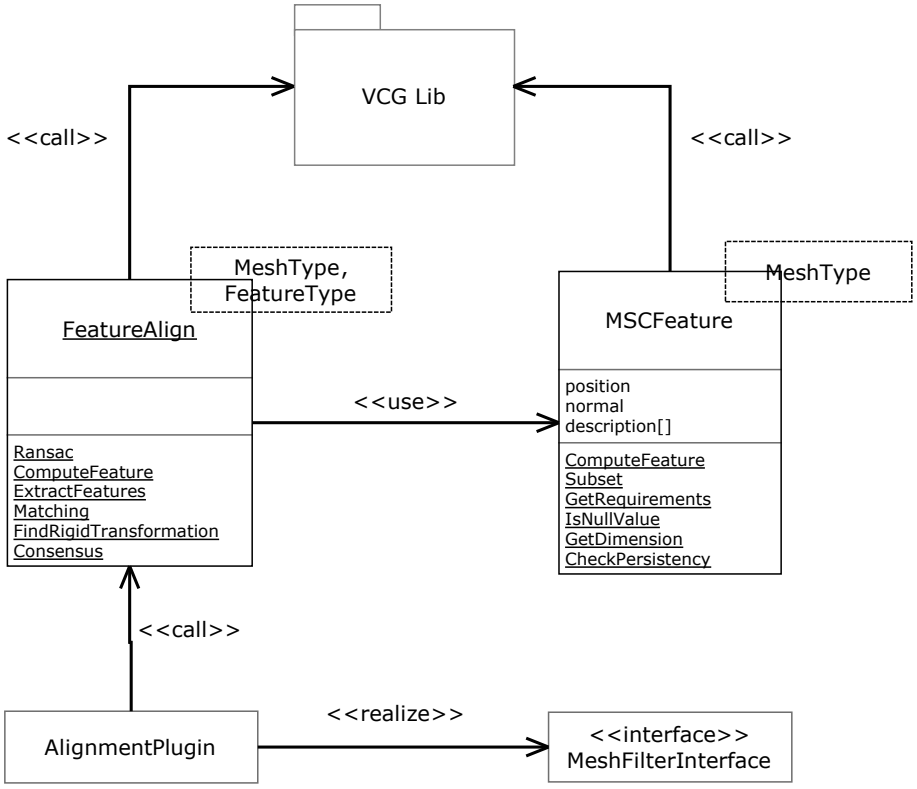


Figura 4.1: *Architettura del software. Le feature sono classi templatate che espongono una specifica interfaccia. L'algoritmo è racchiuso in una classe statica con funzioni templatate rispetto al tipo della mesh e della feature impiegata per l'allineamento. Entrambe queste classi usano la libreria VCG Lib. L'algoritmo è integrato in Meshlab tramite un plugin che invoca la procedura di allineamento come qualsiasi altra procedura della libreria VCG Lib.*

La necessità di impiegare queste strutture deriva dal dover eseguire, in contesti diversi, molte *kNN search* per cercare i punti candidati durante il matching o per ricavare il punto più vicino a un'altro durante la fase di consenso. Per ragioni di perfomance è vitale che queste query siano eseguite in modo efficiente, e da qui l'impiego di apposite strutture dati. Tuttavia è importante sottolineare che i due contesti in cui vengono svolte le query sono significativamente diversi e proprio queste differenze motivano la scelta di due strutture diverse tra loro.

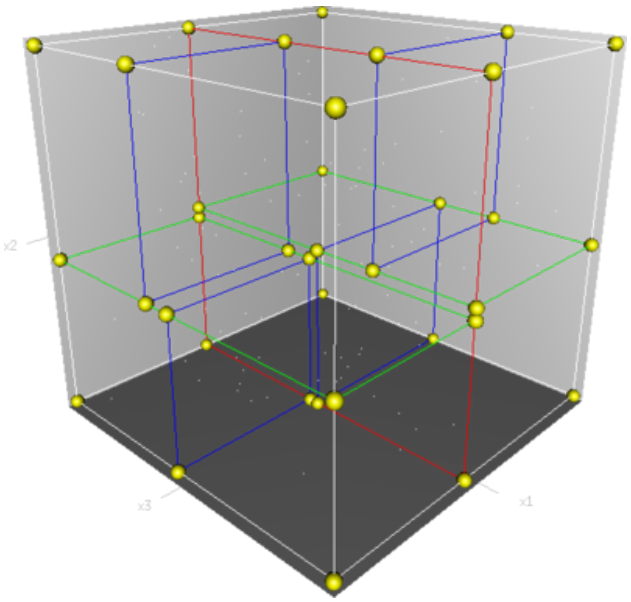


Figura 4.2: *Un esempio di kd-Tree con dimensione 3. Il primo piano di taglio (rosso) divide in due celle lo spazio; ognuna di queste è a sua volta divisa in due dal secondo taglio (verde). Infine le quattro celle sono divise ulteriormente dall'ultimo piano di taglio (blu).*

Come già spiegato nella sezione 3.6.1, la *kNN search* opera su elementi appartenenti allo spazio dei descrittori. Questo spazio è di dimensioni variabili[1] e, in generale, sparso. Il *kd-Tree* è la struttura più flessibile per gestire

[1]Nel caso in questione è lecito assumere $2^d << n$, con n numero di elementi indicizzati

ricerche in spazi di dimensioni variabili e si comporta, in media, in modo più efficiente di altre strutture quando opera in spazi sparsi di elementi.

Durante la fase di consenso si selezionano alcuni punti dalla mesh $\mathcal{M}_{mov}$ e si cercano i punti più vicini sulla mesh $\mathcal{M}_{fix}$ (sezione 3.8.1). In questo caso gli elementi indicizzati sono i punti della mesh $\mathcal{M}_{fix}$, che costituiscono uno spazio denso; nella fattispecie una *griglia* risulta essere, in media, più performante di una struttura gerarchica, come il *kd-Tree*.

Entrambe le strutture sono gestite in maniera efficiente; in particolare le strutture vengono popolate una sola volta nel corso dell'algoritmo e sfruttate per tutta la durata del RANSAC. *VCG Lib* fornisce una implementazione efficiente della *griglia*, che è stata dunque utilizzata all'interno del codice; per quanto riguarda il *kd-Tree*, abbiamo scelto di usare l'implementazione di Mount e Arya [AMS+98], riferita in molti articoli di settore come lo stato dell'arte.

e d dimensione dello spazio.

Capitolo 5

Risultati

5.1 Introduzione

Per validare sperimentalmente l'affidabilità del *framework* e per valutare l'efficacia delle diverse *feature*, il *framework* di allineamento è stato sottoposto ad una lunga batteria di test. Possiamo suddividere i test condotti in tre categorie, in base alla loro finalità. La prima serie di test, che è stata anche la prima a essere condotta in ordine cronologico, si propone di verificare la consistenza dell'implementazione con i comportamenti pronosticati dall'analisi teorica. Questa serie di test è stata importante anche per valutare il corretto funzionamento del framework. Durante i test è stata utilizzata la *feature ground truth*, appositamente progettata a questo scopo.

La seconda serie di test è stata impiegata per analizzare le prestazioni del framework, sia in termini di quantità di allineamenti trovati che in termini di tempo di calcolo impiegato. Questa serie di test è stata condotta su un buon numero di range map appartenenti a data set diversi, in modo che la casistica dei dati in input risultasse la più vasta e completa possibile. La batteria di test è stata ripetuta per tutti i tipi di feature implementate.

L'ultima serie di test condotti è servita sia a valutare gli effetti sulla performance causati da una accurata regolazione dei parametri del RANSAC,

sia a valutare l'impatto di alcune varianti apportate all'algoritmo.

Nella prossima sezione 5.2 saranno riassunti tutti i parametri presenti nel framework di allineamento con i loro valori di default. Nella sezione 5.3 sarà presentata in dettaglio la prima serie di test; in particolare verranno introdotte le misure per la sua comprensione e valutazione e verrano esposti i risultati ottenuti. Nella sezione 5.4 si procede alla descrizione della serie di test per la valutazione della performance; anche in questo caso verranno introdotte le misure per la sua valutazione e discussi i risultati ottenuti. In seguito, nella sezione 5.5 verranno illustrati e commentati gli esperimenti condotti durante la terza serie di test. Infine, la sezione 5.6 riassume i risultati ottenuti e propone una valutazione complessiva del lavoro svolto.

5.2 Parametri del framework

Prima di procedere con la descrizione dei test condotti e dei risultati ottenuti è indispensabile fornire una lista completa dei parametri presenti nel framework di allineamento e delle impostazioni di default con cui sono stati eseguiti i test. La lunga lista di parametri elencata nel seguito di questa sezione serve anche a dare al lettore l'idea di quanto anche un framework basato su algoritmi e concetti non molto difficili, possa rivelarsi notevolmente complesso al momento della sua implementazione. Possiamo aggiungere che, in un certo senso, la grande quantità di parametri in gioco è forse la migliore giustificazione dell'utilità di un framework come quello proposto, che sia generico, facile da modificare e, soprattutto, indipendente dalle feature.

Tutti i parametri sono stati già incontrati nel capitolo 3; ogni parametro è stato descritto contestualmente alla parte del framework in cui viene impiegato. Poichè risulta molto difficile dare una breve ed esauriente descrizione di ogni parametro, la lista sarà ordinata in base alle fasi del framework così da rimandare direttamente il lettore a una più completa descrizione.

5.2.1 Parametri di input

Qui sono elencati i parametri di input del framework di allineamento. Proprio perchè queste informazioni costituiscono l'input della procedura di allineamento, il loro valore deve essere sempre fornito esplicitamente dall'utente.

- ▷ *Input*
  - ▷ $\mathcal{M}_{fix}$ e $\mathcal{M}_{mov}$: rispettivamente la range map a cui allineare e quella da allineare;
  - ▷ *feature*: la feature da utilizzare per l'allineamento (*GaussianMeanAbsolute Curvature*, *MeanSmooth Curvature*, *Ground Truth*);
  - ▷ *overlap*: la percentuale di sovrapposizione della mesh $\mathcal{M}_{mov}$ rispetto alla mesh $\mathcal{M}_{fix}$;
  - ▷ $n_{iter} = 3000$: numero di iterazioni del RANSAC;

Vale la pena di ribadire l'importanza del parametro *overlap*: la stima fornita dall'utente deve essere il più possibile esatta; una sottostima comporterebbe il rischio di ottenere allineamenti erronei o eccessivamente approssimati, mentre una stima troppo ottimistica significherebbe avere una forte riduzione della probabilità di successo.

Il parametro n_{iter}, cioè il numero di iterazioni del RANSAC, è il principale strumento per bilanciare tempo di calcolo e probabilità di successo.

5.2.2 Altri parametri

Di seguito vengono elencati tutti i parametri del framework e i relativi valori di default. Per una descrizione accurata si rimanda alle sezioni indicate.

- ▷ *Funzionalità della feature* (sezione 3.3.2, pag. 41);
  - ▷ Strategia per la selezione dei descrittori = *uniform sampling*;

- ▷ *Selezione delle basi di punti* (sezione 3.5, pag. 49);
  - ▷ $n_{base} = 4$: numero di punti che formano una base;
  - ▷ $n_{subset} = 250$: numero di punti su $\mathcal{M}_{fix}$ tra cui vengono scelte le basi;
  - ▷ $\delta_{base} = 2 \cdot \delta$, i.e 4% della diagonale del BBox di $\mathcal{M}_{mov}$: minima distanza tra i punti di una base;
- ▷ *Matching* (sezione 3.6.1, pag. 50);
  - ▷ $k = 150$: numero di 'vicini', nello spazio del descrittore, tra cui vengono selezionati i punti coincidenti;
  - ▷ $\lambda = 50\%$: distanza percentuale, nello spazio dei descrittori, entro cui vengono selezionati i punti coincidenti;
  - ▷ $\epsilon_{base} = \delta$, i.e 2% della diagonale del BBox di $\mathcal{M}_{mov}$: soglia oltre la quale una base viene considerata non ammissibile;
- ▷ *Consenso* (sezione 3.8, pag. 56);
  - ▷ $\delta = 2\%$ della diagonale del BBox di $\mathcal{M}_{mov}$: distanza entro la quale due punti sono considerati in consenso;
  - ▷ $\alpha = 15°$: angolo entro cui le normali di due punti sono considerate in consenso;
  - ▷ $\Delta_{success} = 90\%$ dell'*overlap*: soglia di consenso oltre la quale l'allineamento è considerato ottimale;
  - ▷ $n_{short} = 200$: numero di punti verificati durante il consenso breve;
  - ▷ $G = 50$: massimo numero di trasformazioni per cui viene eseguito il consenso breve;
  - ▷ $\Delta_{short} = 40\%$ dell'*overlap*: soglia di consenso oltre la quale l'allineamento supera il consenso breve;

- ▷ $n_{full} = 2500$: numero di punti verificati durante il consenso esteso;
- ▷ $g = 8$: massimo numero di trasformazioni per cui viene eseguito il consenso esteso;
- ▷ $\Delta_{full} = 60\%$ dell'*overlap*: soglia di consenso oltre la quale l'allineamento supera il consenso esteso;

Si sottolinea che i parametri sopra riportati sono da considerare parte del *framework* di allineamento e non delle *feature*; quest'ultime posseggono anch'esse un certo numero di parametri, utili per la selezione dei descrittori più significativi, che sono stati già trattati nella sezione 3.3.2.

5.3 Consistenza del framework

Prima di poter valutare l'efficacia e le prestazioni del framework, è necessario garantire che il suo funzionamento sia corretto e consistente con le proprietà dell'algoritmo RANSAC. La prima serie di test si pone come obiettivi proprio la conferma di caratteristiche statistiche del RANSAC. Per raggiungere lo scopo si è impiegata la *ground truth feature* e il test progettato contestualmente ad essa (sezione 3.4.3). I risultati sperimentali ottenuti dalle batterie di test sono confrontati con il modello pronosticato da un'analisi teorica della procedura di allineamento.

5.3.1 Il modello teorico

Il modello teorico deve essere elaborato a partire da una analisi delle caratteristiche dell'algoritmo RANSAC. Quello che si vuole ricavare è una funzione che descriva come varia la probabilità di ottenere un allineamento con l'aumento del numero di iterazioni del RANSAC n_{iter}.

Per prima cosa si deve investigare la probabilità dell'evento *'allineamento trovato dopo una iterazione'*. Questa probabilità è costante e dipende, in primo luogo, dalla percentuale di sovrapposizione tra le due range map. Infatti, affinchè una singola iterazione di RANSAC produca un allineamento è necessario che tutti i punti della base si trovino nella zona di sovrapposizione. Ogni punto della base si trova nella zona di sovrapposizione con probabilità pari alla percentuale di *overlap*; poichè i punti della base sono scelti in modo indipendente, la probabilità cercata vale, in prima analisi, $overlap^{n_{base}}$, che è un valore costante per tutte le iterazioni. Tuttavia questo non è sufficiente per essere sicuri di ottenere un allineamento; infatti un'iterazione non si concluderà mai con successo se la fase di *matching* non fornisce buoni candidati; dunque è necessario che, dopo aver selezionato una base all'interno della zona di sovrapposizione, il *matching* stabilisca delle corrispondenze corrette. Pertanto possiamo ricavare la probabilità cercata utilizzando la probabilità condizionata. Riassumendo, possiamo affermare quanto segue:

$$A = \text{'allineamento trovato dopo una iterazione'} \tag{5.1}$$

$$B = \text{'}n_{base}\text{ punti selezionati dentro la zona di sovrapposizione'} \tag{5.2}$$

$$C = \text{'il } matching \text{ stabilisce le corrispondenze corrette'} \tag{5.3}$$

$$\mathcal{P}(A) = \mathcal{P}(B) \cdot \mathcal{P}(C \mid B) = overlap^{n_{base}} \cdot \phi = k \tag{5.4}$$

Nella formula 5.4 indichiamo con ϕ la probabilità che, una volta individuata una base di punti nella zona di sovrapposizione delle range map, il *matching* fornisca dei punti effettivamente coincidenti. La quantità ϕ esprime l'efficacia della fase di *matching* e dipende dalla distribuzione dei descrittori sulla range map considerata e quindi, in ultima analisi, dipende da quanto efficace sia una *feature* nel dare descrittori univoci ai vari punti della superficie. Dalla formula 5.4 possiamo ricavare la probabilità dell'evento *'nessun allineamento trovato dopo N iterazioni'*. La probabilità di fallire una iterazione vale $(1-k)$;

poiché le iterazioni sono indipendenti tra loro la probabilità cercata vale:

$$\mathcal{P}\left(\text{nessun allineamento trovato dopo N iterazioni}\right) = (1-k)^N \qquad (5.5)$$

Naturalmente a questo punto la probabilità che il RANSAC produca almeno un allineamento dopo N iterazioni può essere calcolata dall'evento complementare:

$$\mathcal{P}\left(\text{almeno un allineamento trovato dopo N iterazioni}\right) = 1-(1-k)^N \quad (5.6)$$

Il grafico della funzione 5.6 per alcuni valori k è riportato in figura 5.1; come si può osservare la probabilità segue una crescita esponenziale e la curva è tanto più ripida quanto più alto è il valore di k.

Un'altro modo di interpretare la funzione 5.6, che ci suggerisce come eseguire i test per calcolare la probabilità di successo, è il seguente: ripetendo per n volte la procedura di allineamento, con numero di iterazioni fissato pari a N, la probabilità di successo p sarà data dal rapporto tra i tentativi conclusi con successo e il numero di tentativi compiuti, formalmente:

$$p = \mathcal{P}\left(k, N\right) = \frac{n_{successi}}{n} \qquad (5.7)$$

Un'ultima considerazione riguarda l'impatto della variazione dell'*overlap* e della presenza di *noise* sulla probabilità di successo. Dall'equazione 5.4 è facile osservare che una diminuzione della percentuale di sovrapposizione comporta un calo del valore di k e un conseguente schiacciamento della curva [Fig. 5.1]. Lo stesso effetto si ottiene con l'aumento del rumore; quest'ultimo comporta una minore efficacia della fase di *matching*, ossia un valore più basso di ϕ, con un conseguente aggiustamento al ribasso del valore di k. In pratica questo significa che aumentando la quantità di rumore o diminuendo la percentuale di sovrapposizione, aumenta il numero di iterazioni neccessarie affinchè si possa essere ragionevolmente sicuri di trovare un allinemento.

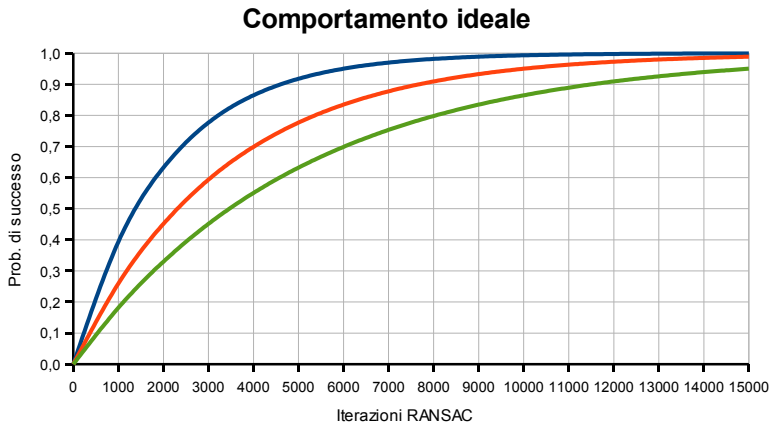


Figura 5.1: *Le curve mostrano il comportamento ideale descritto dall'equazione 5.6 per diversi valori di k. La linea blu ha il valore di k più elevato; la probabilità di successo si stabilizza velocemente a 1. La linea rossa ha un valore di k medio e cresce più lentamente. Infine la linea verde ha un basso valore di k e risulta più schiacciata pur continuando a crescere.*

5.3.2 Il test

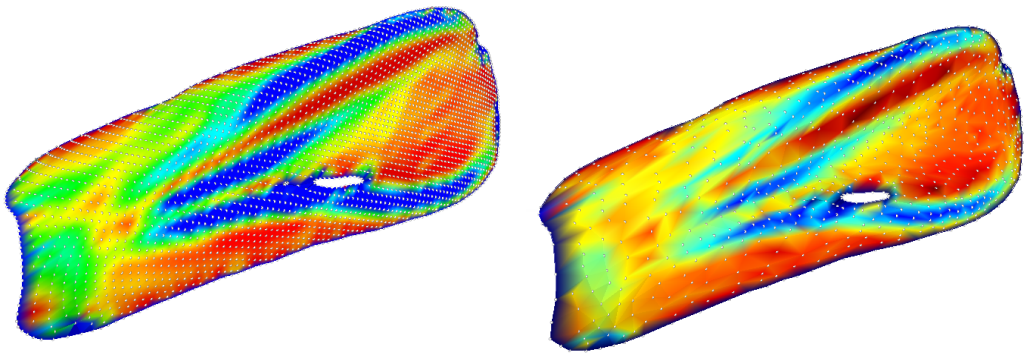
Questa serie di test adopera la *feature ground truth* per verificare la consistenza del framework con il modello teorico. Riassumiamo di seguito lo svolgimento del test, già descritto nella sezione 3.4.3.

Per prima cosa le range map vengono allineate approssimativamente a mano, e poi l'allineamento viene raffinato con ICP. A questo punto viene applicato il colore per ogni vertice delle range map. Per determinare il colore dei vertici si usa la funzione *perlin noise*, una funzione che dipende dalla posizione dei vertici nello spazio. L'informazione sul colore per vertice viene utilizzata dalla *feature ground truth* come descrittore.

Eseguire questa procedura per colorare le range map ci garantisce che a punti vicini nello spazio sia assegnato un colore simile; ciò è fondamentale in quanto la condizione sopracitata costituisce un requisito senza il quale

la fase di *matching* non può assolutamente funzionare. Tuttavia la funzione *perlin noise* non offre nessuna garanzia che un particolare colore appaia in un unico punto della range map; al contrario è probabile che chiazze della stessa tonalità di colore si presentino in più parti della mesh. Quest'ultimo fatto ha un'impatto negativo sull'efficacia della fase di *matching* e ci fa presagire probabilità di successo non eccezionali.

La non univocità dei descrittori è una proprietà sensata e controllabile tramite la frequenza del *perlin noise*. Sarebbe stato possibile utilizzare una *feature ground truth* che garantisse l'univocità dei descrittori e comportasse quindi un *matching* sempre corretto; questo scenario oltre a essere del tutto irrealistico è anche di scarso interesse al fine della valutazione dell'affidabilità del *framework* poichè non fornisce nessuna indicazione circa il buon funzionamento della fase di *matching*.



(a) Alta risoluzione **(b)** Bassa risoluzione

Figura 5.2: *Due dettagli di una range map che mostrano la curvatura assoluta a due differenti risolzioni. (a) Dettaglio di una range map a alta risoluzione, circa 36000 vertici. (b) Dettaglio di una range map a bassa risoluzione, circa 7500 vertici. Si può notare che il numero di vertici all'interno delle zone blu è molto inferiore, fatto che comporta un aumento dell'efficacia del matching.*

I test sono stati eseguiti su delle range map semplificate a circa 10000 vertici. Ridurre il numero di vertici della mesh prima dell'allineamento aumenta l'efficacia della fase di *matching*, con conseguente miglioramento della probabilità di successo. Ciò accade perchè, per mesh con un gran numero di vertici, le zone con descrittori simili tra loro sono composte da centinaia di punti, dunque la *kNN search* effettuata durante la fase di *matching* rischia di 'esaurirsi' all'interno di una zona che non coincide con quella cercata [Fig. 5.2]. Il processo di semplificazione delle mesh non ha però impatti significativi sulla qualità dell'allineamento trovato; ciò significa che la trasformazione rigida ottenuta può essere applicata alle range map originali e a partire da questo allineamento si può ricavarne uno molto preciso con ICP.

Per verificare il comportamento del framework in presenza di rumore, il test è stato ripetuto dopo aver applicato una data quantità di rumore al colore di una delle due range map; invece per riprodurre diverse percentuali di *overlap* il test è stato ripetuto dopo aver asportato una porzione da una delle due range map. La prossima sezione riporta i risultati ottenuti.

5.3.3 Risultati

Il grafico 5.3 mostra i dati osservati sperimentalmente eseguendo batterie di 100 *run* per ogni valore del parametro n_{iter}. Le due range map contano circa 10000 vertici e hanno un *overlap* del 75%; il colore non contiene rumore.

Il primo test presentato è stato eseguito su un'intervallo molto ampio di iterazioni con il fine di mostrare che la probabilità di successo cresce fino a raggiungere il valore massimo, così come previsto dal comportamento ideale. La curva mostrata è stata calcolata interpolando i dati osservati per più ripetizioni del test. La linea così ottenuta aderisce bene al modello teorico.

Il grafici in figura 5.4 mostrano i dati osservati su un intervallo di iterazioni più piccolo. La figura 5.4a mostra l'andamento della probabilità di

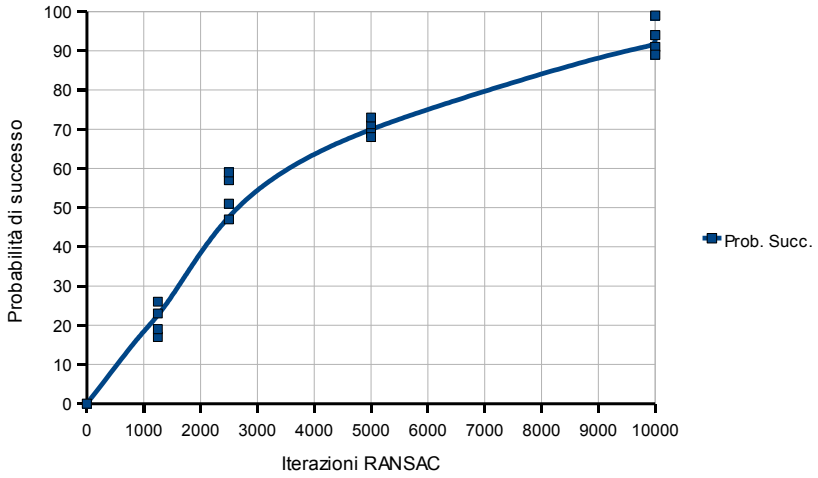
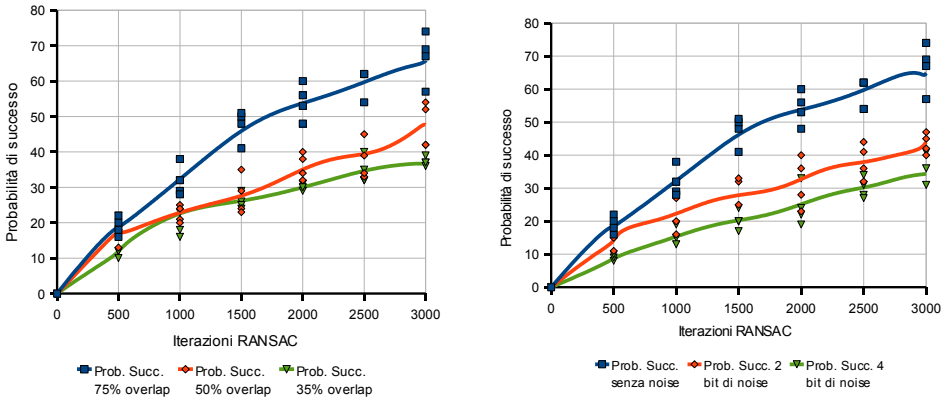


Figura 5.3: *Il grafico mostra la crescita della probabilità di successo in funzione del numero di iterazioni del RANSAC. La curva è stata estrapolata dai dati osservati per diverse ripetizioni del test, eseguiti su delle range map di circa 10000 vertici. Si può notare che la probabilità di successo tende al 100% all'aumentare delle iterazioni del RANSAC, come indicato dal modello teorico.*

successo per diversi valori di *overlap*, mentre la figura 5.4b mostra l'andamento per diverse quantità di rumore. Anche in questo caso i risultati ottenuti confermano le tendenze pronosticate dal modello teorico.

5.4 Prestazioni del framework

Dopo aver appurato di disporre di un framework funzionante e consistente con il modello teorico si può procedere a valutare le prestazioni del framework. Il framework è stato messo alla prova con range map appartenenti a diversi data set così da ricavare una valutazione quanto più completa; inoltre i test effettuati sono stati ripetuti per ogni tipo di feature implementata. Le grandezze prese in considerazione per valutare le prestazioni framework sono: il



(a) Diverse quantità di overlap **(b)** Diverse quantità di noise

Figura 5.4: *I risultati del test, che confermano la tendenza generale individuata dal modello teorico. (a) Il grafico riporta la crescita della probabilità di successo per situazioni con una zona di overlap sempre minore. (b) Il grafico riporta la crescita della probabilità di successo per situazioni con una crescente quantità di noise.*

tempo medio di completamento di n_{iter} iterazioni, la probabilità di successo dopo n_{iter} iterazioni e la probabilità di fallimento per secondo. Queste grandezze sono discusse nella prossima sezione.

5.4.1 Valutazione delle prestazioni

Ogni test eseguito fornisce le seguenti grandezze:

- ▷ Tempo medio di completamento di n_{iter} iterazioni di RANSAC.
- ▷ Probabilità di successo dopo n_{iter} iterazioni di RANSAC.
- ▷ Probabilità di fallimento per secondo.

La prima grandezza non ha bisogno di molte spiegazioni: vorremmo che il *framework* riuscisse ad allineare due scansioni nell'ordine dei secondi e

non dei minuti. Anche la seconda grandezza, già impiegata nei test per la verifica della consistenza, è di semplice comprensione: ci interessa sapere quante iterazioni di RANSAC bisogna eseguire per avere una probabilità di successo tale da consentirci di trovare un allineamento in un paio di tentativi. La probabilità di successo è stata calcolata effettuando 100 differenti *run* dell'intera procedura di allineamento, per ogni valore del parametro n_{iter}, e contando il numero dei *run* che hanno avuto successo. Inoltre ciascun gruppo di 100 *run* è stato ripetuto quattro volte, questo ci permette una percezione intuitiva della varianza dalla distribuzione dei valori osservati. I grafici mostrati nelle sezioni 5.4.2 e 5.4.3 sono stati disegnati interpolando i dati restituiti dalle svariate ripetizioni dei test. Meno intuitiva è l'utilità della terza grandezza, ossia la probabilità di fallimento per secondo.

Indicando con k la probabilità di trovare almeno un allineamento in una singola iterazione 5.4 e con t i secondi impiegati per completare N iterazioni di RANSAC, possiamo definire la probabilità di fallimento per secondo p_{fps} come:

$$p_{fps} = (1 - k)^{\frac{1}{t}} \tag{5.8}$$

Un comportamento corretto da parte del *framework* corrisponde a una probabilità di fallimento per secondo costante; se così non fosse significherebbe che la probabilità di successo non cresce in modo esponenziale rispetto alle iterazioni del RANSAC e questo sarebbe indice di un malfunzionamento del *framework*, come visto nella sezione 5.3.1. Inoltre il valore della probabilità di fallimento per secondo ci fornisce una stima dell'efficienza della fase di *matching*, poichè inversamente proporzionale al valore di ϕ.

Avremmo potuto ottenere lo stesso risultato considerando la probabilità di fallimento rispetto a un numero fissato di iterazioni, invece si è scelto di rapportarci all'unità di tempo per poter osservare anche gli effetti degli *overhead* dovuti all'inizializzazione delle strutture dati del *framework*. Infatti

durante le prime esecuzioni della procedura di allineamento devono essere allocate e inizializzate le strutture dati utilizzate; questo comporta un tempo di completamento maggiore nelle prime iterazioni. Tutto ciò si riflette sulla probabilità di fallimento per secondo che tende a stabilizzarsi dopo i primi *run*, come si può osservare in tutti i grafici riportati nelle sezioni 5.4.2 e 5.4.3.

Per ogni coppia di scansioni allineate saranno riportati i grafici delle tre grandezze sopracitate.

5.4.2 Risultati per la feature GMA Curvature

In questa sezione sono presentati i risultati ottenuti utilizzando la feature *GaussianMeanAbsolute Curvature* (in breve *GMA Curvature*, sezione 3.4.1). I test riportati riguardano tre diversi data set e sono stati condotti su range map semplificate a circa 10000 vertici. Ulteriori informazioni specifiche sono riportate nelle didascalie relative ai risultati dei singoli test. I dati osservati sono riportati a pagina 82 e seguenti.

Dalla analisi dei risultai ottenuti riceviamo innanzitutto un'ulteriore conferma del corretto funzionamento del framework e la constatazione che i tempi medi di esecuzione variano pochissimo e rimangono sempre sotto la soglia del secondo e mezzo, un valore più che soddisfacente.

Per quanto riguarda invece le probabilità di fallimento per secondo e le probabilità di successo per numero di iterazioni, si possono osservare scenari opposti: due data set [Fig. 5.5, 5.7] forniscono prestazioni eccellenti e conferiscono la quasi certezza di allineare le scansioni in un paio di tentativi; di contro altri data set [Fig. 5.6, 5.8] forniscono risultati sostanzialmente scarsi e insoddisfacenti.

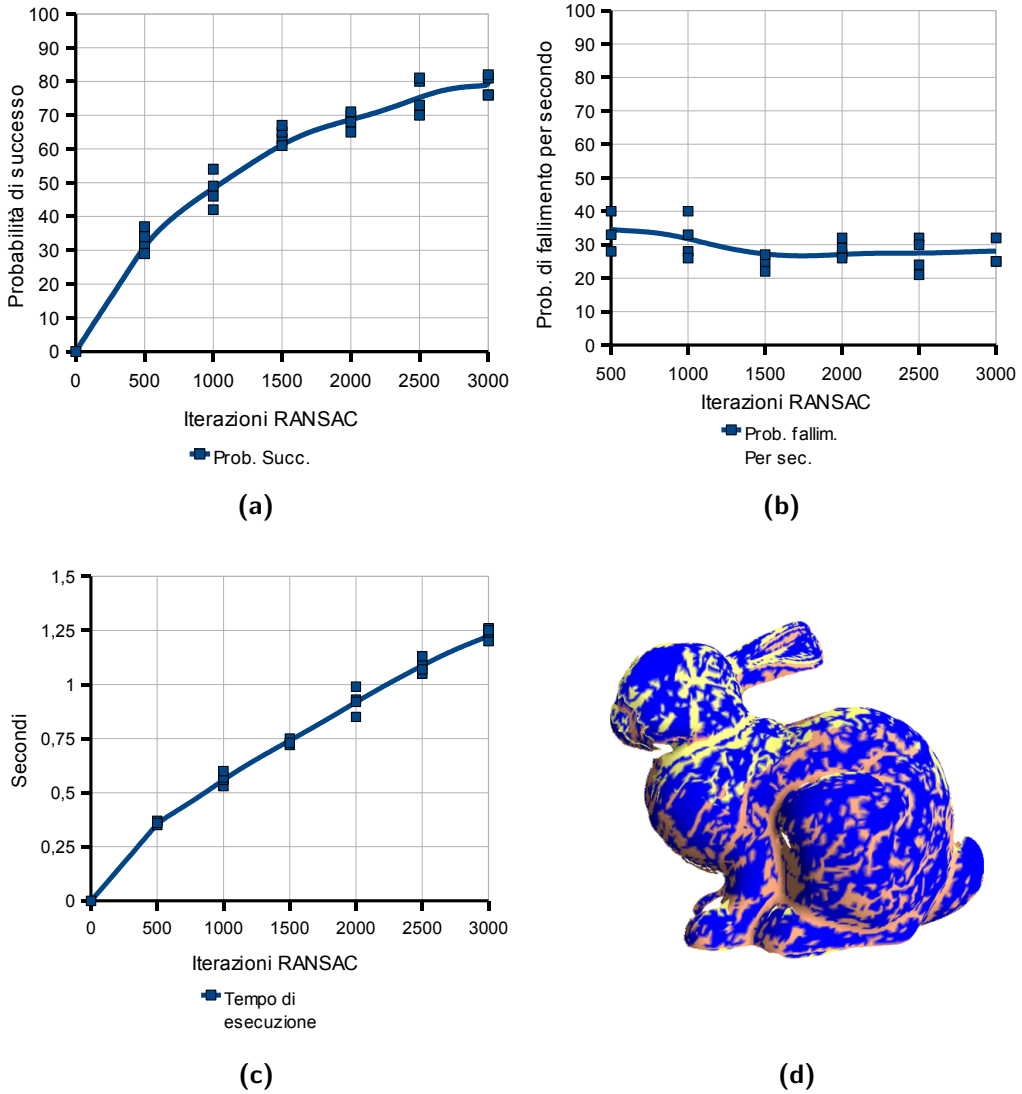


Figura 5.5: *Standford Bunny, GMA Curvature, 75% di overlap. Prestazioni ottime. I grafici mostrano che in poco più di un secondo è possibile allineare le scansioni con una probabilità dell'80%. La probabilità di fallimento per secondo si assesta su un valore basso, a indicare una buona efficienza del framework. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Un allineamento ottenuto; le parti in blu corrispondono ai descrittori.*

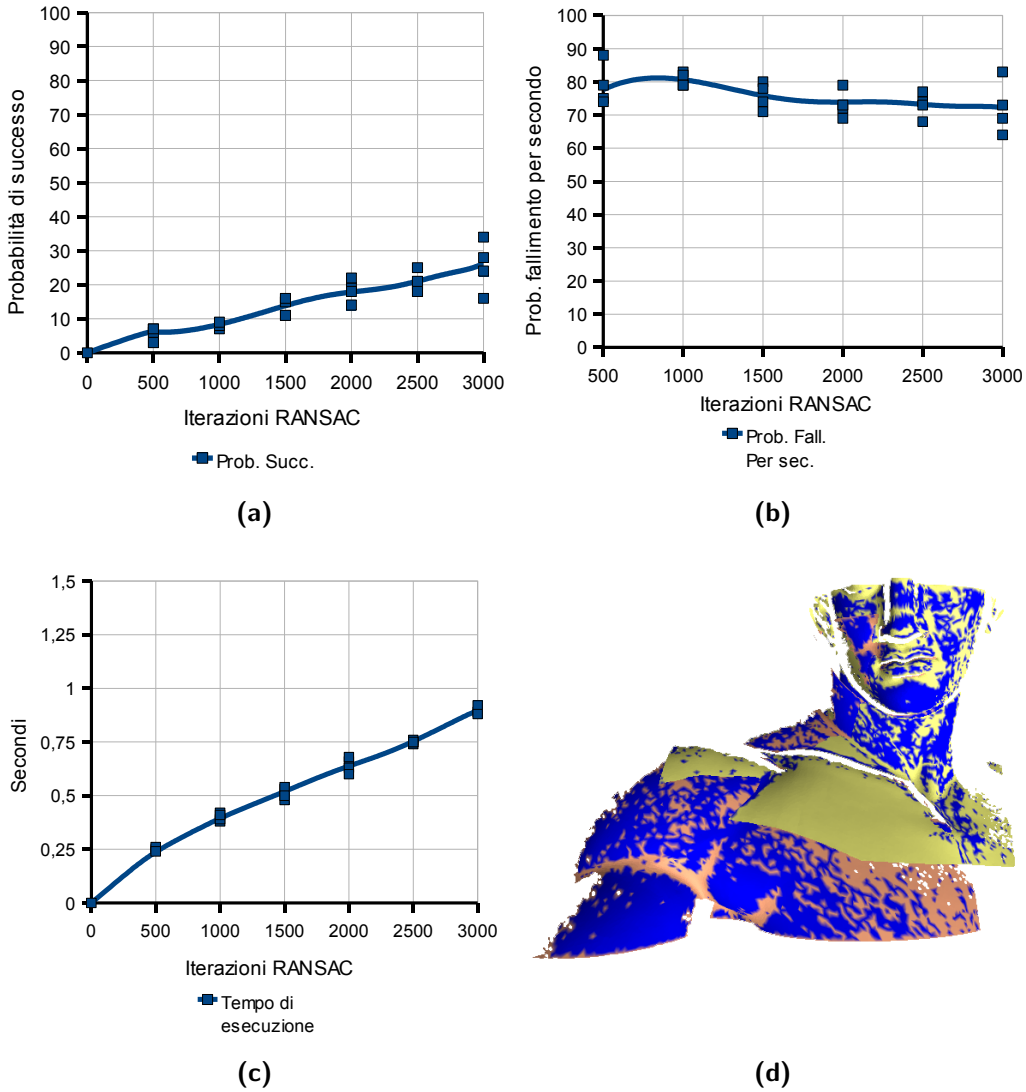


Figura 5.6: *Il Diadumeno, GMA Curvature, 75% di overlap. Prestazioni scadenti. La probabilità di successo raggiunta dopo 3000 iterazioni è bassa, conseguentemente la probabilità di fallimento per secondo si assesta su valori alti; i tempi di esecuzione, sempre sotto il secondo, sono molto buoni. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Un allineamento ottenuto; le parti in blu corrispondono ai descrittori.*

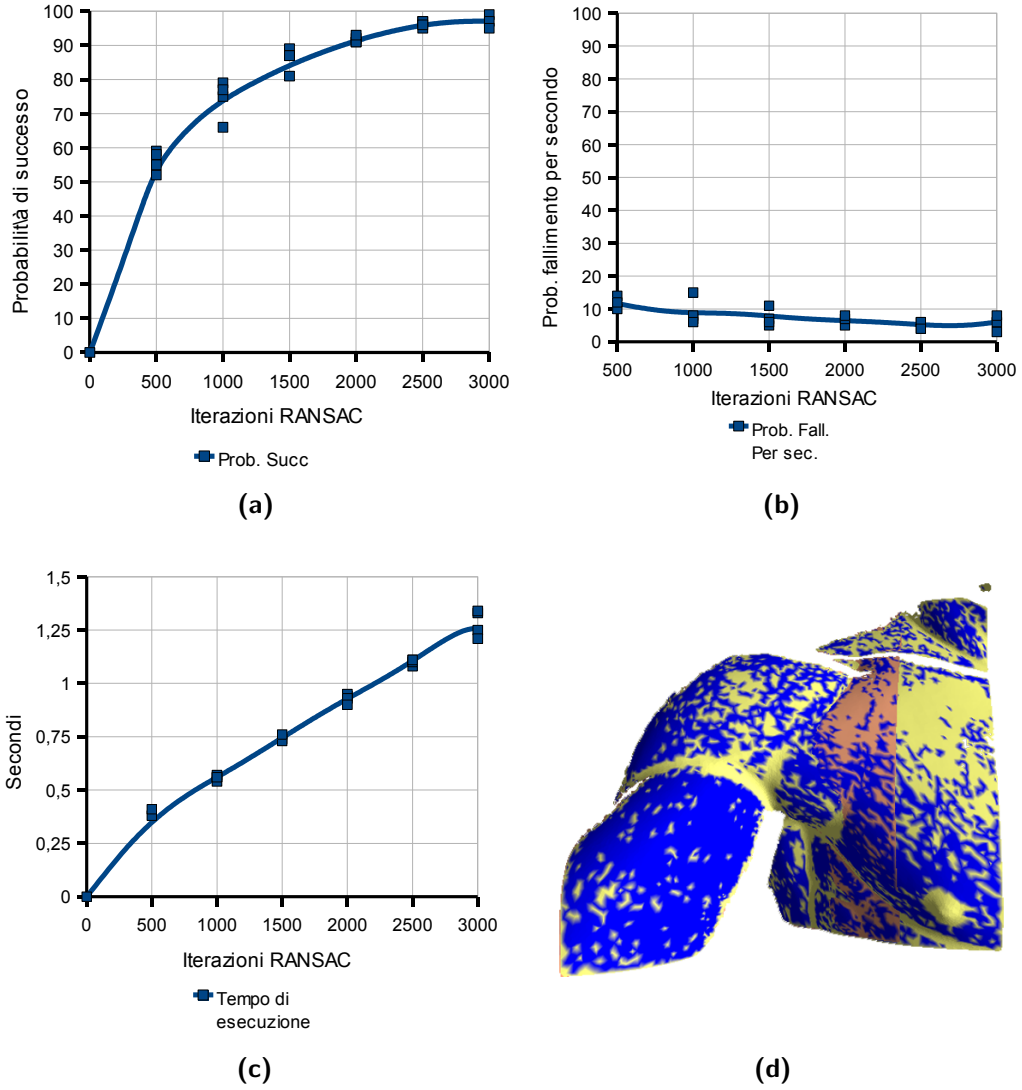


Figura 5.7: *Il Diadumeno, GMA Curvature, 80% di overlap. Prestazioni eccellenti; con 3000 iterazioni abbiamo quasi la certezza di allineare le scansioni in poco più di un secondo. La probabilità di fallimento rimane sotto il 10%. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Un allineamento ottenuto; le parti in blu corrispondono ai descrittori.*

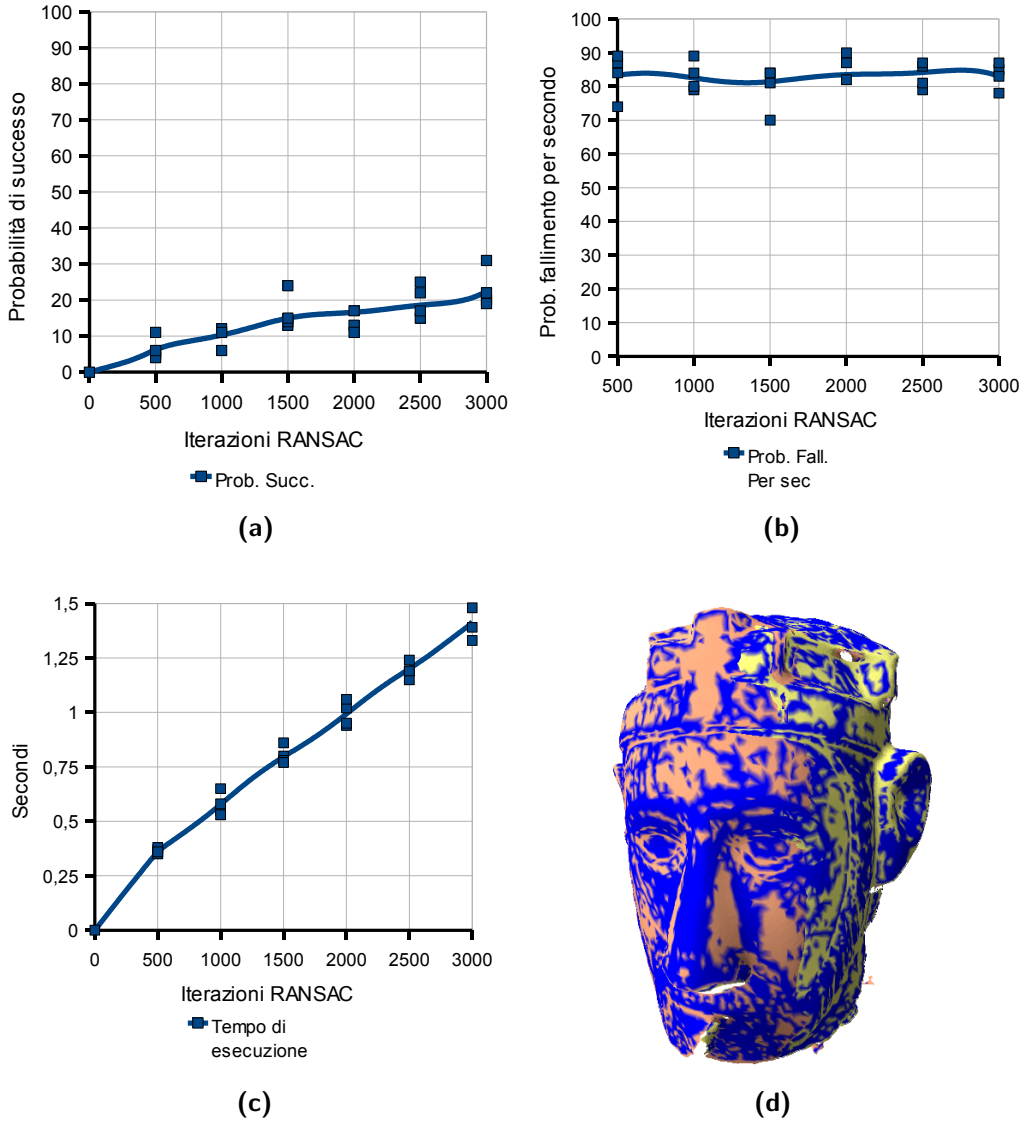


Figura 5.8: *King's head, GMA Curvature, 60% di overlap. Prestazioni scadenti. La probabilità di successo raggiunta dopo 3000 iterazioni è bassa, conseguentemente la probabilità di fallimento per secondo si assesta su valori alti, intorno all'80%; anche i tempi di esecuzione crescono leggermente sebbene non in modo significativo. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Un allineamento ottenuto; le parti in blu corrispondono ai descrittori.*

5.4.3 Risultati per la feature MS Curvature

In questa sezione sono presentati i risultati ottenuti utilizzando la feature *MeanSmooth Curvature* (in breve *MS Curvature*, sezione 3.4.2). I test riportati riguardano tre diversi data set e sono stati condotti su range map semplificate a circa 10000 vertici. Ulteriori informazioni specifiche sono riportate nelle didascalie relative ai risultati dei singoli test. I dati osservati sono riportati a pagina 88 e seguenti.

Dai risultati si può notare immediatamente che i tempi di esecuzione rimangono in linea con i test effettuati utilizzando la feature *GaussianMeanAbsolute Curvature*. Osservando invece le probabilità di successo ottenute per i diversi data set, notiamo che *Il Diadumeno* [Fig. 5.11] ha prestazioni eccellenti, mentre la *King's head* [Fig. 5.12] fornisce prestazioni insoddisfacenti. Si deve tuttavia tenere in considerazione che i risultati ottenuti sono del tutto simili a quelli conseguiti con la feature *GaussianMeanAbsolute Curvature* [Fig. 5.7, 5.8].

Gli altri due data set raggiungono una buona percentuale di successo nell'intervallo di iterazioni esaminato. Sebbene le prestazioni non siano ottime, abbiamo comunque la ragionevole certezza di pervenire a un allineamento in pochi tentativi, e quindi sempre nell'ordine di pochi secondi. Tuttavia il risultato più interessante è fornito dal confronto con i test eseguiti impiegando la feature *GaussianMeanAbsolute Curvature*. Lo *Standford Bunny* allineato per mezzo della feature *MeanSmooth Curvature* [Fig. 5.9] ottiene prestazioni inferiori all'allineamento ricavato con la feature *GaussianMeanAbsolute Curvature* [Fig 5.5]; al contrario *Il Diadumeno* ha prestazioni significativamente migliori se allineato con la feature *MeanSmooth Curvature* [Fig. 5.10, 5.6].

Dal confronto dei risultati forniti dalle feature implementate emerge dunque che l'alternanza di esperimenti con esiti positivi e negativi sia da attribuire, in via quasi esclusiva, alla 'qualità' dei descrittori selezionati, anziché

alla procedura di allineamento.

Intervenendo sulle feature e correggendo opportunamente le soglie che regolano la selezione dei valori dei descrittori (sezione 3.4.1) è possibile ottenere dei miglioramenti nei risultati. Tuttavia, come già menzionato nella sezione 3.3.2, gli istogrammi dei descrittori delle feature variano sensibilmente per ogni specifica coppia di scansioni; ideare un metodo generale che, tramite approfondite analisi statistiche, riesca a selezionare sempre intervalli dei descrittori significativi può considerarsi uno degli sviluppi futuri di questo lavoro.

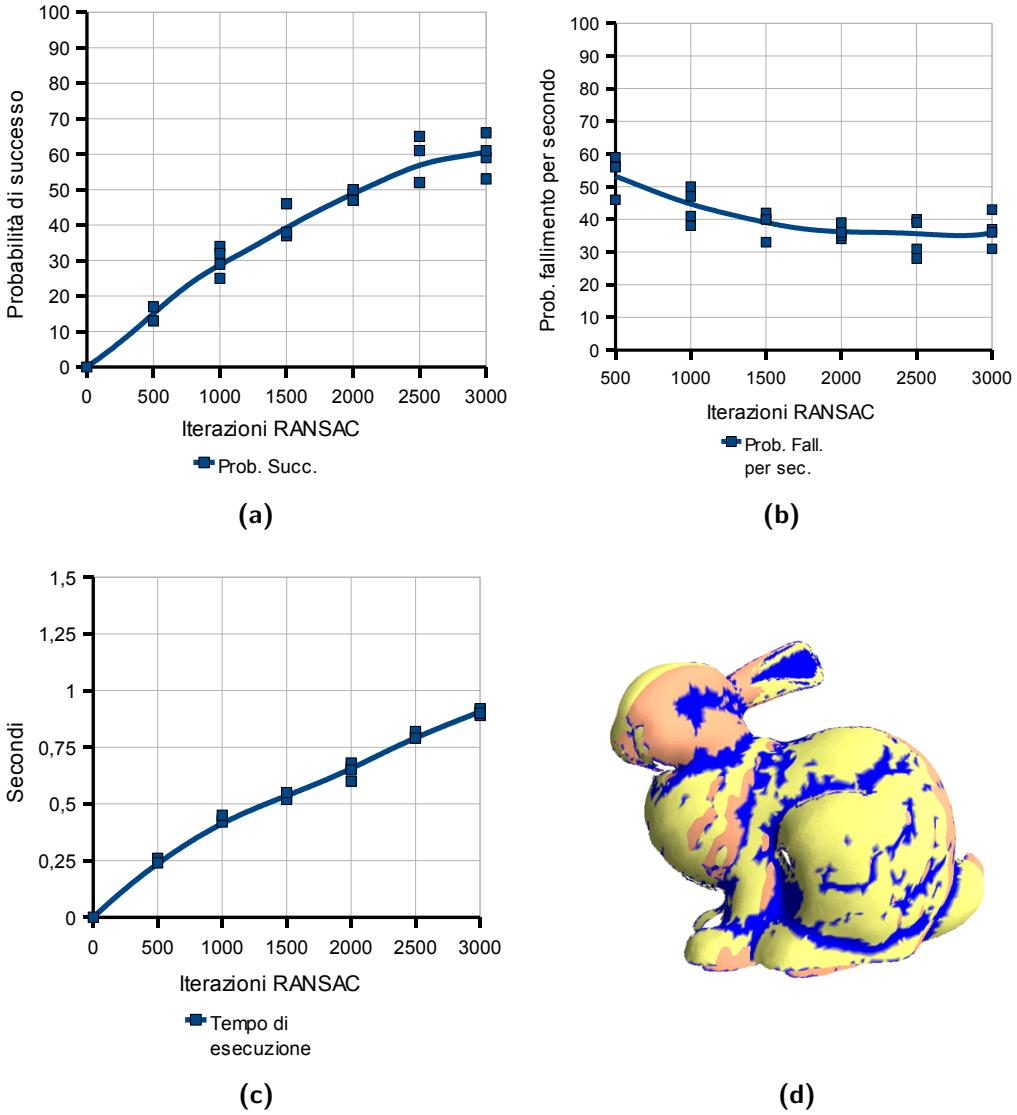


Figura 5.9: *Standford Bunny, MS Curvature, 75% di overlap. Prestazioni discrete. La probabilità di successo sale fino al 60% dopo 3000 iterazioni, il tutto in meno di un secondo. La probabilità di fallimento per secondo si assesta attorno al 35% dopo le 1500 iterazioni. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Un allineamento ottenuto; le parti in blu corrispondono ai descrittori.*

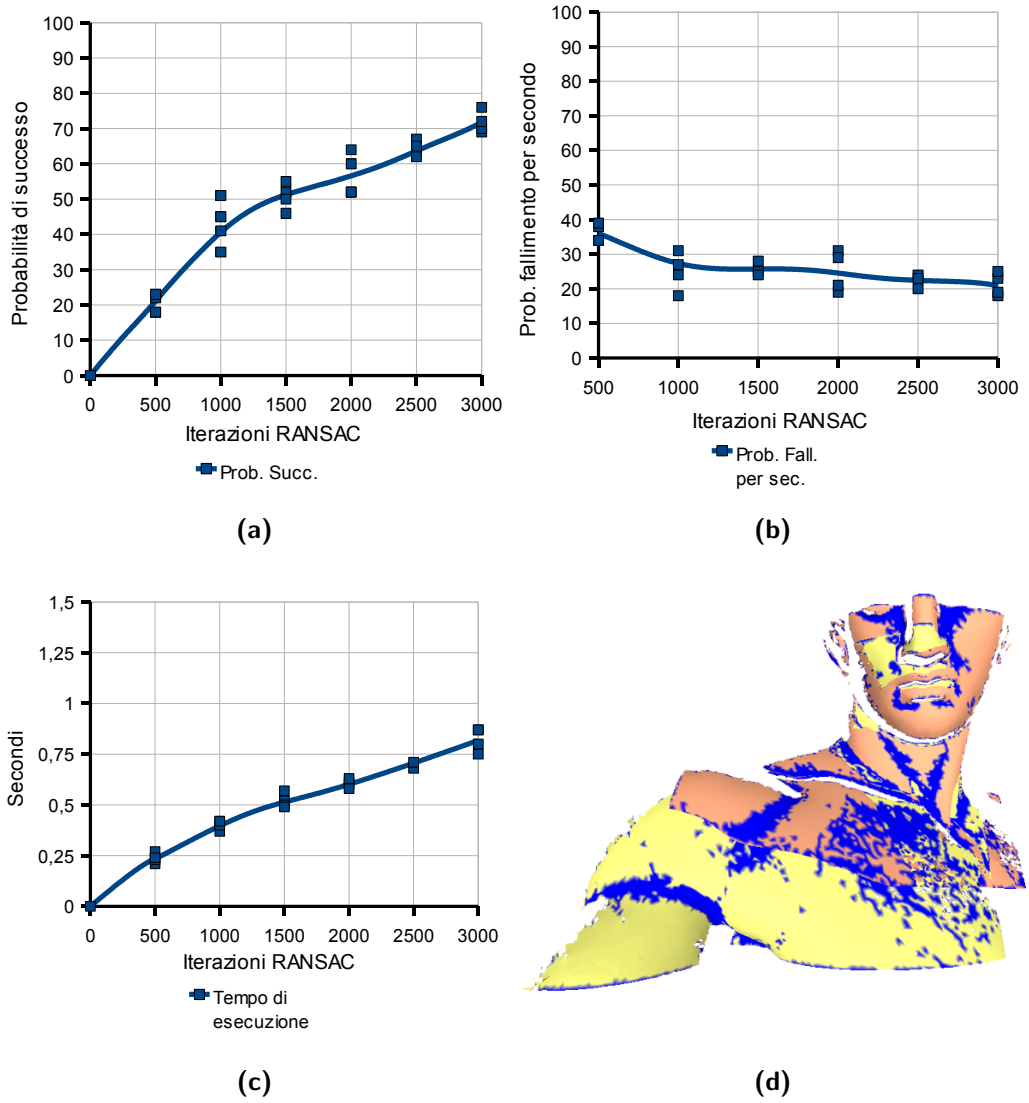


Figura 5.10: *Il Diadumeno, MS Curvature, 75% di overlap. Prestazioni buone. In meno di un secondo sono eseguite 3000 iterazioni con una probabilità di successo di circa il 70%; conseguentemente la probabilità di fallimento per secondo è stabile intorno al 25%. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Un allineamento ottenuto; le parti in blu corrispondono ai descrittori.*

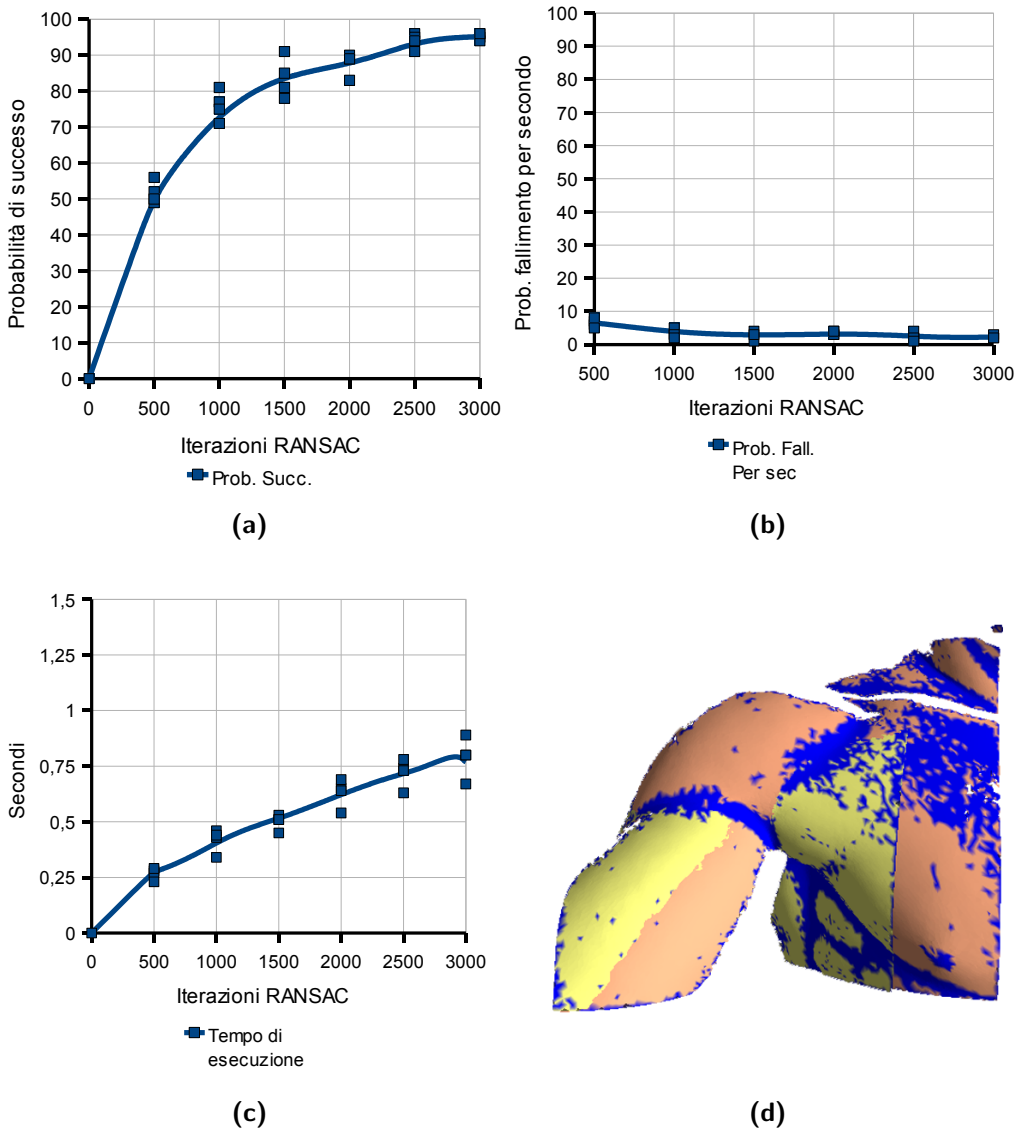


Figura 5.11: *Il Diadumeno, MS Curvature, 80% di overlap. Prestazioni eccellenti; già oltre le 2000 iterazioni la probabilità di successo supera il 90%. La probabilità di fallimento per secondo è sempre sotto il 10% e il tempo sotto il secondo. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Un allineamento ottenuto; le parti in blu corrispondono ai descrittori.*

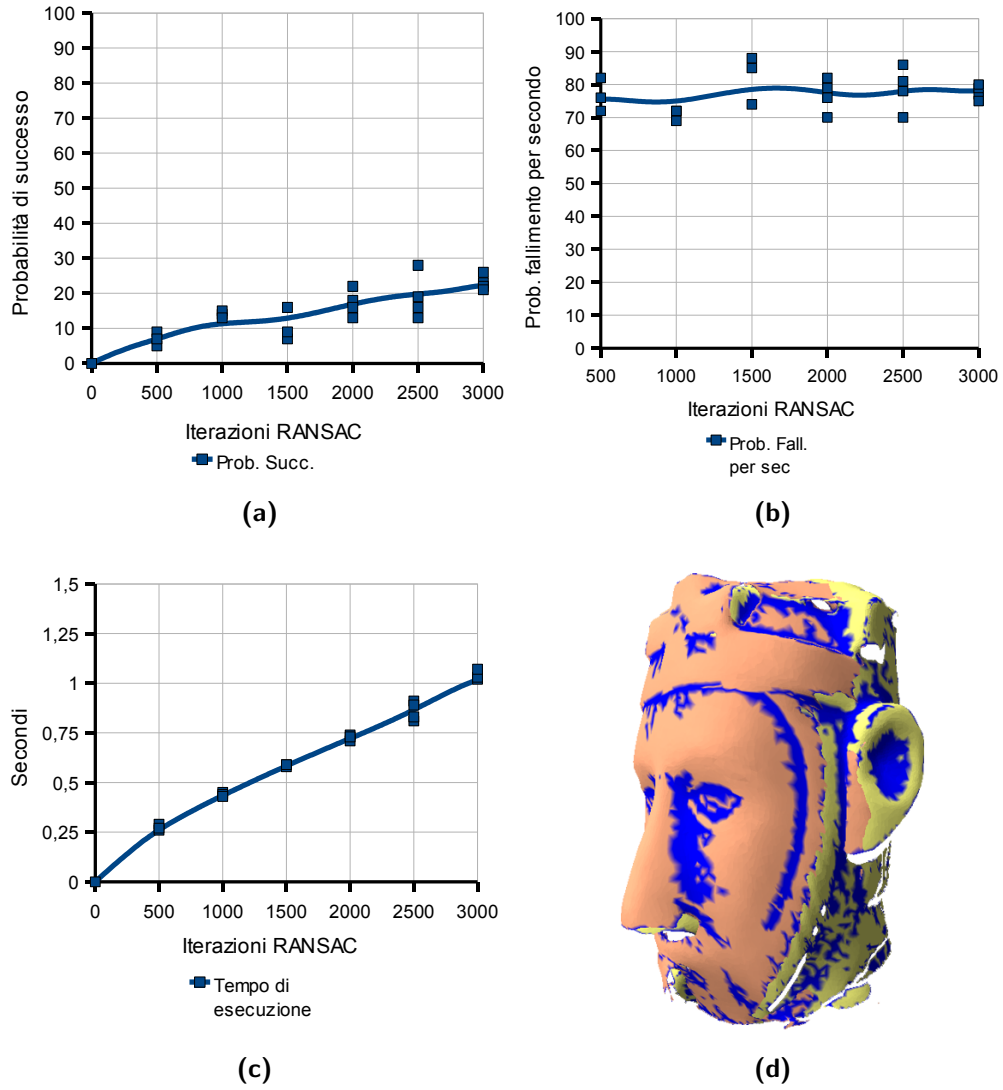


Figura 5.12: *King's head, MS Curvature, 60% di overlap. Prestazioni scadenti. La probabilità di fallimento per secondo oscilla tra il 70 e l'80%; la probabilità di successo raggiunge al massimo il 20% dopo 3000 iterazioni. I tempi di esecuzione sono nella norma. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Un allineamento ottenuto; le parti in blu corrispondono ai descrittori.*

5.5 Ulteriori esperimenti

Sfruttando la genericità e la flessibilità del framework implementato, sono stati condotti due esperimenti aggiuntivi. Il primo di questi si pone l'obbiettivo di verificare se il *fine tuning* dei parametri del framework può comportare miglioramenti significativi nelle prestazioni. Il secondo esperimento vuole invece verificare se la modifica della strategia con cui sono selezionati i 'migliori' descrittori comporta un'incremento della probabilità di successo del framework; nella fattispece la strategia *uniform sampling* sarà sostituita dalla strategia *Poisson disk sampling*. Nel seguito si entra nel dettaglio degli esperimenti eseguiti.

5.5.1 Tuning dei parametri

La sezione 5.2.2 riporta un elenco dei parametri del framework. Il massimo numero di consensi brevi eseguiti, indicato con G (sezione 3.8.1), è stato ritenuto, tra tutti i parametri, quello che può avere maggiore impatto sulle prestazioni del framework.

Questa considerazione deriva dal fatto che la fase di consenso, ed in particolare il numero di consensi effettivamente eseguiti, è quella che ha maggiori conseguenze sul tempo di completamento; d'altro canto eseguire più consensi significa esaminare più trasformazioni rigide candidate e quindi avere maggiori possibilità di trovare un allineamento. Per questo si è pensato di eseguire un apposito test per constatare se, regolando accuratamente il valore di G, si avesse qualche miglioramento in termini di efficienza.

Il test consiste nell'osservare l'andamento della probabilità di fallimento per secondo al variare del numero di iterazioni del RANSAC e del parametro G. In particolare, per far risaltare un eventuale cambiamento significativo nella tendenza dei valori osservati, a G sono stati attribuiti valori piccoli,

intermedi e grandi. I grafici ricavati dai test eseguiti su due diversi data set di scansioni sono riportati in figura 5.13.

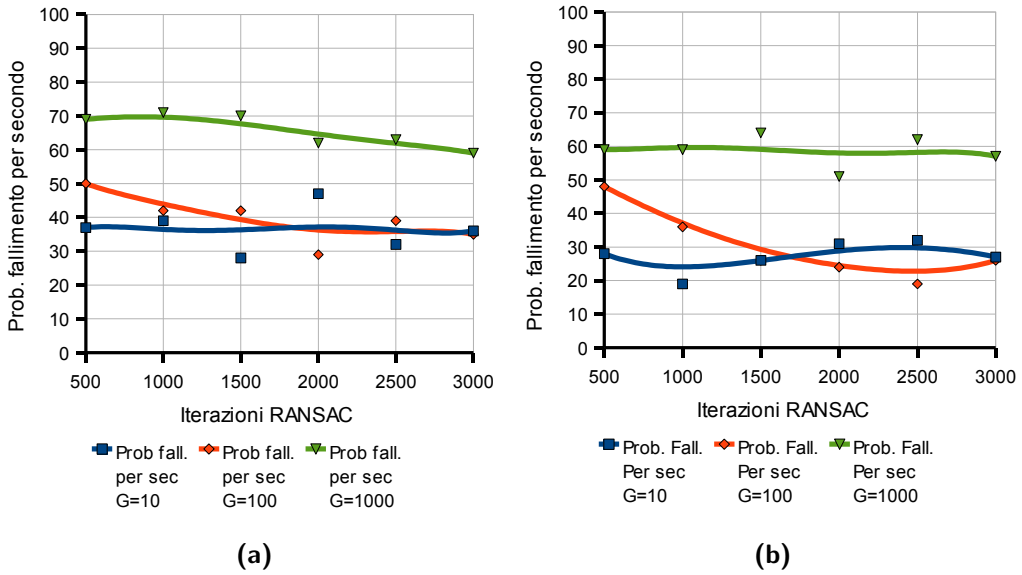


Figura 5.13: *Andamento della probabilità di fallimento per secondo in funzione del numero di iterazioni e del parametro G. In entrambi i casi si può osservare che, per valori del parametro compresi tra 10 e 100, le curve non si allontanano in modo significativo l'una dall'altra (linee blu e rosse). Valori molto alti comportano solo uno spreco tempo di calcolo (linea verde). (a) Standford Bunny, MS Curvature, 75% di overlap. (b) Il Diadumeno, MS Curvature, 75% di overlap.*

Dai risultati osservati si può concludere che le prestazioni del framework non risentono sensibilmente delle variazioni apportate al parametro G; in altri termini non ha senso investire tempo nel *fine tuning* di questo parametro (e degli altri). Tutto questo ci guida verso un'ulteriore importante risultato: le prestazioni della procedura di allineamento in sé risultano essere molto buone; l'introduzione di ulteriori sofisticazioni, a fronte di lunghi tempi di sviluppo, non comporterebbe miglioramenti sostanziali. I risultati variabili per quanto

riguarda gli allineamenti valutati nella sezioni 5.4.2 e 5.4.3, sono dunque da attribuire alla 'bontà' delle feature impiegate. Investire più tempo nell'analisi e nello sviluppo delle feature sembra essere la strada da intraprendere indicata dai nostri esperimenti per gli sviluppi futuri.

5.5.2 Poisson disk sampling

Nella sezione 3.3.2 abbiamo visto come i descrittori vengano vagliati per cercare di ridurne il numero e nel contempo mantenere solo quelli più rappresentativi della superficie. Nella stessa sezione si è anche spiegato che solo i punti associati a un piccolo sottoinsieme di descrittori, selezionati con una specifica strategia, vengono utilizzati per formare le basi relative alla mesh $\mathcal{M}_{fix}$. Per tutte le feature implementate è stata sin qui utilizzata la strategia *uniform sampling* [Fig. 3.2a]; ciò significa che tutti i descrittori hanno la stessa probabilità di essere estratti.

Questa strategia non offre garanzie sulla distribuzione spaziale dei punti. Ai fini dell'allineamento può essere interessante garantire che i descrittori selezionati siano uniformemente distribuiti in tutte le zone 'salienti' della mesh, proprietà che può essere soddisfatta adoperando la strategia *Poisson disk sampling* [Fig. 3.2b]. Questa seleziona solo descrittori spazialmente più distanti tra loro di una determinata distanza d. Naturalmente, all'aumentare del numero di descrittori che si desidera selezionare la distanza che può essere garantita diminuisce.

Sono stati eseguiti degli esperimenti per verificare se l'impiego del *Poisson disk sampling* portasse a miglioramenti effettivi delle prestazioni del framework. I grafici riportati in figura 5.14 e 5.15 mostrano i dati osservati dai test condotti.

I risultati stabiliscono che l'impiego della strategia *Poisson disk sampling* non comporta miglioramenti significativi, al contrario si riscontra un peggio-

ramento delle prestazioni. Infatti, mentre la complessità di questa tecnica di sampling produce un importante aumento del tempo di calcolo, il numero di allineamenti ottenuti non cresce rispetto all'impiego della tecnica *uniform sampling*.

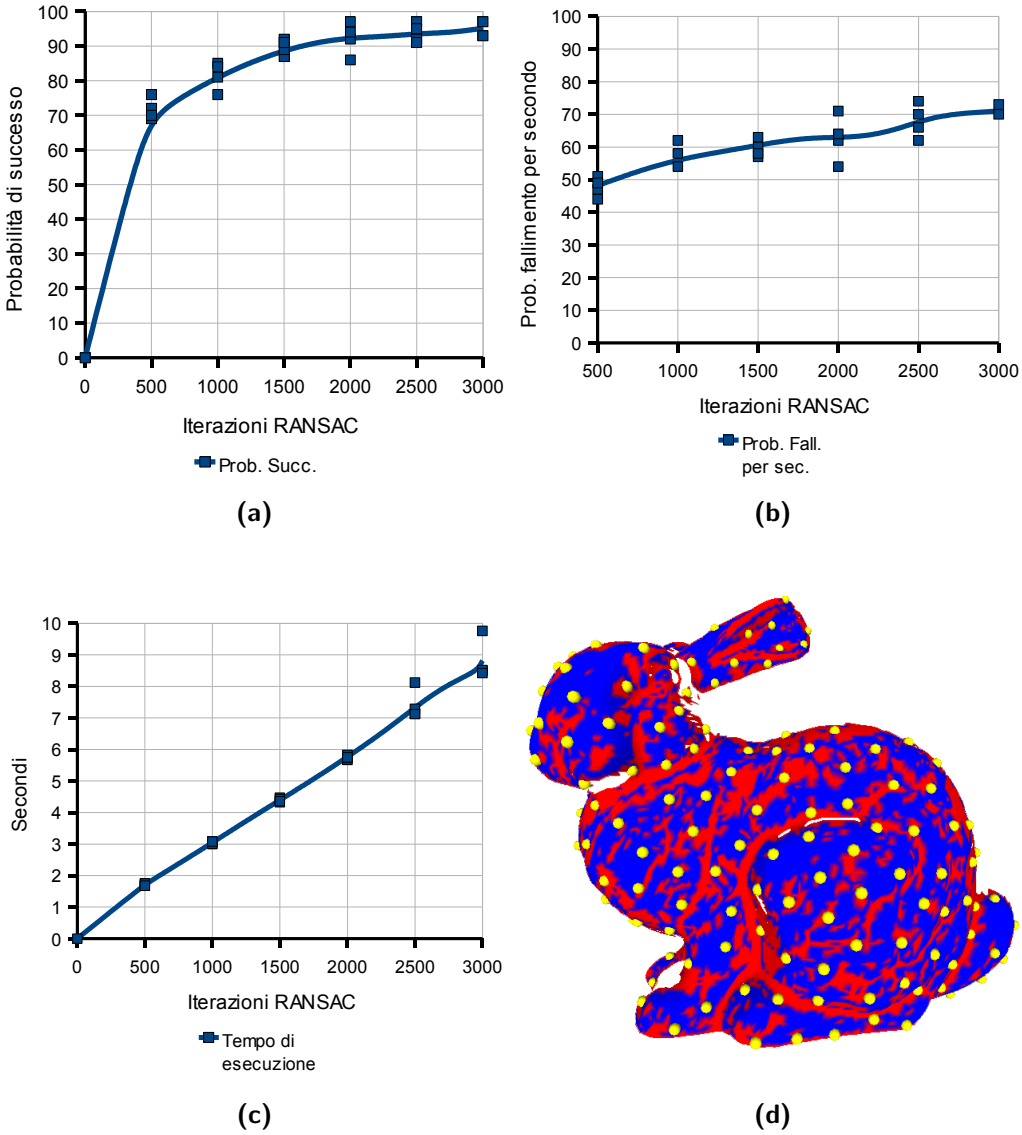


Figura 5.14: *Standford Bunny, GMA Curvature, 75% di overlap. Si riscontra un peggioramento delle prestazioni rispetto al test condotto con uniform sampling [Fig. 5.5]. La probabilità di successo non cambia, ma sia il tempo di esecuzione che la probabilità di fallimento per secondo aumentano considerevolmente. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Punti selezionati per la formazione delle basi sulla mesh $\mathcal{M}_{fix}$.*

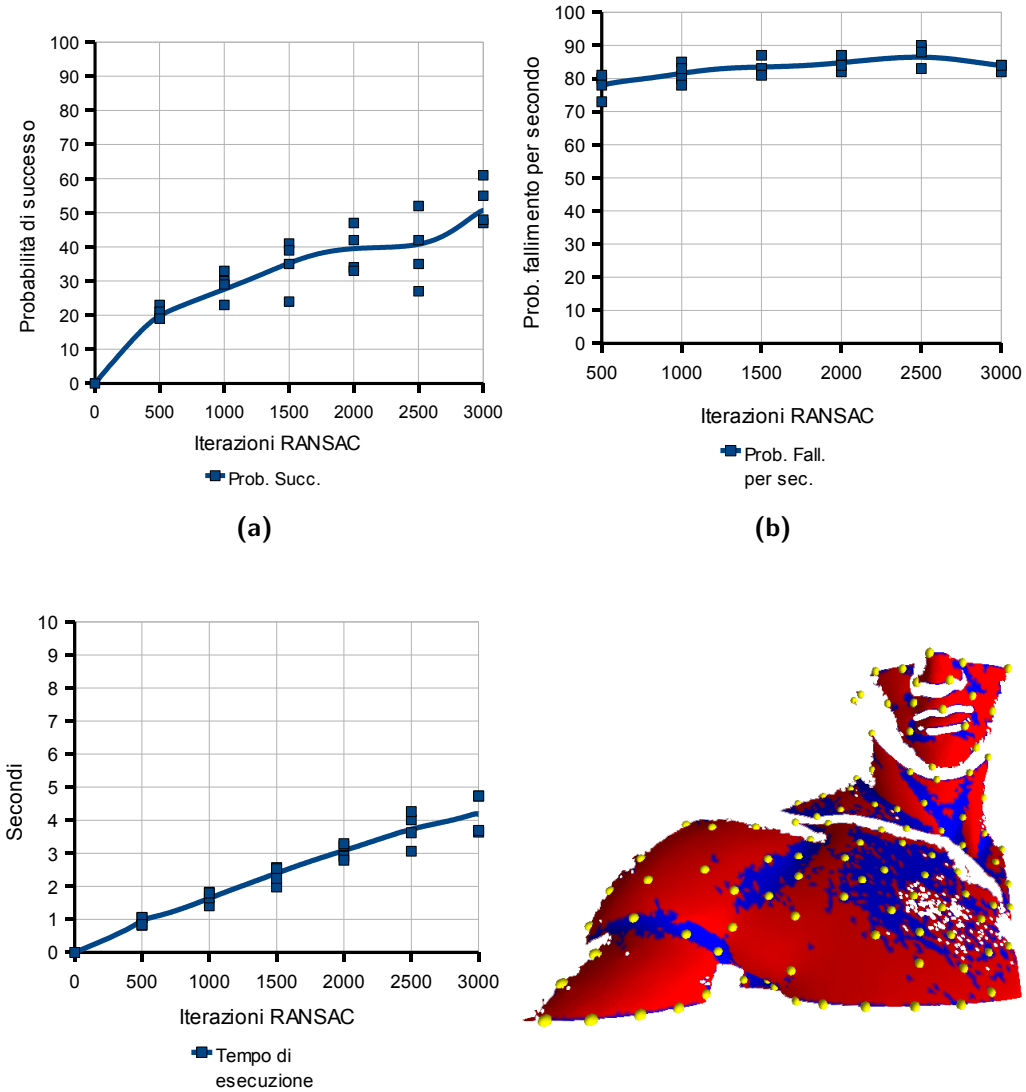


Figura 5.15: *Il Diadumeno, MS Curvature, 75% di overlap. Si riscontra un peggioramento delle prestazioni rispetto al test condotto con uniform sampling [Fig. 5.10]. La probabilità di successo diminuisce e anche il tempo di esecuzione e la probabilità di fallimento per secondo in modo significativo. (a) Probabilità di successo al variare del numero di iterazioni. (b) Probabilità di fallimento per secondo. (c) Tempo medio impiegato per completare un allineamento. (d) Punti selezionati per la formazione delle basi sulla mesh $\mathcal{M}_{fix}$.*

5.6 Conclusioni

Dopo aver esaminato tutti i risultati dei test presentati in questo capitolo possiamo concludere traendo una valutazione complessiva del framework.

Innanzitutto possiamo asserire che il framework è consistente con le valutazioni teoriche effettuate (sezione 5.3.3); inoltre abbiamo anche stabilito che il framework presenta già un buon livello di sofisticazione, per cui ulteriori interventi sul procedimento di allineamento, come il *tuning* dei diversi parametri o l'introduzione di raffinamenti nelle fasi del RANSAC, non comporterebbero miglioramenti tali da giustificare il tempo impiegato nello sviluppo (sezioni 5.5.1, 5.5.2).

Utilizzando le feature da noi create per allineare diverse coppie di range map appartenenti a data set di scansioni reali, abbiamo potuto osservare che in molti casi il framework fornisce prestazioni soddisfacenti e perviene all'allineamento desiderato in pochi secondi [Fig. 5.5, 5.7, 5.9, 5.10, 5.11], mentre in un certo numero di situazioni occorrono svariati tentativi per ottenere un allineamento [Fig. 5.6, 5.8, 5.12]. Tuttavia il confronto diretto tra i risultati ottenuti dalle due feature ci porta a pensare che gli esiti insufficienti siano da attribuire alla natura delle feature, vale a dire che in alcuni casi la feature non fornisce valori dei descrittori significativi.

Quest'ultima osservazione conferma l'importanza di avere a disposizione un framework per l'allineamento di range map generico, flessibile e del tutto indipendente dalle feature, proprio come quello da noi sviluppato.

Gli sviluppi futuri prevedono:

- ▷ la realizzazione di nuovi e diversi tipi di feature;
- ▷ l'impiego del framework per il confronto dei risultati ottenuti dalle nuove feature su un vasto campione di data set e in presenza di diversificate quantità di *overlap* e di *noise*;

- ▷ l'ideazione di una sofisticata tecnica basata sull'elaborazione statistica degli istogrammi dei valori dei descrittori che consenta di selezionare sempre i descrittori più rapprsentativi della superficie per ogni coppia di range map considerata.

Capitolo 6

Conclusioni

La tesi ha descritto la realizzazione di un generico *framework* per l'allineamento di *range map* automatico. Il *framework* utilizza un approccio *RANdom SAmple Consensus* per valutare la qualità degli allineamenti prodotti a partire dall'accoppiamento di *feature* corrispondenti e selezionare il migliore di questi. Per ricavare un allineamento si selezionano delle quadruple di punti caratteristici su una scansione e si cercano tutte le possibili quadruple corrispondenti sull'altra scansione. Per stabilire le corrispondenze tra i punti delle due scansioni si sfruttano delle *feature*, ossia dei valori che descrivono in modo locale e invariante alle trasformazioni rigide la superficie delle range map. Tutte le trasformazioni rigide calcolate dalle quadruple di punti coincidenti vengono così valutate per selezionare quella che meglio allinea le due scansioni.

Nel contesto di questa strategia di allineamento sono possibili numerose varianti sia in termini di *feature* che in termini dei parametri usati per generare e valutare gli allineamenti trovati. Per questo motivo è stato analizzato il concetto di *feature* e ne è stata data una interfaccia generale così da consentire la realizzazione e il confronto di *feature* diverse e da conseguire una totale indipendenza tra *framework* e *feature*.

Contemporaneamente allo sviluppo del *framework* di allineamento sono state implementate anche una feature *ground truth* e due *feature* basate su di-

versi tipi di curvatura. La prima è stata utilizzata per verificare la correttezza sperimentale e la consistenza del *framework*, le altre sono state impiegate e confrontate per darne una valutazione oggettiva delle prestazioni.

Per valutare il *framework* è stato elaborato un modello teorico che tiene conto delle caratteristiche del RANSAC e della bontà della feature utilizzata. Il *framework* è stato sottoposto a svariati test sia per verificarne sperimentalmente la conformità al modello teorico elaborato, sia per valutare le effettive prestazioni su data set di scansioni reali utilizzando le *feature* implementate. Con ulteriori batterie di test si sono osservati anche gli effetti del *tuning* dei principali parametri del *framework*.

In conclusione è stato accertato che il *framework* di allineamento proposto ed implementato funziona correttamente ed è consistente con il modello teorico elaborato. Le impostazioni di *default* dei molti parametri del *framework* garantiscono una buona *performance* ed è stato appurato che un ulteriore *tuning* dei parametri e l'introduzione di altre ottimizzazioni non comporterebbero cambiamenti significativi nelle prestazioni.

Le *feature* implementate hanno fornito risultati soddisfacenti per molte delle coppie di *range map* esaminate. Il confronto tra le prestazioni ottenute dai diversi tipi di *feature* ha chiarito che quest'ultime costituiscono l'elemento principale su cui intervenire per ottenere dei forti miglioramenti nelle prestazioni.

L'elaborazione di nuove e potenti *feature*, lo sviluppo di tecniche che ne permettano un migliore sfruttamento e lo studio comparativo delle prestazioni da esse ottenute costituiscono i principali obiettivi degli sviluppi futuri.

In ultima analisi, l'esito del lavoro svolto conferma l'importanza di uno strumento come quello implementato: un *framework* per l'allineamento automatico, generico e del tutto indipendente dalle *feature*, che possa essere utilizzato per conseguire una più profonda conoscenza delle caratteristiche

delle *feature* proposte in letteratura e giungere così ad un avanzamento dello stato dell'arte.

In futuro, come naturale evoluzione di questo lavoro, verranno realizzate diverse altre *feature* già esistenti in letteratura e sarà utilizzato il *framework* di allineamento sia per valutare separatamente l'efficacia di ogni *feature*, sia per confrontare tra loro varie *feature*. Inoltre sarà avviato lo studio di un metodo statistico più sofisticato che permetta di estrarre sempre i descrittori più significativi qualunque sia la scansione considerata. Per migliore ulteriormente la *performance* del *framework*, le parti computazionalmente più dispendiose saranno affidate interamente alla GPU.

Bibliografia

[AK04] N. Amenta and Y. J. Kil. Defining point-set surfaces. *ACM Trans. Graph.*, 23(3):264–270, 2004.

[AMCO] Dror Aiger, Niloy J. Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration.

[AMS+98] S. Arya, Mount D. M., Netanyahu N. S., Silverman R., and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimension. *ACM*, 45(6):891–923, 1998.

[BJ86] P.J. Besl and R.C. Jain. Invariant surface characteristics for 3d oject recognition in range images. *CVGIP: Image Understanding*, 33(1):33–80, Jan 1986.

[BM92] P. J. Besl and N. D. McKay. A method for registration of 3d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[BMR+99] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for

surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5, October-December 1999.

[BR02] Fausto Bernardini and Holly Rushmeier. The 3d model acquisition pipeline. *COMPUTER GRAPHICS forum*, 21(2):149–172, 2002.

[BSGL96] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a general multiview registration technique. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):540–547, May 1996.

[CM92] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *International Journal of Image and Vision Computing*, 10(3):145–155, April 1992.

[Cur99] Brian Curless. From range scans to 3d models. In *Computer Graphics*, volume 33. Nov 1999.

[DWJ97] C. Dorai, J. Weng, and A. K. Jain. Optimal registrationof object views using range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1131–1138, October 1997.

[FB81] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[FHK$^+$] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bulow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors.

[Gar99] M. Garland. Multiresolution modelling: survey and future opportunities. In *State of the Art Report (STAR)*. Eurographics '99, 1999.

[GMGP05] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *Eurographics Symposium on Geometry Processing*, 2005.

[GMO94] M. T. Goodrich, J. S. B. Mitchell, and M. W. Orletsky. Practical methods for approximate geometric pattern matching under rigid motions. In *Symp. on Computational Geometry*, pages 103–112, 1994.

[HFG+] Qi-Xing Huang, Simon Flory, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching.

[Hor87] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4:629–642, 1987.

[HSIW96] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *Fourth European Conference on Computer Vision*, pages 117–126, 1996.

[IR96] S. Irani and P. Raghavan. Combinatorial and experimental results for randomized point matching algorithms. In *Symposium on Computational Geometry*, pages 68–77, 1996.

[JH99] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, May 1999.

[JNHL04] Mitra N. J., Gelfand N., Pottmann H., and Guibas L. Regist. of point cloud data from a geometric opt. perspective. *Geometry Processing*, 2:23–32, 2004.

[LC87] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–170, 1987.

[LG05] Xinju Li and Igor Guskov. Multi-scale features for approximate alignment of point-based surfaces. In *Eurographics Symposium on Geometry Processing*, 2005.

[LGB] Xinju Li, Igor Guskov, and Jacob Barhak. Robust alignment of multi-view range data to cad model.

[MY95] T. Masuda and N. Yokoya. A robust method for registration and segmentation of multiple range images. *Computer Vision and Image Understanding*, 61(3):295–307, May 1995.

[OB08] A. Cengiz Oztireli and Cagatay Basdogan. A new feature-based method for robust and efficient rigid-body registration of overlaping point clouds. *Visual Comput*, 24:679–688, 2008.

[PFC+05] Paolo Pingi, Andrea Fasano, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. Exploiting the scanning sequence for automatic registration of large sets of range maps. *EUROGRAPHICS 2005*, 24(3), 2005.

[Pul99] K. Pulli. Multiview registration of large data sets. In *In Proc 2nd Int.l Conf. on 3D Digital Imaging and Modeling*, pages 160–168. IEEE, 1999.

[RL] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm.

[Rus04] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization and Transmission.*, 2004.

[RZ03] R.Hartley and A. Zisserman. *Multiple View Geometry in computer vision.* Cambridge University Press, 2003.

[Sco03] Roberto Scopigno. 3d scanning: potenzialit e limiti delle tecnologie di acquisizione automatica. *Disegno Digitale e Design*, 718(5), Gen-Mar 2003.

[SL92] M. Soucy and D. Laurendeau. Multi-resolution surface modeling from multiple range views. In *Proceedings of CVPR'92*, pages 348–353, June 1992.

[SLW00] Gregory C. Sharp, Sang W. Lee, and David K. Wehe. Icp registration using invariant features. September 2000.

[Tau95a] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 902. IEEE Computer Society, 1995.

[Tau95b] Gabriel Taubin. A signal processing approach to fair surface design, 1995.

[TL94] Greg Turk and Mark Levoy. Zippered polygon meshes from range images. In *Computer Graphics, SIGGRAPH 94 Conference Proceedings*, pages 311–318, 1994.

[WSI98] M. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. In *Sixth International Conference on Computer Vision*, pages 917–924. IEEE, 1998.

[Zha94] Z. Y. Zhang. Iterative point matching for registration of free form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, October 1994.