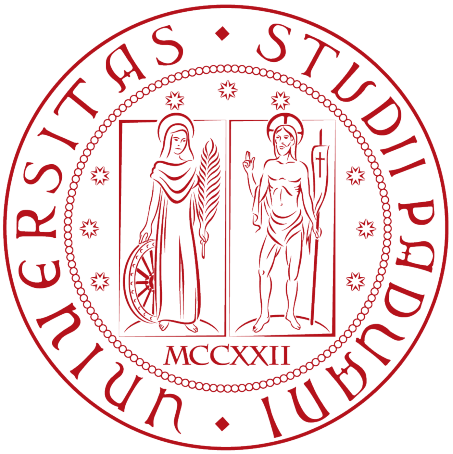
Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica



Ultimo Tango a Mountain View

Tesi di laurea triennale

Relatore

Prof. Gilberto Filè

Laureando

Tommaso Padovan

Anno Accademico 2015-2016

xxxx frase fica

— xxxx autore

Dedicato a ... xxxx

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecentoventi ore, dal laureando Tommaso Padovan presso l'azienda Vic srl. Gli obbiettivi da raggiungere erano principalmente la progettazione e la codifica di un prototipo di applicazione in grado di sfruttare gli innovativi dispositivi *Tango* di Google per produrre scansioni tridimensionali degli oggetti inquadrati.
In primo luogo era richiesto lo studio delle soluzioni *OpenSource* già presenti nel mercato, al fine di massimizzare il riuso. In secondo luogo era richiesta l'ideazione e la progettazione di una applicazione in grado di registrare ed eleborare i dati catturati dai sensori del *tablet* utilizzando le *API Tango* offerte da *Google*. Il terzo obbiettivo era la progettazione e codifica di una applicazione dotata di una interfaccia grafica minimale, ma capace di ricostruire gli oggetti inquadrati ed inviare i dati, in formato *Point Cloud* (*.pcd*), ad un *Server*. In ultimo luogo era richiesto lo sviluppo dell'applicazione lato *Server* allo scopo di effettuare ottimizzazioni sui *Point Cloud* ed il calcolo del volume. Tali operazioni sono state realizzate lato *Server* in quanto sarebbero troppo onerose per un dispositivo *mobile*.

xxxx "Life is really simple, but we insist on making it complicated"

— xxxx Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Gilberto Filè, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Sept 2016 Tommaso Padovan

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

xxxx Introduzione al contesto applicativo.

Esempio di utilizzo di un termine nel glossario
Application Program Interface (API).

Esempio di citazione in linea
site:agile-manifesto

Esempio di citazione nel pie' di pagina
citazione[1]

1.1 L'azienda

VIC è stata fondata da Alessio Bisutti che, dopo aver sviluppato una lunga esperienza nel campo ispettivo, ha deciso di costituire una società in grado di offrire ai propri clienti un servizio professionale, chiaro ed affidabile, appoggiandosi sulle nuove tecnologie.
VIC iniziò a Venezia 7 anni fa come piccola società di ispezione locale ed ora il gruppo VIC è uno dei più grandi attori del mercato globale.
Fin dall'inizio, l'obiettivo principale di VIC è stata la riduzione del tempo tra ispezione e reporting al cliente. Ora l'obiettivo è raggiunto, perché VIC sta fornendo ai suoi clienti tutti i risultati e le informazioni importanti in tempo reale, senza alcun ritardo, grazie agli investimenti fatti nel campo della tecnologia e delle applicazioni mobile.
VIC è la prima ed unica azienda in campo ispettivo ad offrire un'ampia gamma di servizi tecnologici a completa disposizione dei propri clienti.

1.2 L'idea

Mansioni come determinare la corretta forma, peso, quantità e dimensioni degli oggetti da ispezionare sono tra le più importanti per i controlli effettuati dall'azienda.
Gli ispettori possono scattare molte fotografie, prendere appunti e sfruttare la loro esperienza per fornire stime accurate; si è manifestata però la necessità di affiancare queste ultime a dei dati quanto più possibile oggettivi e rapidi da ottenere.

[1] **womak:lean-thinking**.

Da qui nasce l'idea di fornire agli ispettori uno strumento informatico in grado di effettuare queste stime. Grazie alla ricostruzione computerizzata resa disponibile dai *Tango device* sarà possibile non solo visualizzare su uno schermo il modello 3D del soggetto della ispezione, ma anche ottenere ulteriori vantaggi quali:

* Avere una stima del volume e quindi del peso della materia prima.
* Confrontare l'oggetto con un modello idea, permettendo così un rapido controllo eventuali di danni o deformazioni.

Allo stato attuale sono rese disponibili solamente le funzionalità di ricostruzione dell'oggetto e calcolo (approssimato) del volume.
Alcune operazioni sarebbero troppo pesanti per le potenzialità del tablet, quindi è stato realizzato un *backend server* per tutte l'elaborazione delle ricostruzioni, mentre la realizzazione delle stesse è affidata all'applicativo per tablet.

1.3 Il Prodotto - lato dispositivo

L'applicazione prodotta risponde, in maniera minimale, alle esigenze citate nel punto precedente.
La sua realizzazione presenta molti punti critici e rischi piuttosto difficili da prevedere. Per questo sono stati realizzati molti prototipi, al fine di escludere vie non percorribili e trovare una soluzione soddisfacente.
Lo scopo principale della applicazione lato tablet è quello di rilevare ed elaborare un corretto *Point Cloud* dell'oggetto che si vuole ispezionare.
Un *Point Cloud* non è altro che una descrizione algebrica di un oggetto tridimensionale ottenuta tramite un insieme, il più possibile fitto, di punti che lo compongono. I dispositivi Tango infatti, grazie al sensore di profondità, cercano di rilevare le triplette di coordinate del maggior numero di punti possibile. Sfruttando questi dati è possibile posizionare dei punti nello spazio in maniera da fornire all'utente una rappresentazione comprensibile dell'oggetto.

1.3.1 Primo prototipo

Il primo prototipo realizzato risponde all'esigenza di catturare e salvare in formato leggibile da un *render* grafico i dati forniti dal sensore di profondità. Nella sua semplicità ha dato modo allo studente di testare la stabilità delle *API* e produrre della documentazione interna che riportava quali fossero i metodi delle *API* da utilizzare e quali fossero invece quelli poco stabili, sperimentali o addirittura non ancora implementati dal produttore.

1.3.2 Secondo prototipo: Cloude

Affrontare la discrepanza tra coordinate assolute e coordinate relative

Un solo *Point Cloud* non è sufficiente a ricostruire un oggetto. Ovviamente il dispositivo, registrando la nuvola di punti inquadrata in un determinato istante, riesce a rilevare solamente i punti che "riesce a vedere": i punti presenti nella parte posteriore dell'oggetto scansionato non possono essere "visti" e conseguentemente nemmeno misurati. Se si vuole avere una ricostruzione completa e non solamente di una facciata è necessario prendere più rilevazioni ed integrarle.

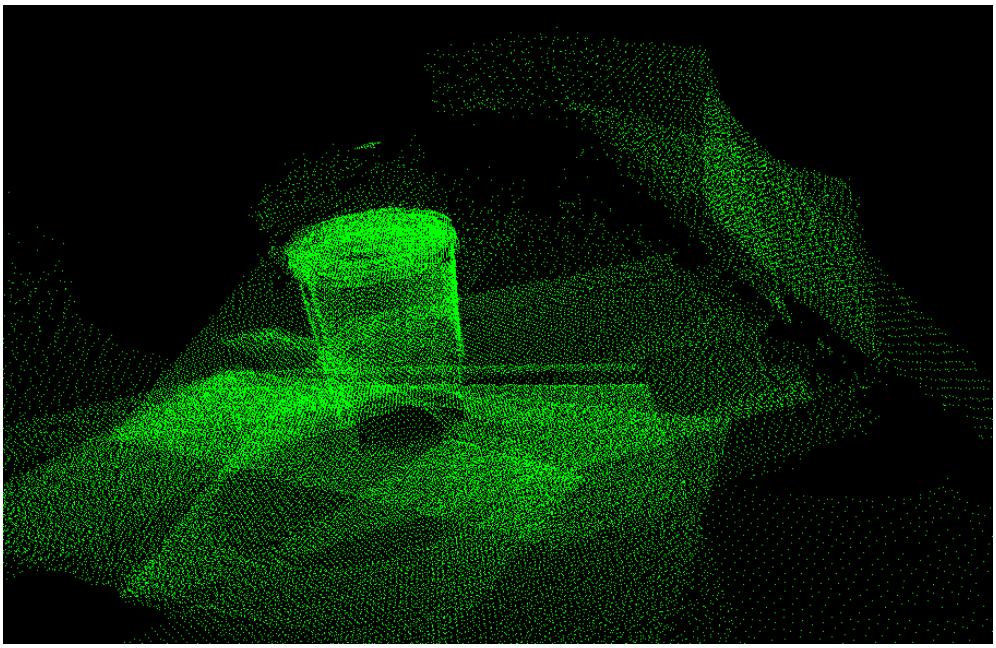


figura 1.1: Point Cloud di un bidone Conico

Le seguenti immagini mostrano il *Point Cloud* che descrive la parte anteriore di una scatola rettangolare; dato che la ripresa è stata effettuata da di fronte ed in alto solo le facce superiore ed anteriore sono state memorizzate, mentre delle altre non si hanno dati. I contorni sono stati evidenziati successivamente per permettere una migliore comprensione della forma.

Approccio

Questo prototipo, denominato *Cloude*, è stato realizzato allo scopo di rispondere a questa esigenza. L'idea che ne sta a fondamento è la seguente:

* Permettere all'utente di scattare alcune foto all'oggetto, quindi di rilevare diversi *Point Cloud*.
* Tenere costantemente traccia della posizione del dispositivo, in particolare delle posizioni nei momenti in cui vengono scattate le foto.
* Usare la posizione relativa al *Point Cloud* per traslare e ruotare lo stesso punto a punto, riducendo così le coordinate a dei valori assoluti.
* Ora le nuvole di punti registrate sono sovrapponibili le une con le altre e forniscono una prima ricostruzione dell'oggetto.

1.3.3 Terzo prototipo: Samba

Il prototipo precedente generava delle ricostruzioni riconoscibili, ma piuttosto imprecise. Una analisi dello stesso ha fatto emergere diverse criticità che sono state documentate, assieme alle possibili soluzioni, all'interno di un documento descrittivo. Quest'ultimo è stato alla base dello sviluppo di *Samba*.

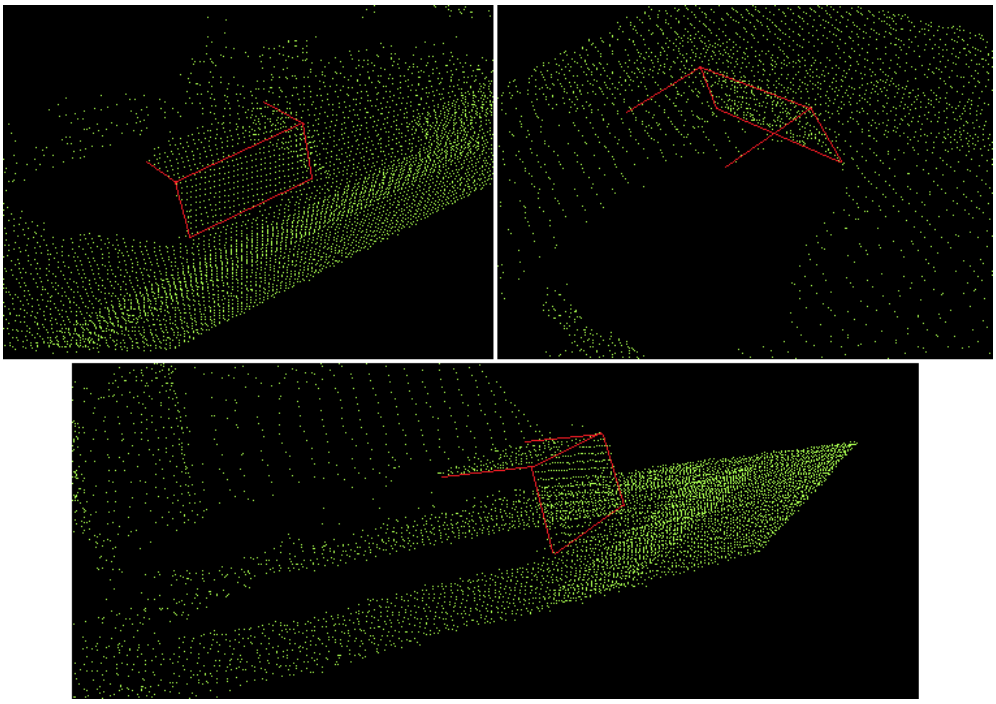


figura 1.2: Un singolo *Point Cloud*

Eccessiva complessità dell'elaborazione

Cloude sfrutta un metodo delle librerie *Tango* che trasforma le coordinate di un singolo punto in coordinate assolute fruttando la posizione relativa a cui si trovava il dispositivo, permette di scrivere poco codice, ma ha una elevata complessità. Ciò comporta un sensibile rallentamento dell'elaborazione dei *Point Cloud*. Un *cloud* medio conta intorno ai 90000 punti e con questo approccio richiede mediamente 1,5-2 secondi per essere completamente elaborato, tempo non accettabile per lo scopo per cui l'applicazione è pensata.
In *Samba* è stato cambiato radicalmente approccio:

* Ad ogni *Point Cloud* viene associata una matrice di trasformazione e non la posizione stessa.
* In questo modo è sufficiente moltiplicare ogni punto (vettore) per la matrice, che viene calcolata una sola volta per ogni *Point Cloud*.
* Si è ottenuta così una complessità di `O(n)` sul numero dei punti da trasformare riducendo i tempi di elaborazione da 1,5-2s a circa 200ms (sullo stesso dispositivo).

Bassa qualità delle ricostruzioni

Nelle ricostruzioni generate da *Cloude* gli oggetti appaiono deformati, spesso i vari *Point Cloud* non si sovrappongono correttamente generando fenomeni di *ghosting*, talvolta rendendo addirittura irriconoscibile l'oggetto.
Questo è dovuto ad una scorretta stima della posizione del dispositivo, che induce il calcolo di una erronea matrice di trasformazione, e quindi ad un errato posizionamento

delle nuvole di punti all'interno dello spazio.
Il fenomeno in questione è detto *"drifting"*: i *device Tango*, esattamente come le più comuni applicazioni in realtà aumentata, utilizzano la tecnica del *Motion Tracking* che consiste nel calcolare la propria posizione frequentemente ed in maniera relativa alla coordinate acquisite nella stima precedente. Per quando queste stime siano estremamente precise generano una catena di piccoli errori che sommati tra loro molto presto portano ad una importante discrepanza tra la posizione stimata dal dispositivo e quella reale. Ad esempio partendo da una determinata posizione e camminando in cerchio è praticamente impossibile che la traiettoria stimata passi nuovamente per il punto di partenza. Ciò è un limite fisico dei dispositivi, ed è nella pratica impossibile da

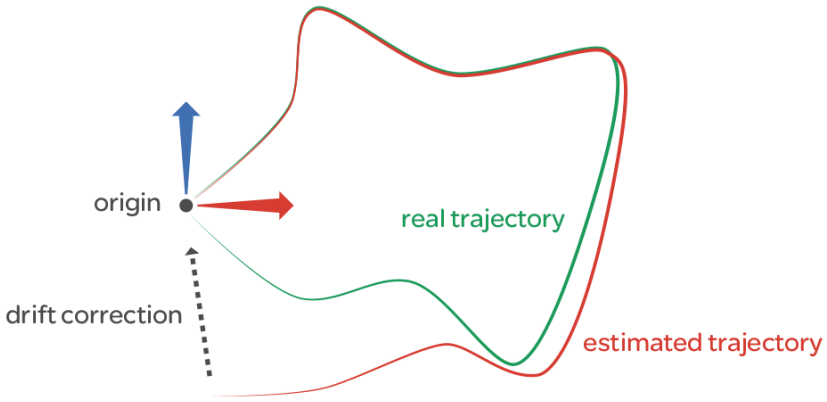


figura 1.3: Motion Tracking

eliminare, in quanto sarebbe necessario azzerare completamente gli errori relativi.
La tecnologia *Tango* però fornisce un altro meccanismo di localizzazione: l'*Area Learning*. Le applicazioni ideate per questo tipo di dispositivi infatti hanno la possibilità di mantenere memoria degli spazi che visitano, e successivamente usare queste informazioni per localizzarsi.
Il meccanismo è piuttosto simile a quello della memoria spaziale umana: una persona portata bendata all'interno di un edificio sconosciuto, una volta liberata, non avrà alcun mezzo per intuire dove si trovi; se invece la stessa persona fosse condotta all'interno della propria abitazione, alla prima sbirciata noterebbe immediatamente qualche particolare che gli farebbe immediatamente recuperare l'orientamento.
Allo stesso modo il tablet è in grado memorizzare alcune *features* all'interno dell'ambiente ed usarle come faro per la triangolazione.
Memorizzare completamente un ambiente tuttavia è una operazione che richiede parecchio tempo e costringe l'utente a muoversi per diversi minuti inquadrando tutti i dettagli del luogo dove si trova. Per rendere l'applicazione maggiormente responsiva e più vicina alle esigenze dell'utenza *Samba* adotta un approccio detto *Drift Correction*: inizialmente è richiesto all'utente di inquadrare per una ventina di secondi l'ambiente, in maniera da permettere la creazione di una minimale memorizzazione, successivamente il *Motion Tracking* è usato per piccoli spostamenti ma viene corretto nonappena venga inquadrata qualcuna delle (poche) *features* memorizzate. Trasparentemente all'utente, in background, il processo di *Area Learning* continua, memorizzando sempre nuovi dettagli e conseguentemente aumentando sempre più la qualità della registrazione.

Dimensioni eccessive dei file, ridondanza dei punti sovrapposti

Data la grande mole di punti registrati dai sensori di profondità i *file* contenenti le ricostruzioni generati da *Cloude* sono di dimensione eccessiva, anche più di 10Mb una decina scatti. Considerando che idealmente gli scatti da riprendere potrebbero essere molti e spesso dovranno essere inviati al *Server* tramite connessione a consumo il peso di questi *file* non è da trascurare.
Inoltre c'è una grossa ridondanza di punti: è comune caso d'uso che una stessa zona venga inquadrata in più scatti, quindi tali *Point Cloud* ruotati ed uniti presenterebbero molti punti con le stesse coordinate e semplicemente sovrapposti, quindi senza dare alcuna informazione aggiuntiva.
Samba risolve questo problema utilizzando un leggero *voxeling*, ovvero suddividendo lo spazio in cubi o *voxel* di lato prefissato e registrando quali sono i *voxel* che contengono i punti della nuvola. Scegliendo una opportuna definizione, ovvero una opportuna dimensione dei *voxel*, si può ottenere una ricostruzione comunque con un buon livello di dettaglio, ma priva di ridondanza dei punti e quindi meno pesante in termini di memoria.
Dopo diversi test è emerso che definizioni più fine di un millimetro non portano a nessun effettivo miglioramento della qualità della ripresa, e quindi questo è stato scelto come definizione predefinita.

1.3.4 Ulteriore incremento di Samba, il prototipo finale

Samba, nella sua prima realizzazione, soddisfaceva completamente gli obiettivi fissati per quanto riguarda la *business logic* tuttavia è risultato carente nell'interfacciarsi con l'utente.
Quindi è stato pianificato un ulteriore ciclo di raffinamento che ha portato questo prototipo allo stato attuale. Segue una breve lista delle principali migliorie.

Preview dell'inquadratura

La sola preview della fotocamera a colori si è rivelata non sufficiente. Oltre a non fornire una chiara idea dei punti che verranno registrati non da la possibilità all'utente di controllare se la foto che sta per scattare sia "buona" oppure no.
Il sensore di profondità sfrutta la tecnologia *infrared* e per questo è soggetto a tutti i limiti fisici di quest'ultima; sono emerse, infatti, grosse difficoltà nel misurare i punti

* di una superficie molto scura.
* di una superficie riflettente o particolarmente lucida.
* all'interno di stanze con illuminazione scarsa o assente.

A volte il problema è insormontabile e la ricostruzione non potrà avvenire con successo, altre è sufficiente trovare una buona posizione per permettere al sensore di effettuare le misurazioni. Questo rende necessario mostrare sullo schermo non solo quello che "vede" la fotocamera, ma anche quello che "vede" il dispositivo *infrared*.
A questo scopo è stata parzialmente riusata una applicazione di prova fornita sotto licenza *Open Source* dalla *Google*[2] e che utilizza la libreria grafica *Rajawali*[3].

[2] **GitHub: https://github.com/googlesamples/tango-examples-java**.
[3] **GitHub: https://github.com/Rajawali/Rajawali**.

Il *render* in questione è moto semplice ma efficace ed aumenta il valore aggiunto dell'applicazione dandole un aspetto gradevole e permettendo all'utente di avere aggiornamenti in tempo reale sul *Point Cloud* inquadrato.

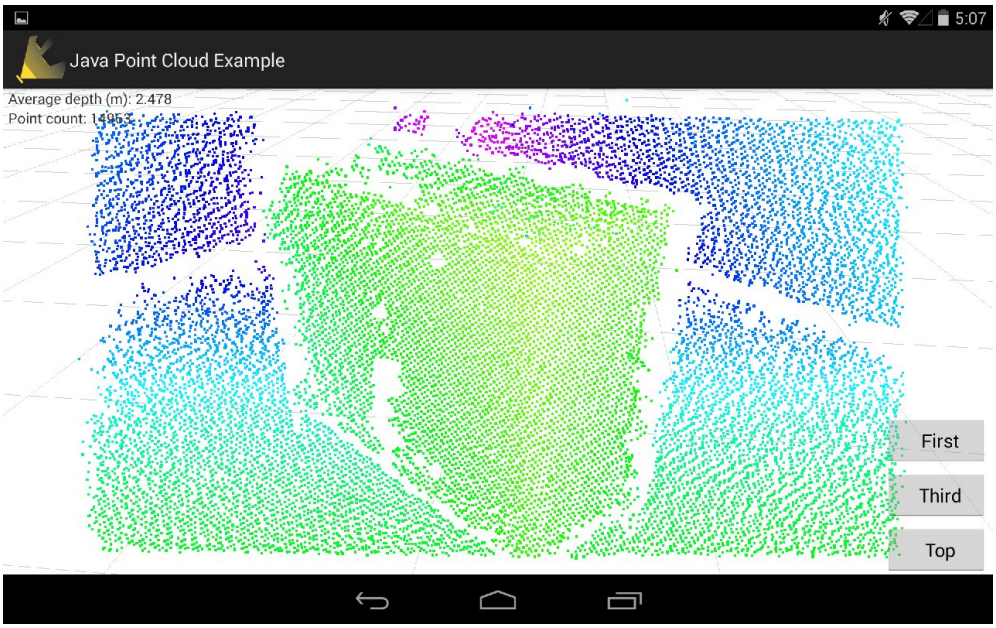


figura 1.4: Il render di Java Point Cloud Example mentre viene inquadrata una credenza

Visualizzazione su tablet del *Point Cloud* ricostruito

Con il *render* descritto nel punto precedente è possibile visualizzare ciò che "vede" il sensore di profondità in tempo reale. Questo tuttavia non è sufficiente: mano a mano che si effettua la ripresa, in background viene calcolata la ricostruzione dell'oggetto in analisi, non poterlo visualizzare sul *tablet*, ma solo lato *server* appare quantomeno frustrante e soprattutto non da la possibilità all'utente di avere una idea dello stato in cui è la ricostruzione.
Per questo si è pensato di sfruttare lo stesso meccanismo di *rendering* per far visualizzare all'utente la ricostruzione che sta effettuando. É quindi stata aggiunta la possibilità di alternare tra la visualizzazione in tempo reale e la visualizzazione dell'intera ricostruzione generata.

Aggiunta operazioni di undo

Nonostante gli sforzi per mantenere una alta qualità delle riprese qualcuno dei *Point Cloud* catturati continua a presentare grossi difetti (come errato posizionamento, deformità, grossa presenza di rumore etc). Importante miglioria effettuata in questa fase è stata l'inserimento della possibilità di annullare un certo numero di operazioni; in questo modo le riprese che contengono dei difetti possono essere scartate e ripetute.

Istruzioni per l'utente

Questa applicazione, come tutte le applicazioni *Tango*, introduce nuove azioni che l'utente deve compiere per mettere il dispositivo nella condizione di poter operare al meglio; ed è compito degli applicativi istruire l'utente sul comportamento da tenere.
Ad esempio, durante l'avvio dell'*app*, l'utente deve avere cura di mantenere il *tablet* in posizione verticale ed il più possibile fermo. Questa, come la maggior parte delle indicazioni, possono essere notificate all'utente tramite un *framework* messo a disposizione dal produttore in grado di integrare segnali e notifiche all'interno del ciclo di vita dell'applicazione stessa.
Fanno eccezione le istruzioni che devono essere date all'utente durante la fase di localizzazione. Le *API Tango* non forniscono alcun aiuto per stabilire se il dispositivo si sia orientato o meno. È stato quindi necessario studiare delle euristiche capaci di intuire se il dispositivo "si senta o meno a proprio agio".
L'idea sta nello sfruttare il fatto che i dispositivi *Tango*, una volta imparato un ambiente, fissano l'origine O del loro sistema di riferimento (per quel particolare ambiente) in un punto arbitrario dello stesso. Alla luce di questo è sufficiente richiedere, durante la fase di localizzazione, la posizione della suddetta origine rispetto ad un punto speciale detto *start-of-service point* che chiameremo S, ovvero il punto in cui era il dispositivo al momento dell'avvio dell'applicazione.

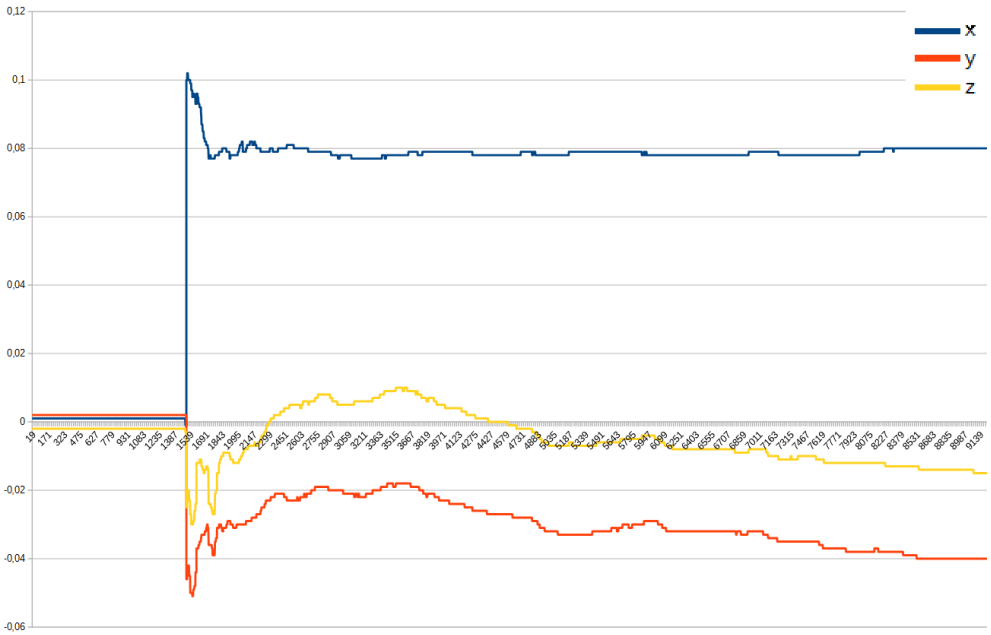


figura 1.5: Grafico della coordinate di S rispetto ad O durante la fase di inizializzazione della *Drift Correction*

Il grafico in figura rappresenta le coordinate x, y e z della distanza con segno tra O ed S nel primo minuto circa di attività dell'applicazione. Esso può essere diviso in tre fasi:

* Inizialmente la distanza molto è vicina a zero. Prima che avvenga la localizzazione il sistema *Tango* non ha altra scelta che posizionare O nelle stesse coordinate di

S.

* Successivamente avverrà un brusco cambiamento nella distanza tra questi due punti perché tutto d'un tratto il sistema riconoscerà qualche *feature* e sposterà immediatamente l'origine degli assi. Questa fase non è ancora stabile in quanto l'ambiente è ancora in fase di riconoscimento e l'origine O verrà traslata spesso.
* Una volta riconosciuto correttamente l'ambiente (nella seconda metà del grafico) si può notare che tutte le coordinate si stabilizzano ad una distanza quasi fissa. Da questo momento in avanti si avrà una buona localizzazione e l'utente potrà iniziare la rilevazione.

La distanza tra O ed S dovrebbe essere idealmente costante. Si è notato sperimentalmente, però, che non è così: a causa di limiti fisici del dispositivo essa continuerà a fluttuare entro un paio di centimetri di raggio. Ciò a volte può creare qualche piccolo errore nella ricostruzione, ma per oggetti piuttosto grandi esso è trascurabile.

1.4 Il Prodotto - lato server

L'applicazione lato server si occupa di "pulire" la ricostruzione dagli elementi inutili, come ad esempio pavimento, rumore, oggetti di sfondo e convertirla in un formato portabile.

1.4.1 Point Cloud Library

PCL o *Point Cloud Library* è una libreria *Open Source* e *Large Scale* per l'elaborazione di 2D e 3D di immagini e *Point Cloud*. Fornisce diversi filtri ed algoritmi in grado di risolvere molti dei problemi che sono stati riscontrati per quando riguarda l'elaborazione delle ricostruzioni.
L'applicativo lato server fa vasto uso di questa libreria.

1.4.2 Generazione mesh

Per poter elaborare facilmente la ricostruzione ed ottenere un modello portabile è necessario trasformare la nuvola di punti in una *mesh*, ovvero una rappresentazione di un oggetto 3D che consiste in un insieme di facce poligonali, solitamente semplici triangoli. Inoltre tutti i più diffusi formati per oggetti tridimensionali come `obj` e `ply` sono in grado di rappresentare solamente *mesh*.
Per ottenere una buona riproduzione dell'oggetto l'applicativo applica diversi filtri:

* **sparse filter / filter radius**: vengono eliminati i punti isolati ed i punti ad una eccessiva distanza dal centro, i quali sono quasi sempre frutto di errori nelle misurazioni.
* **filter ground**: viene eliminato il pavimento.
* **voxel filter**: viene effettuata una ulteriore operazione di voxeling allo scopo di ridurre la mole di calcoli e regolarizzare il *Point Cloud*.
* **cluster exstractor**: si cerca di suddividere la ricostruzione nei vari oggetti da cui è composta, dopodiché si mantiene solamente l'oggetto (o *cluster*) che si trova più al centro.

* **meshing**: i punti rimanenti sono solamente quelli dell'oggetto in analisi. Essi vengono usati per generare la *mesh*.

Al termine di questo processo si ottiene l'oggetto 3D, che può essere convertito nel formato preferito. L'immagine sottostante mostra il processo di elaborazione dell'immagine di una scatola rettangolare. Si possono osservare, da in alto a sinistra ad in basso a destra: la ricostruzione fornita dal *tablet*, il *Point Cloud* risultante da *sparseFilter* e *filterRadius*, l'output della operazione di rimozione del pavimento, il risultato del *voxelFilter*, il *cluster* estratto ed infine la *mesh* dell'oggetto.

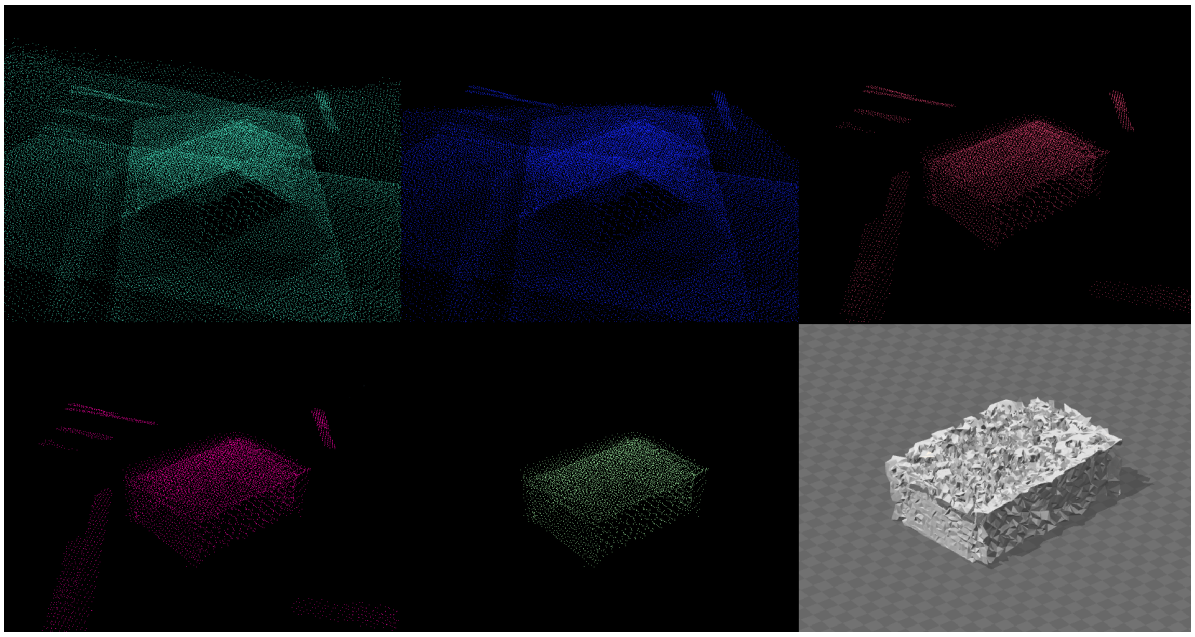


figura 1.6: Input ed Output del processo di meshing

1.4.3 Calcolo del volume

Una volta ottenuta la *mesh* il calcolo del volume è piuttosto immediato. A questo fine è stato sfruttato il risultato di una pubblicazione di *Cha Zhang* e *Tsuhan Chen* dal titolo *EFFICIENT FEATURE EXTRACTION FOR 2D/3D OBJECTS IN MESH REPRESENTATION*[4]. Il trucco è calcolare, per ogni triangolo che compone la *mesh*, il volume con segno del tetraedro che ha il triangolo stesso come base e il quarto vertice in un punto fissato, scelto internamente alla *mesh*, per evitare eventuali problemi di instabilità numerica. Il segno del volume è dato dalla direzione della normale al piano del triangolo. Questi volumi, sommati tra loro, restituiscono il volume convesso della *mesh*.

1.5 Organizzazione del testo

xxxx

[4] **site: http://research.microsoft.com/en-us/um/people/chazhang/publications/icip01˙ChaZhang.pdf**.

Il secondo capitolo descrive ...

Il terzo capitolo approfondisce ...

Il quarto capitolo approfondisce ...

Il quinto capitolo approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Processi e metodologie

xxxx Brevissima introduzione al capitolo

2.1 Processo sviluppo prodotto

xxxx

Capitolo 3

Studio di fattibilità ed analisi dei rischi

Iniziare, quasi da zero, un progetto così impegnativo basandosi su di una tecnologia al limite dell'essere sperimentale espone a gravi rischi di fallimento. Per ciò in questa fase sono state investite molte risorse.

3.1 Introduzione al progetto

Data la natura innovativa del progetto è stato necessario produrre diversi prototipi ed effettuare l'*Analisi dei Rischi* e lo *Studio di Fattibilità* in diverse fasi.
Questo approccio è stato estremamente utile per far emergere rischi dovuti sia alla non piena maturità delle *API*, sia ai limiti fisici del dispositivo in dotazione.

3.2 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

3.2.1 Rischi generali

1. Immaturità di API/librerie/documentazione
Probabilità: Alta.
Gravità: Media.
Descrizione: Le tecnologie adottate sono innovative e tuttora in fase di sviluppo, molte sono ancora segnalate come *"Sperimentali e soggette a cambiamenti"*. Per questo le librerie usate potrebbero rivelarsi instabili o potrebbero mancare di adeguata documentazione.
Contromisure: Iscrizione ai vari canali di segnalazione e supporto offerti da *Google* per gli sviluppatori, sviluppo di piccoli esempi giocattolo per testare le funzionalità offerte dalle *API* da cui è stata generata della documentazione interna.

2. Limiti fisici del dispositivo
Probabilità: Alta.
Gravità: Media.
Descrizione: Il dispositivo è dotato di sensori infrarossi e sfrutta la riflessione della luce per determinare la distanza dei punti che è in grado di individuare. Superfici riflettenti o molto scure possono compromettere la qualità della misurazione, allo stesso modo situazioni di illuminazione scarsa o assente.
Contromisure: Accurata analisi della documentazione fornita dal produttore[1], test preventivi nelle situazioni critiche utilizzando una semplice applicazione di prova fornita da *Google*[2].

3.2.2 Rischi specifici

3. Difficoltà nel Motion Tracking
Probabilità: Media.
Gravità: Alta.
Descrizione: Determinare la posizione e l'orientamento del dispositivo in maniera assoluta è fondamentale per permettere la ricostruzione dell'oggetto inquadrato. Il *device* fornisce ad intervalli regolari la sua posizione tramite una tripletta di coordinate ed la sua rotazione rappresentata come un quaternione. La somma di piccoli errori relativi nella stima della posizione crea un fenomeno detto *drifting* che comporta imporanti errori nella stima finale. Questo rischio può portare al fallimento del progetto, in quanto se non opportunamente mitigato renderebbe le ricostruzioni tridimensionali totalmente errate..
Contromisure: Si è per questo deciso di adottare una tecnica denominata *Area Learning*. Il disposivo quindi riconoscerà alcune *features*, ovvero dei punti fissi, rispetto ai quali determinerà la sua posizione..

4. Necessità di azioni specifiche da parte dell'utente
Probabilità: Alta.
Gravità: Alta.
Descrizione: Tutte le applicazioni che usano la tecnologia *Tango* iteragiscono strettamente con i movimenti e la posizione dell'utente. La scarsa diffusione di questa tecnologia fa si che la maggior parte dell'utenza non sia a conoscenza del comportamento che deve tenere. Azioni compiute dall'utente in maniera scorretta possono compromettere il buon funzionamento dell'applicazione..
Contromisure: Tutto lo sviluppo dell'applicazione deve tenere conto di questo fatto. Devono essere fornite chiare informazioni all'utente e si devono studiare soluzioni che non costringano l'*user* ad un comportamento troppo antiintuitivo..

5.
Probabilità: .
Gravità: .
Descrizione: .
Contromisure: .

[1] **site: https://developers.google.com/tango/overview/depth-perception**.
[2] **site: https://github.com/googlesamples/tango-examples-java/tree/master/java˙point˙cloud˙example**.

3.3 Studio di fattibilità

3.3.1 Preventivo

Prima di iniziare il progetto è stato effettuato un accurato studio di fattibilità basato sulla ricerca di progetti con funzionalità simili e sulla lettura delle numerose discussioni e pubblicazioni presenti nel *web* a riguardo di *Project Tango*. xxxx

Capitolo 4

Analisi dei requisiti

Breve introduzione al capitolo

4.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo Unified Modeling Language (UML) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Essendo il progetto finalizzato alla creazione di un tool per l'automazione di un processo, le interazioni da parte dell'utilizzatore devono essere ovviamente ridotte allo stretto necessario. Per questo motivo i diagrammi d'uso risultano semplici e in numero ridotto.

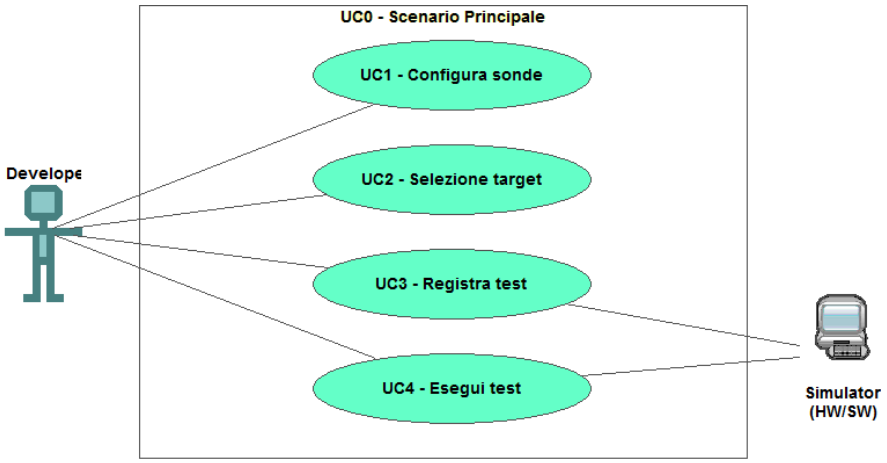


figura 4.1: Use Case - UC0: Scenario principale

UC0: Scenario principale

Attori Principali: Sviluppatore applicativi.

Precondizioni: Lo sviluppatore è entrato nel plug-in di simulazione all'interno dell'IDE.

Descrizione: La finestra di simulazione mette a disposizione i comandi per configurare, registrare o eseguire un test.

Postcondizioni: Il sistema è pronto per permettere una nuova interazione.

4.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Il codice dei requisiti è così strutturato R(F/Q/V)(N/D/O) dove:

R = requisito

F = funzionale

Q = qualitativo

V = di vincolo

N = obbligatorio (necessario)

D = desiderabile

Z = opzionale

Nelle tabelle 4.1, 4.2 e 4.3 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

tabella 4.1: Tabella del tracciamento dei requisti funzionali

Requisito	Descrizione	Use Case
RFN-1	L'interfaccia permette di configurare il tipo di sonde del test	UC1

tabella 4.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Use Case
RQD-1	Le prestazioni del simulatore hardware deve garantire la giusta esecuzione dei test e non la generazione di falsi negativi	-

tabella 4.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Use Case
RVO-1	La libreria per l'esecuzione dei test automatici deve essere riutilizzabile	-

Capitolo 5

Progettazione e codifica

Breve introduzione al capitolo

5.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati.

Tecnologia 1

Descrizione Tecnologia 1.

Tecnologia 2

Descrizione Tecnologia 2

5.2 Ciclo di vita del software

5.3 Progettazione

Namespace 1

Descrizione namespace 1.

Classe 1: Descrizione classe 1

Classe 2: Descrizione classe 2

5.4 Design Pattern utilizzati

5.5 Codifica

Capitolo 6

Verifica e validazione

Capitolo 7

Conclusioni

7.1 Sviluppi futuri

7.1.1 icp on tablet

7.1.2 texture dei punti

7.1.3 ingrazione cpp lato tablet

7.2 Consuntivo finale

7.3 Raggiungimento degli obiettivi

7.4 Conoscenze acquisite

7.5 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Bibliografia

Riferimenti bibliografici

Siti Web consultati