

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA

CORSO DI LAUREA IN INFORMATICA



Project Tango e PCL: un'applicazione
pratica alle 'Nuvole di Punti'

Tesi di laurea triennale

Relatore

Prof. Ombretta Gaggi

Laureando

Davide Bortot

ANNO ACCADEMICO 2015-2016

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

— Oscar Wilde

Dedicato a ...

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecentoventi ore, dal laureando Davide Bortot presso l'azienda VIC S.r.l. Obiettivo del tirocinio era lo sviluppo di un'applicazione prototipale per il tablet Project Tango per l'acquisizione di scansioni di oggetti reali sotto forma di 'Nuvola di Punti'.

In primo luogo era richiesto il filtraggio e l'elaborazione dei dati acquisiti in modo da isolare l'oggetto scansionato, e successivamente di creare una mesh tridimensionale a partire dai punti rimanenti. Scopo finale era calcolare il volume dell'oggetto tridimensionale così ottenuto.

“Life is really simple, but we insist on making it complicated”

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine alla Prof.ssa Ombretta Gaggi, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

In secondo luogo, ringrazio Michele Marchetto, tutor aziendale, così come tutti i colleghi Tommaso Padovan, Lorenzo Ceccon e Luca D'Ambros per aver reso lo stage un'esperienza di crescita.

Ringrazio col cuore i miei genitori Giulia e Carlo per aver creduto in me in ogni momento e avermi sempre sostenuto.

Infine ringrazio i miei amici di sempre e gli amici nuovi che ho incontrato nel mio percorso universitario, per avermi reso quello che sono.

Padova, Oct 2016

Davide Bortot

Indice

1	Introduzione	1
1.1	L'azienda	1
1.2	L'idea	1
1.3	Cos'è Project Tango	2
1.4	Il Prodotto - lato client	3
1.4.1	Primo prototipo: Cloude	3
1.4.2	Secondo prototipo: Samba	4
1.4.3	L'applicativo attuale: VIC-Tango	6
1.5	Il Prodotto - lato server	10
1.5.1	Point Cloud Library	10
1.5.2	Elaborazione di un Point Cloud	10
1.6	Calcolo del volume	13
1.7	Organizzazione del testo	13
2	Processi, tecnologie e strumenti	15
2.1	Processo sviluppo prodotto	15
2.1.1	Kick-Off	15
2.1.2	Concept Preview	15
2.1.3	Product Prototype	15
2.1.4	Fasi successive	16
2.2	Tecnologie e Strumenti	16
2.2.1	Codice	16
2.2.2	IDE ed editor	17
2.2.3	Framework	17
3	Studio di fattibilità ed analisi dei rischi	19
3.1	Introduzione al progetto	19
3.2	Analisi preventiva dei rischi	19
3.3	Requisiti e obiettivi	19
3.4	Pianificazione	19
4	Analisi dei requisiti	21
4.1	Casi d'uso	21
4.2	Tracciamento dei requisiti	22
5	Progettazione e codifica	25
5.1	Tecnologie e strumenti	25
5.2	Ciclo di vita del software	25
5.3	Progettazione	25

5.4	Design Pattern utilizzati	25
5.5	Codifica	25
6	Verifica e validazione	27
7	Conclusioni	29
7.1	Consuntivo finale	29
7.2	Raggiungimento degli obiettivi	29
7.3	Conoscenze acquisite	29
7.4	Valutazione personale	29
A	Appendice A	31
	Bibliografia	35

Elenco delle figure

1.1	L' hardware di <i>Project Tango</i>	2
1.2	Esempio di <i>Point Cloud</i> : un cestino cilindrico	3
1.3	<i>Drifting</i> nel <i>Motion Tracking</i>	4
1.4	Benefici della <i>Drift Correction</i>	5
1.5	Effetti del <i>voxeling</i>	6
1.6	<i>Point Cloud Registration</i> tramite l'algoritmo <i>ICP</i>	8
1.7	Errore nella <i>Registration</i> con <i>GOICP</i>	9
1.8	<i>Sparse filtering</i>	11
1.9	<i>Radius filtering</i>	12
1.10	<i>Ground filtering</i>	12
4.1	Use Case - UC0: Scenario principale	21

Elenco delle tabelle

4.1	Tabella del tracciamento dei requisiti funzionali	23
4.2	Tabella del tracciamento dei requisiti qualitativi	23
4.3	Tabella del tracciamento dei requisiti di vincolo	23

Capitolo 1

Introduzione

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 alle nuove tecnologie. Si occupa di controlli per grandi ordini, sia di materie prime che di semi-lavorati, di cui individua e riporta eventuali danni, carenze nella spedizione e non conformità con quanto ordinato. VIC viene fondata a Venezia 7 anni fa come piccola società di ispezione locale. Fin dall'inizio, l'obiettivo principale di VIC è stato la riduzione del tempo tra ispezione e reporting al cliente. Ora l'obiettivo è raggiunto, perchè 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 mobili.

1.2 L'idea

I più importanti obbiettivi del controllo qualità effettuato durante un ispezione sono il determinare la corretta forma, peso, quantità e dimensioni degli oggetti da esaminare. Gli ispettori possono scattare 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.

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 dati utili quali:

- * Una stima del volume, e se necessario del peso, dell'oggetto.
- * L'esito del confronto dell'oggetto con un modello ideale, per evidenziare eventuali danni o deformazioni.

Con il prototipo realizzato durante lo *stage* sono rese disponibili solamente le funzionalità di ricostruzione dell'oggetto e calcolo (approssimato) del volume.

Le operazioni troppo computazionalmente intensive da effettuare su tablet, quali il filtraggio e *meshing* dei punti acquisiti, sono state delegate ad un *backend server* che effettui le elaborazioni necessarie ed invii i risultati al richiedente, mentre all'applicativo

per tablet è stato affidato il compito di acquisire e visualizzare un oggetto sotto forma di 'Nuvola di Punti' e poterne esaminare la *mesh* ottenuta.

1.3 Cos'è Project Tango

Project Tango è un *tablet* sperimentale prodotto da *Google* in grado di scandire tridimensionalmente l'ambiente circostante e di tracciare la propria posizione rispetto ad esso. Ciò è possibile attraverso il ricco hardware di cui è dotato (si veda la figura 1.1), tra cui:

- * una fotocamera RGB-IR
- * una fotocamera *Fisheye*
- * un sensore di profondità IR
- * accelerometro e giroscopio

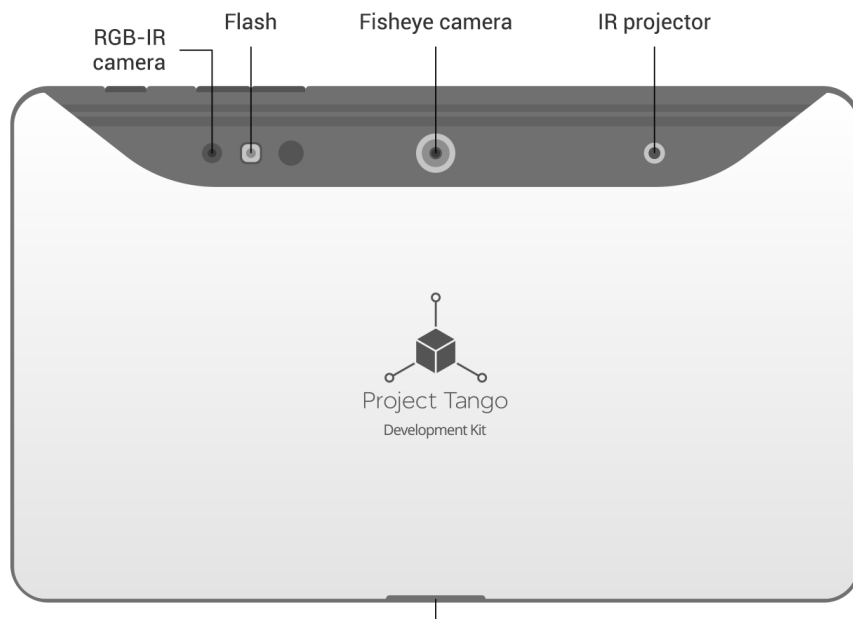


figura 1.1: L' hardware di *Project Tango*

Project Tango è quindi specificamente pensato per sviluppare applicazioni che necessitano di comprendere ed estrapolare informazioni dal mondo reale (ad es. realtà aumentata). Le funzionalità del *tablet* sono accessibili attraverso le *Tango API*, le *API* ufficiali per lo sviluppo di applicazioni *Tango*.

1.4 Il Prodotto - lato client

L'applicazione su *tablet* prodotta realizza, seppur non in maniera completa, le esigenze citate al punto precedente.

La sua realizzazione ha incontrato molte problematiche talvolta critiche e difficili da prevedere. Per questo durante lo sviluppo sono stati implementati più prototipi, al fine di esplorare le potenzialità e soprattutto i limiti del *tablet* e delle *Tango API*.

Lo scopo principale dell'applicazione lato *tablet* è quello di rilevare una corretta 'Nuvola di Punti' dell'oggetto che si vuole esaminare.

Una 'Nuvola di Punti' è una descrizione matematica di un oggetto tridimensionale ottenuta tramite un insieme, il più possibile fitto, di punti che lo compongono, definiti dalle loro coordinate (x,y,z) rispetto ad un fissato sistema di riferimento.

Tale rappresentazione, riferita spesso d'ora in poi con il più elegante termine inglese *Point Cloud*, è facilmente comprensibile all'utente se visualizzata come in figura 1.2.

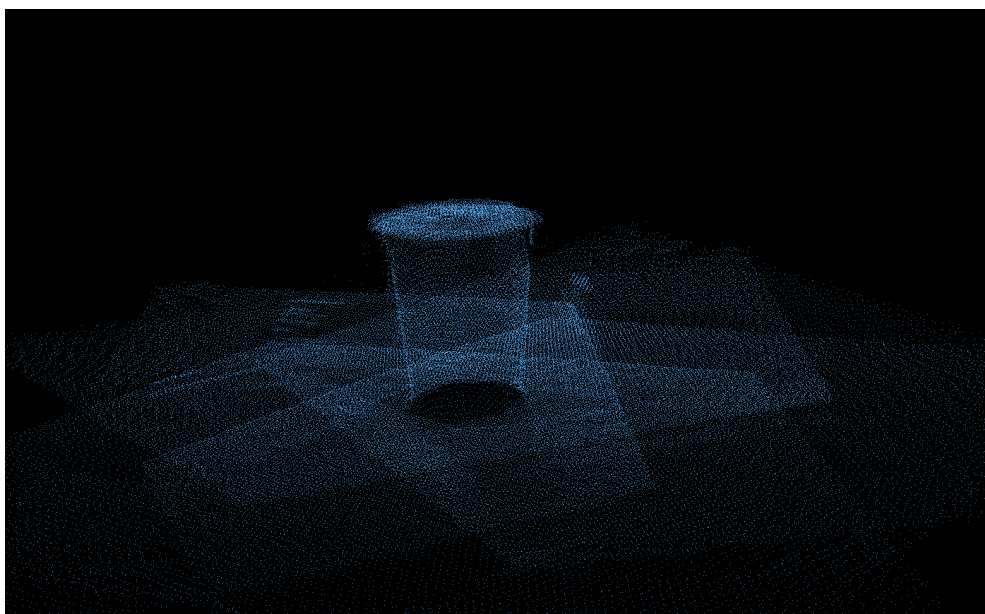


figura 1.2: Esempio di *Point Cloud*: un cestino cilindrico

1.4.1 Primo prototipo: Cloude

Il prototipo denominato *Cloude* risponde all'esigenza di catturare più *Point Cloud* da più angolazioni in modo da ricostruire l'oggetto scansionato sovrapponendo i dati acquisiti. *Cloude* è il risultato di più prototipazioni precedenti effettuate dallo stagista che ha iniziato il progetto prima di me e con cui ho strettamente collaborato.

Il prototipo permette quindi di acquisire più *Point Cloud* in successione, ognuno dei quali verrà adeguatamente ruotato e traslato rispetto allo spostamento del *tablet*, in modo che la sovrapposizione degli stessi vada infine a ricostruire tutte le parti dell'oggetto esaminato.

Una sola acquisizione da una certa angolazione non può infatti comprendere un oggetto nella sua interezza, come dimostrato nelle figure //TODO è necessario ricostruire passo per passo un unico *Point Cloud* rappresentante l'oggetto completo. Sulla base

dei risultati di *Cloude* ho iniziato il mio lavoro sulla parte di elaborazione lato *server* dei *Point Cloud* acquisiti; questo è stato inoltre il punto di inizio per lo sviluppo dei prototipi successivi in collaborazione con l'altro stagista.

1.4.2 Secondo prototipo: Samba

Il secondo prototipo parte dai risultati ottenuti con *Cloude* e ne migliora prestazioni e precisione, aggiungendo alcune funzionalità utili.

Samba affronta e risolve alcuni problemi del prototipo precedente, come il "*Drifting*" e alcuni problemi prestazionali dovuta alla mole elevata di dati acquisiti. Introduce inoltre la funzionalità di visualizzazione delle *mesh* elaborate dal *server*.

Drift correction

Il prototipo precedente, seppur funzionale, generava ricostruzioni di scarsa qualità, afflitte soprattutto dal problema del "*drifting*". Il *drifting* è un problema comune nelle applicazioni di realtà aumentata, che come *Project Tango* usano la tecnica del *Motion Tracking*, cioè aggiornano costantemente la propria posizione relativamente alle coordinate acquisite nella posizione precedente, mantenendo così una storia dei movimenti del *device* rispetto all'ambiente circostante.

Ad ogni aggiornamento della posizione è normale ed inevitabile che la misurazione, per quanto precisa, introduca un piccolo errore; la catena di errori sommati porta quindi col passare del tempo ad un'importante discrepanza tra la posizione stimata del *device* e la sua posizione reale, come evidenziato in figura 1.3 Per ovviare in parte a questo

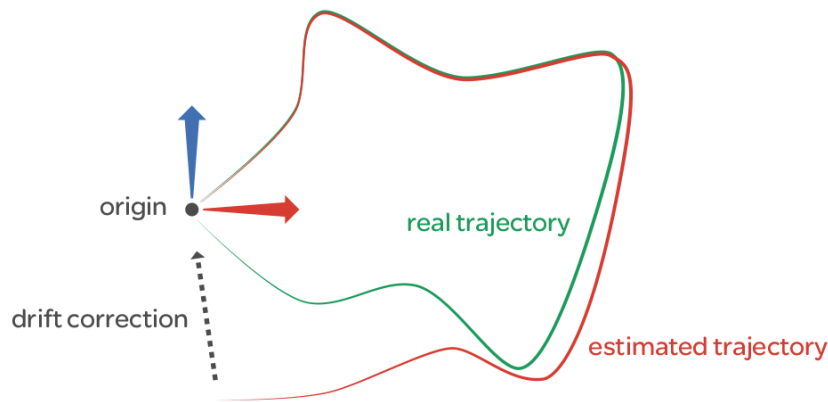


figura 1.3: *Drifting* nel *Motion Tracking*

problema *Samba* utilizza una tecnica chiamata *Drift Correction* che si appoggia sulle funzionalità di *Area Learning* del *device Tango*.

L' *Area Learning* consiste nella capacità del dispositivo di estrarre dallo spazio fisico che sta analizzando una serie di punti significativi (o *key features*), facilmente riconoscibili, e di salvare tali informazioni per confrontarle con le successive acquisizioni. In questo modo il dispositivo è capace di riconoscere un'area precedentemente visitata, e può quindi applicare le necessarie correzioni alla propria stima della traiettoria, di qui il nome *Drift Correction*.

Con l'implementazione di questa funzionalità è necessario, prima di iniziare la ricostruzione di un *Point Cloud*, riprendere l'oggetto e i suoi dintorni per un po' di tempo e da diverse angolazioni, in modo da permettere al *tablet Tango* di costruire una mappa, detta *Area description*, dell'area circostante, e di stabilizzare la traiettoria stimata con le informazioni acquisite.

I risultati si vedono confrontando queste due ricostruzioni di una scatola, vista dall'alto (fig. 1.4a e fig. 1.4b), rispettivamente senza e con *Drift correction*.

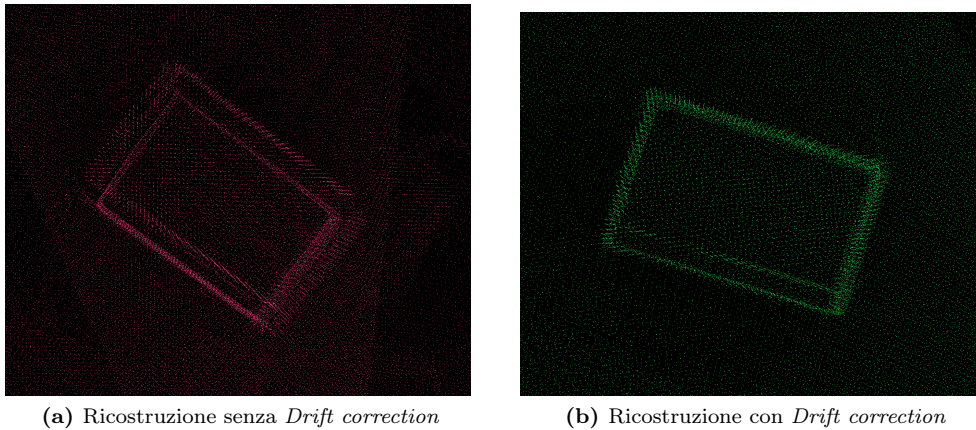


figura 1.4: Benefici della *Drift Correction*

Come si nota in figura 1.4b il risultato non è ancora ottimale, infatti un lato della scatola appare sdoppiato e spostato di qualche centimetro; si tratta di un problema di *ghosting* di cui verrà trattato più avanti.

Prestazioni

Il prototipo precedente presentava due problemi prestazionali principali:

- * La ricostruzione passo per passo del *Point Cloud* finale era troppo lenta
- * La mole dei dati trattati al sovrapporsi di più *Point Cloud* diventava proibitiva

Il primo problema si presentava utilizzando i metodi forniti dalla libreria *Tango* per trasformare le coordinate relative dei punti in coordinate assolute per permetterne la giusta sovrapposizione. Tale metodo, seppur di facile utilizzo, diventava troppo dispendioso all'aumentare delle dimensioni del *Point Cloud*, che raggiunge facilmente gli 80.000 punti.

Per risolvere il problema viene quindi creata, per ogni *Point Cloud*, una *matrice di rototraslazione* che rappresenta lo spostamento e la rotazione del dispositivo rispetto al sistema di riferimento, e viene moltiplicato il vettore delle coordinate di ogni singolo punto per la matrice generata.

In questo modo si sono ridotti i tempi di elaborazione dell'80%;

Il secondo problema era causato dal sovrapporsi di molti punti quasi identici, ad esempio i punti rappresentanti il pavimento. Partendo dall'osservazione che una certa quantità di punti molto vicini può essere trasformata in un unico punto, valore medio di tutti gli altri, senza una significativa perdita d'informazione, *Samba* risolve il problema

attraverso una tecnica di *voxeling*.

Lo spazio viene suddiviso in tanti piccoli parallelepipedi (tipicamente cubi) di uguali dimensioni; tutti i punti che ricadono all'interno di un singolo parallelepipedo, o *voxel*, vengono considerati come un unico punto. Così facendo si riducono sensibilmente le dimensioni del *Point Cloud* senza alterarne negativamente la precisione. Di seguito la differenza visivamente evidente tra lo stesso *Point Cloud* filtrato prima con voxel cubici di lato 1cm (fig. 1.5a) e poi di lato 3cm (fig. 1.5b), con una differenza di circa 60.000 punti rimossi in più nella seconda elaborazione.

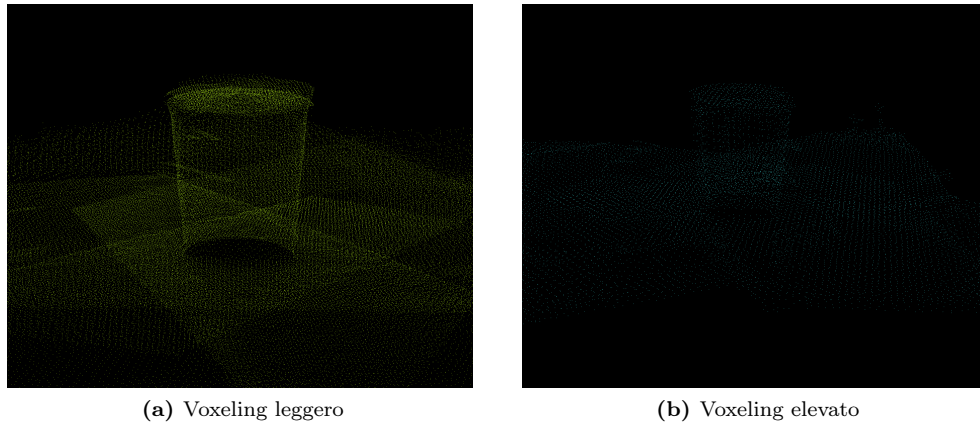


figura 1.5: Effetti del *voxeling*

Visualizzatore di mesh

Nel prototipo è stata introdotta la possibilità di caricare e visualizzare le *mesh* risultato dell'elaborazione lato server di un *Point Cloud*. Una *mesh* poligonale è una collezione di vertici, spigoli e facce che definiscono la forma di un oggetto poliedrico; in *Samba* è ottenuta a partire dalla nuvola di punti elaborata. Tale *mesh* viene salvata dal *server* in formato *OBJ*, comunemente usato nelle applicazioni 3D. Dall'applicazione viene quindi richiesto di caricare e salvare nella memoria locale del tablet le *mesh* disponibili sul server, per poi poter essere visualizzate ed esaminate.

1.4.3 L'applicativo attuale: VIC-Tango

L'ultimo prototipo si avvicina già ad uno stato più stabile e funzionale e si pone come base su cui sviluppare l'applicazione in futuro. VIC-Tango è una versione rifinita e migliorata di *Samba*, della quale sono stati risolti svariati bug, immancabili allo stadio prototipale. Introduce inoltre alcune nuove funzionalità che migliorano la User experience e lasciano la porta aperta a sviluppi futuri.

Firestore

Firestore è un *framework* associato all'omonimo *web service* offerto da *Google* per lo scambio di messaggi tra applicazioni *Android* e un *server*. In particolare è stato utilizzato il servizio *Firestore* di cloud messaging per la gestione e invio affidabile di

messaggi a più dispositivi differenti.

Il suo utilizzo in VIC-Tango ha permesso di implementare un servizio di notifica da parte del server, che può così inviare messaggi asincroni al device Tango, il quale li riceverà come notifica di sistema Android, informandolo ad esempio che il Point Cloud inviato è stato elaborato e una nuova mesh è stata generata. L'utente può così continuare l'ispezione e acquisire nuovi Point Cloud e, alla ricezione della notifica, caricare e visualizzare automaticamente le mesh quando disponibili.

Camera preview

Fin dal prototipo precedente era stata introdotta una piccola preview di quanto ripreso dalla fotocamera RGB-IR. Tuttavia la funzionalità era stata implementata utilizzando una soluzione general-purpose applicabile a più dispositivi Android, che causava però problemi prestazionali e crash occasionali dell'applicativo.

La preview è stata quindi reimplementata utilizzando le funzionalità esposte dalle Tango API, soluzione inizialmente scartata perchè quest'ultime erano mal documentate, ottenendo un miglioramento delle prestazioni e la correzione dei bug rilevati.

JNI e Point Cloud Registration

La Java Native Interface o JNI è un framework del linguaggio Java che consente al codice Java di richiamare codice cosiddetto "nativo", ovvero specifico di un determinato sistema operativo scritto in altri linguaggi di programmazione, in particolare C/C++. L'integrazione della JNI con lo sviluppo Android, in particolare il suo utilizzo nell'IDE Android Studio, è un compito tutt'altro che semplice, dato il supporto ancora scarso all'integrazione del framework con l'IDE e il build-tool Gradle.

Dopo svariati tentativi iniziati già i primi giorni di stage, quindi già dai prototipi precedenti, è stato infine possibile usufruire della JNI utilizzando versioni sperimentali di Android Studio e Gradle.

Attraverso la JNI si è provato ad importare la libreria PCL, così da poterne utilizzare le potenti funzionalità direttamente dal tablet. Purtroppo l'integrazione con il build-tool Gradle richiede un makefile apposito, compito improponibile per una libreria notevolmente complessa come PCL, le cui potenzialità sono quindi rimaste nel lato server.

Si è riuscito invece ad importare, date le ridotte dimensioni, una libreria per la *Registration* di Point Cloud. Col termine *Registration* s'indica il processo che trasforma in coordinate assolute e allinea due set di Point Cloud in un unico set che minimizza le distanze tra punti corrispondenti (ad es. fig. 1.6).

La libreria in questione, GOICP, implementa una versione ottimizzata dell'algoritmo ICP (Iterative Closest Point), spesso utilizzato in applicazioni nelle quali sia necessario ricostruire una superficie tridimensionale a partire da più scansioni, che punta a minimizzare la distanza tra i punti delle due Point Cloud.

L'algoritmo ICP tenta di minimizzare l'errore quadratico medio (in inglese Mean Squared Error, MSE) cioè la discrepanza quadratica media fra i valori dei punti della coppia di Point Cloud, definita come:

$$MSE = \frac{\sum_{i=1}^n (x_i - x'_i)^2}{n}$$

Si è testato GOICP sulle coppie di Point Cloud acquisite con VIC-Tango, per ovviare ai problemi già presentati del *drifting* (fig. 1.4a) e del *ghosting* (fig. 1.4b); si sono ottenuti però risultati insoddisfacenti. Sono sorti infatti i seguenti problemi:

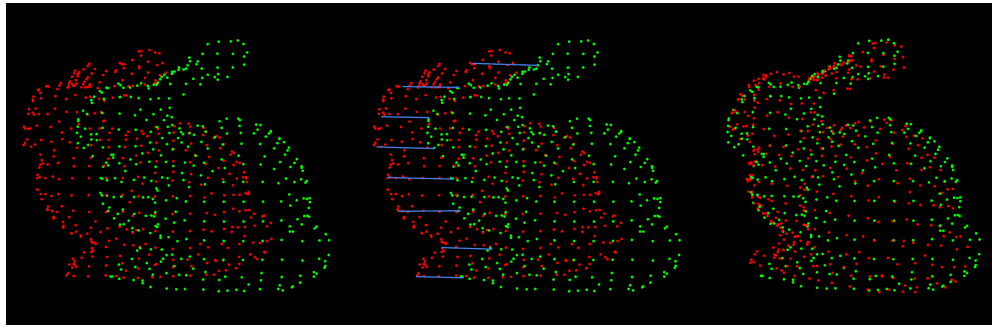


figura 1.6: *Point Cloud Registration* tramite l'algoritmo *ICP*

- * Il processo è eccessivamente dispendioso da effettuare su tablet, con tempi di elaborazione tra i 10 e i 60 secondi.
- * L'algoritmo ICP, basandosi sul MSE come valore di bontà, converge spesso lentamente e a soluzioni errate, per colpa della grande quantità di punti planari sul pavimento, che rendono l'errore quadratico medio un valore non attendibile su cui basare il successo di una *Registration*

Come si vede in figura 1.7c l'algoritmo, pur ottenendo un basso valore di MSE, non effettua la *Registration* desiderata, dato che vengono allineati perlopiù i punti planari.

Per questi motivi GOICP non potrà essere utilizzata nei futuri prototipi, se non si effettuerà almeno una delle seguenti correzioni:

- * Trovare una misura più adatta del MSE per valutare la convergenza dell'algoritmo ICP
- * Alternativamente, eliminare dal Point Cloud i punti planari appartenenti al pavimento così da poter utilizzare l'errore quadratico medio come affidabile valore di convergenza
- * Ottimizzare GOICP o implementare in algoritmo ICP ad hoc per ridurre drasticamente i tempi di elaborazione

Tali correzioni trattano problemi non triviali, che l'esiguo tempo di tirocinio rimasto non mi ha permesso di affrontare adeguatamente. La Point Cloud Registration rimane quindi una funzionalità necessaria ad assicurare un'affidabile e corretta ricostruzione degli oggetti scansionati, che dovrà essere implementata nei futuri prototipi.

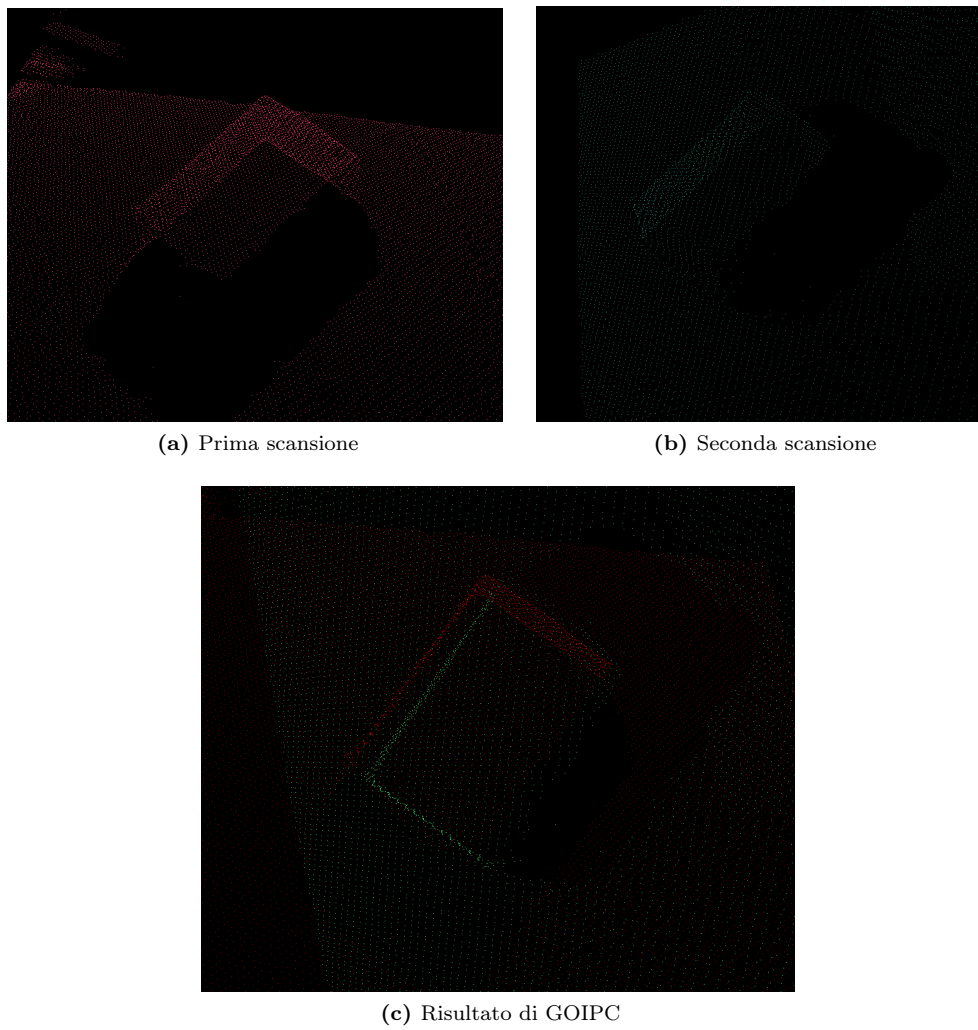


figura 1.7: Errore nella *Registration* con *GOIPC*

1.5 Il Prodotto - lato server

Il lato server dell'applicazione si occupa di ricevere dai *Tango device* i Point Cloud acquisiti, elaborarli utilizzando la libreria PCL per estrarre i punti rappresentanti il solo oggetto scansionato, produrre una mesh dell'oggetto così ottenuto e calcolarne il volume. Invia poi quando necessario i risultati elaborati ai dispositivi che li richiedono. Quando un Point Cloud viene ricevuto dal server tramite una richiesta HTTP POST, inserito in un oggetto JSON, viene salvato in un file PCD. Il formato PCD (Point Cloud Data) usato nella libreria PCL è molto semplice: è formato da un header contenente alcune informazioni utili come il numero totale di punti e il tipo e numero di valori associati ad ogni punto; segue poi l'effettivo Point Cloud codificato riga per riga, dove ogni riga contiene una successione di valori che rappresenta un punto.

Il file salvato viene poi elaborato sfruttando la libreria PCL.

1.5.1 Point Cloud Library

PCL o *Point Cloud Library* è una libreria *Open Source*, scritta in C++, di algoritmi per l'elaborazione di *Point Cloud* tridimensionali. Contiene algoritmi per il filtraggio, la segmentazione, la *registration* e il *meshing* di Point Cloud, per citarne alcuni.

La libreria è ampiamente utilizzata da qualsiasi applicativo debba trattare nuvole di punti, ed oltre ad essere utile ed efficiente è anche ben documentata, con molti esempi d'utilizzo reperibili online.

Le notevoli potenzialità della libreria sono state utilizzate intensivamente nell'applicativo con lo scopo di isolare i punti appartenenti all'oggetto scansionato dal resto del Point Cloud, ed effettuarne quindi il meshing.

1.5.2 Elaborazione di un Point Cloud

La nuvola di punti salvata in un file PCD viene elaborata da un applicativo scritto in C++ che sfrutta le numerose funzionalità della PCL. L'elaborazione di un Point Cloud è composta di più passi sequenziali:

1. Viene caricato il nuovo Point Cloud da file PCD salvato localmente
2. Vengono rimossi i punti isolati della nuvola attraverso la libreria Sparse Filtering
3. Vengono rimossi i punti troppo esterni della nuvola attraverso la libreria Radius Filtering
4. Vengono rimossi i punti che corrispondono al piano del pavimento attraverso la libreria Ground Filtering
5. Viene effettuato il *downsample* del *dataset* attraverso la libreria Voxel Filtering
6. Viene estratto il cluster maggiore dai punti rimanenti attraverso la libreria Cluster Extraction
7. Viene effettuato il meshing del cluster estratto, e la mesh viene salvata in due file OBJ e VTK
8. Viene calcolato e salvato il volume della mesh prodotta

Per ogni passo di filtraggio la nuvola di punti viene salvata su un distinto file PCD, per poter analizzare visivamente la bontà dell'elaborazione. Di seguito vengono descritti in dettaglio filtri applicati e il processo di meshing.

Sparse filtering

Questo primo passo di filtraggio si occupa di rimuovere dal Point Cloud i punti isolati. Per ogni punto viene effettuata una analisi statistica della distanza media dai suoi vicini. I punti la cui distanza dai vicini supera un certo valore sono quindi isolati e possono essere eliminati dalla nuvola. Un esempio di tale tecnica, sul quale si basa l'implementazione di questo filtro, è lo "Statistical Outlier Removal"¹ di PCL.

In figura 1.8 vediamo i risultati di tale filtro, con una rimozione di circa 10.000 punti.

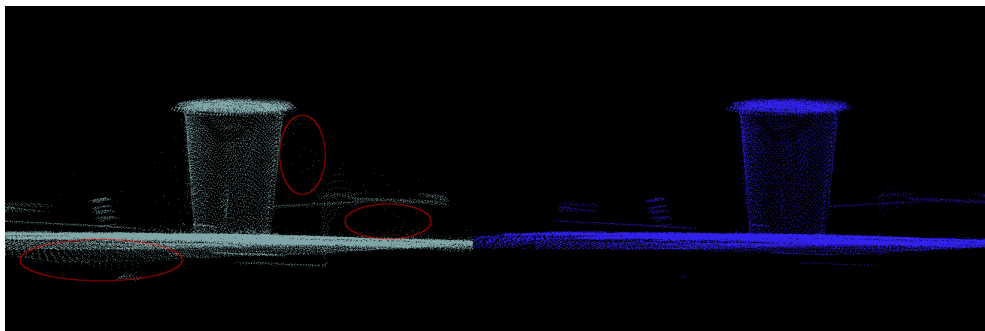


figura 1.8: *Sparse filtering*

Radius filtering

Questo passo di filtraggio si occupa di eliminare i punti troppo distanti dal centro, che solitamente appartengono alle pareti circostanti o comunque ad oggetti diversi da quello scansionato, che sarà sempre verso il centro della nuvola.

Partendo da questo presupposto, il filtro calcola il punto centrale del Point Cloud, media del valore di tutti i punti, e procede con l'eliminare i punti che superano una certa distanza (*radius*), attraverso l'algoritmo PCL "PassThrough Filter"², fino a che non è stata eliminata una definita percentuale del point set iniziale.

In figura 1.9 vediamo i risultati di tale filtro, con una rimozione di circa 24.000 punti.

Ground filtering

Questo passo di filtraggio si occupa di eliminare i punti planari appartenenti al pavimento, problema che si è rilevato essere uno dei più difficili da trattare. In primo luogo non sempre c'è un unico componente planare che rappresenta il pavimento, ma solitamente ve n'è più d'uno, a causa delle imprecisioni nell'acquisizione dei Point Cloud lato client. Per l'implementazione del filtro si è fatto uso degli algoritmi di *segmentation* di PCL (ad es. "Planar segmentation"³), che consentono di estrarre i punti della nuvola che giacciono sullo stesso piano. Senza ulteriori aggiustamenti però l'algoritmo elimina anche piani utili, come il lato superiore di una scatola o di un tavolo. Si è reso quindi necessario applicare i seguenti passi:

1. Calcolare l'altezza del Point Cloud e dividere la parte superiore da quella inferiore, che deve contenere il pavimento

¹http://pointclouds.org/documentation/tutorials/statistical_outlier.php/statistical-outlier-removal.

²<http://pointclouds.org/documentation/tutorials/passthrough.php/passthrough>.

³http://pointclouds.org/documentation/tutorials/planar_segmentation.php.

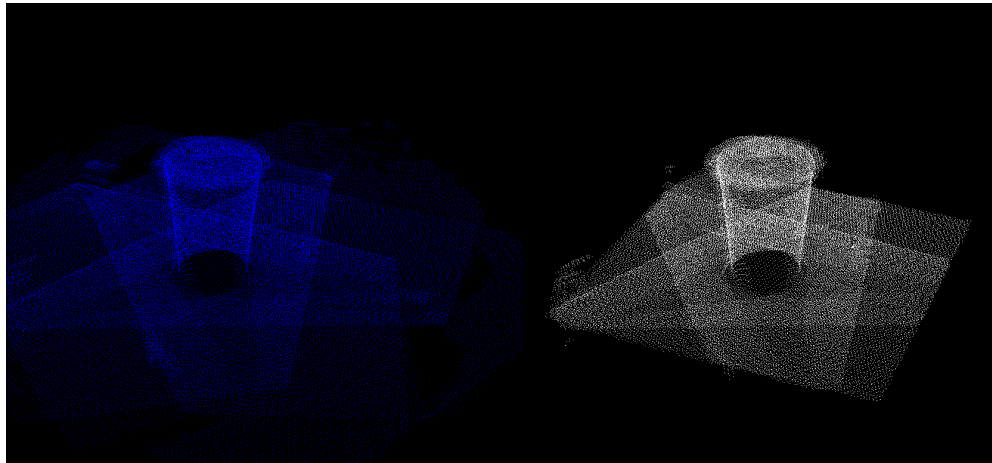


figura 1.9: *Radius filtering*

2. Applicare l'algoritmo di *planar segmentation* alla sotto-nuvola estratta fino a che non è stato eliminato un numero adeguato di punti
3. Ricongiungere la parte inferiore filtrata alla parte superiore del Point Cloud

Il filtro così impostato ottiene ottimi risultati nella maggior parte dei casi. Tuttavia se il Point Cloud originale presenta troppo *drifting*, e i piani del pavimento sono molto distanti tra loro, diventa impossibile eliminarli senza togliere anche punti importanti dell'oggetto scansionato.

In figura 1.10 vediamo i risultati di tale filtro, applicato ad un Point Cloud già sottoposto ai filtri precedenti, con una rimozione di circa 50.000 punti.

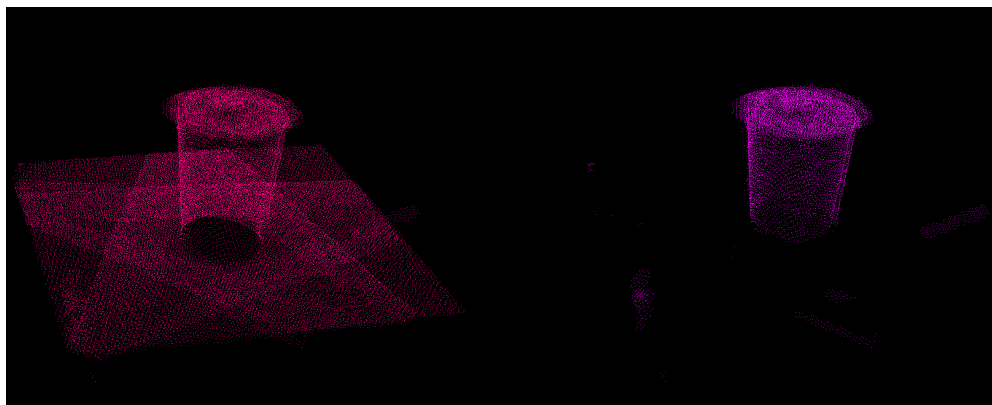


figura 1.10: *Ground filtering*

Voxel filtering

Questo passo di filtraggio si occupa di eliminare i punti doppi e ridurre le dimensioni del Point Cloud per il futuro meshing. Il filtro utilizza la stessa tecnica già implementata nel lato client, cioè suddividere lo spazio in tanti *voxel* cubici e calcolare il punto

medio di ogni voxel come approssimazione dei punti che ricadono al suo interno. L'implementazione prende spunto dall'esempio di PCL "VoxelGrid Filter"⁴. Per i risultati del *voxel filtering* si rimanda alle figure 1.5a e 1.5b.

Cluster extraction

Meshing

1.6 Calcolo del volume

1.7 Organizzazione del testo

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*.

⁴http://pointclouds.org/documentation/tutorials/voxel_grid.php.

Capitolo 2

Processi, tecnologie e strumenti

Brevissima introduzione al capitolo

2.1 Processo sviluppo prodotto

In ambito aziendale si è scelto di provare, per questo progetto, un processo di sviluppo *software* basato sulla filosofia *Lean*.

Dato che l'obiettivo principale era fornire un prototipo ci si è limitati solamente alle tre fasi iniziali dello sviluppo di *Lean*, ovvero *Kick-Off*, *Concept Preview* e *Product Prototype*.

2.1.1 Kick-Off

Questa è la prima fase dello sviluppo *software*, coincide con la prima riunione ufficiale del team di progetto, aperta anche agli *Stakeholder*.

Si pone lo scopo di iniziare la fase l'*allestimento* e l'*avviamento* in cui viene determinata la natura e lo scopo del progetto.

2.1.2 Concept Preview

Fase in cui è reso disponibile un primo campione di prova del prodotto, detto *concept*. Esso può essere incompleto e affetto da errori, ma deve essere in grado di dimostrare agli *stakeholder* le caratteristiche principali che avrà il prodotto finito.

Esso è soggetto a un riesame che ha lo scopo di valutare se è in linea con gli obiettivi definiti nella *Value Proposition*, la *milestone* non può essere raggiunta senza che questo riesame abbia esito positivo.

Questa fase coincide anche con l'inizio della progettazione, che deve essere portata avanti fino ad un livello di dettaglio ritenuto opportuno dal *team*.

2.1.3 Product Prototype

Fase in cui è messo a disposizione il primo prototipo del nuovo prodotto, completo nelle sue funzioni (sviluppo finito) ma non ancora messo a punto mediante verifiche

e correzioni, per garantirne funzionalità e prestazioni. Il prototipo deve essere ad un stato tale da poter essere dato in valutazione agli *stakeholder*.

Esso è soggetto a un riesame che ha lo scopo di valutare se è in linea con gli obiettivi definiti sia *Value Proposition* che nella *Requirements Specification*, la *milestone* non può essere raggiunta senza che questo riesame abbia esito positivo.

Questa fase coincide con il termine della fase di progettazione e l'inizio della fase di esecuzione, cioè l'insieme dei processi necessari a soddisfare i requisiti del progetto.

2.1.4 Fasi successive

Le fasi successive, ovvero *Product Design Freeze* e *Start Of Production*, possono essere avviate nel futuro a partire dal *Product Prototype* se ciò verrà ritenuto opportuno dall'azienda.

2.2 Tecnologie e Strumenti

L'azienda ha lasciato grande libertà riguardo agli strumenti da utilizzare per questo progetto, quindi essi sono stati fissati inizialmente e successivamente incrementati al crescere delle necessità.

2.2.1 Codice

Segue la lista degli strumenti utilizzati per la codifica.

- * **Java**¹: Il linguaggio preferito per l'applicazione lato *tablet*. È stato scelto seguendo le *Best Practice* dello sviluppo *Android*.
- * **C++**²: Il linguaggio preferito per l'applicazione lato *Server*. È stato scelto perché tutti le più diffuse librerie per l'elaborazione dei *Point Cloud*, ed in particolare *PCL* sono disponibili in questo linguaggio.
- * **PCL, Point Cloud Library**³: Una delle librerie di maggior rilievo nel campo della *Computer Vision*, mette a disposizione notevoli funzionalità per l'elaborazione, il filtraggio e l'ottimizzazione dei *Point Cloud*.
- * **Tango API**⁴: Le *API* ufficiali per lo sviluppo di applicazioni *Tango*.
- * **PHP**⁵: Il linguaggio usato per ricevere ed inviare le richieste *HTTP* necessarie alla comunicazione tra *Server* e dispositivo.
- * **Python**⁶: Il linguaggio usato in combinazione con *PHP* per implementare la logica del lato *Server*.

¹<https://www.java.com>

²www.cplusplus.com

³<http://pointclouds.org>

⁴<https://developers.google.com/tango/apis/overview>

⁵<https://secure.php.net>

⁶<https://www.python.org>

2.2.2 IDE ed editor

Segue la lista degli ambienti per la codifica utilizzati durante il progetto.

- * **Android Studio**⁷: L'*IDE* ufficiale per le applicazioni *Android*.
- * **QT**⁸: L'*IDE* scelto per lo sviluppo del codice *C++*.
- * **Geany**⁹: L'*editor* di testo usato per scrivere gli *script php* e *Python*.

2.2.3 Framework

Segue la lista dei *Framework* usati per durante il tirocinio.

- * **Gradle**¹⁰: È stato usato come *tool* di *build* per tutta l'applicazione *Android*.
- * **Rajawali3D**¹¹: È stato usato come *framework* grafico per la realizzazione del *render*.
- * **OkHttp**¹²: È stata usata come *framework* di riferimento per le richieste *http*, come indicato nella *Android Best Practices*.
- * **TangoUx**¹³: *Framework* messo a disposizione dalla *Google* assieme alle *API Tango*. È stato usato per gestire le notifiche all'utente relative ai comportamenti che deve tenere per permettere il buon funzionamento del dispositivo e dei sensori.
- * **Jni**¹⁴: Questo *framework* permette di richiamare metodi 'nativi' (scritti in *C/C++*) dal codice *Java*.
- * **Firebase**¹⁵: *Framework* associato all'omonimo *web service* offerto da *Google* per lo scambio di messaggi tra applicazioni *Android* e un *server*.
- * **GOICP**¹⁶: libreria che implementa una variante dell'algoritmo *ICP* per la *Registration* di set di *Point Cloud*.

⁷<https://developer.android.com/studio/index.html>

⁸<https://www.qt.io/ide>

⁹<https://www.geany.org>

¹⁰<https://gradle.org>

¹¹<https://github.com/Rajawali/Rajawali>

¹²<http://square.github.io/okhttp>

¹³<https://developers.google.com/tango/ux/ux-framework>

¹⁴<http://docs.oracle.com/javase/7/docs/technotes/guides/jni>

¹⁵<https://firebase.google.com>

¹⁶<http://iitlab.bit.edu.cn/mcislabs/yangjiaolong/go-icp/>

Capitolo 3

Studio di fattibilità ed analisi dei rischi

Breve introduzione al capitolo

3.1 Introduzione al progetto

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.

1. Performance del simulatore hardware

Descrizione: le performance del simulatore hardware e la comunicazione con questo potrebbero risultare lenti o non abbastanza buoni da causare il fallimento dei test.

Soluzione: coinvolgimento del responsabile a capo del progetto relativo il simulatore hardware.

3.3 Requisiti e obiettivi

3.4 Pianificazione

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'I-DE.

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 requisiti 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 Consuntivo finale

7.2 Raggiungimento degli obiettivi

7.3 Conoscenze acquisite

7.4 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Bibliografia

Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

Siti Web consultati

Manifesto Agile. URL: <http://agilemanifesto.org/iso/it/>.