POLITECNICO DI TORINO

UNIVERSIDAD POLITÉCNICA
DE MADRID

I Facoltà di Ingegneria

Laurea Magistrale in Ingegneria Biomedica
A.A. 2017/2018

Tracciamento di strumenti laparoscopici basato su visione al computer in dispositivi mobili di tipo Android

Relatori:
Prof. Filippo Molinari
Prof. Ignacio Oropesa Garcìa

Candidata:
Michela Mollo

Marzo 2018

Indice

Indice Figure

Indice Grafici

Indice Tabelle

Abstract

Nel corso di questi ultimi anni la chirurgia e la medicina in generale, hanno subito uno sviluppo tale da non voler solo fermarsi a guarire le diverse patologie, ma tentando anche di ridurre al minimo le problematiche post-operatorie.
Con la nascita della chirurgia mininvasiva si è in grado di operare sfruttando piccole vie di accesso, al fine di diminuire lo shock dovuto all'atto chirurgico nel distretto interessato.
La chirurgia mininvasiva può risultare più complicata del previsto, in quanto, non mette a disposizione una visione tridimensionale dello spazio da operare, ma bidimensionale, ed in aggiunta, non vi è presente alcuna prospettiva che possa aiutare i chirurghi ad orientarsi.
Si è pensato così di creare, sulla base di un'applicazione già esistente implementata per i computer, un'applicazione per dispositivi mobili di tipo Android, in grado di valutare le abilità motrici dei chirurghi.
Ciò è stato reso possibile grazie all'aiuto di algoritmi capaci di effettuare un'analisi video-endoscopica che monitorasse i movimenti degli strumenti laparoscopici durante l'esecuzione di determinate attività.
Tutto questo è stato reso possibile migrando il codice esistente in C++ al linguaggio Java ed implementandolo nel software Android Studio.
Una volta completata l'applicazione, è stata messa alla prova più volte in modo da verificare la correttezza e la precisione dei risultati ottenuti.
Per portare a termine le prove sono stati utilizzati un simulatore, alcuni strumenti laparoscopici, una struttura in legno, un tablet, uno smartphone e dei marcatori con proprietà termo-restringenti; il tutto fornito dal centro di chirurgia mininvasiva *Jesùs Usòn* e dall'*Universitad Politécnica* di Madrid (ETSIT).
Come ci si aspettava i risultati sono stati soddisfacenti ed esaurienti.
Si spera che tale applicazione venga utilizzata in futuro in quanto potrebbe realmente consentire all'apprendista chirurgo di perfezionare le proprie conoscenze pratiche.
Inoltre, si aumenterebbe la sicurezza del paziente ritardando il momento in cui il neo-chirurgo sarà pronto ad operare, fino a che non avrà le competenze adeguate per farlo.

Abstract

During these last years, the surgery and the medicine in general developed, not with the only aim to cure different pathologies, but also trying to reduce to the minimum the post-operative set of problems.
Thanks to the birth of the minimally invasive surgery it is possible to operate using small access ways so to be able to reduce the shock due to the surgical act on the interested area. But the minimally invasive surgery can result more complicated than how expected because there is no more tri-dimensional vision of the space where operate, but only a bi-dimensional one and, moreover, there is no perspective able to help surgeons.
An idea came out: to create, basing on an already existing app activated for computers, a new application for Android mobile devices able to evaluate the surgeons' motors skills.
It has been made possible thanks to the algorithms able to effectuate a video-endoscopic analysis that would have monitored the laparoscopic tools movements during some tests, so to be able to use them as primary source of information to make the evaluation.
All this has been made possible migrating the existing code in C++ to the Java language, activating it in Android Studio software. Once the application was completed, it was tested many times, thus to verify the obtained results correctness and precision.
To complete the tests it has been used a simulator, laparoscopic tools, a wooden structure, a tablet, a Smartphone and some markers with heat-shrinkable properties; everything was provided by the Jesùs Usòn minimally invasive surgery centre and by the Universidad Politécnica de Madrid (ETSIT).
As it was predicted, the results have been satisfying and complete. We hope that this application will be used in the future because it could consent the learner surgeon to improve their practical knowledge. Also the patient security will be improved, delaying the moment in which the neo-surgeon will be ready to operate until they will have the competences to do it.

Introduzione

In questi ultimi anni la chirurgia mininvasiva, chiamata anche MIS da *Minimally Invasive Surgery*, grazie ai suoi notevoli vantaggi, è riuscita a scavalcare la chirurgia tradizionale. La chirurgia mininvasiva include gli interventi laparoscopici, toracoscopici e anche interventi in organi cavi come l'endoscopia [1].

Con la MIS infatti, si è in grado di operare sfruttando piccole vie di accesso da poter diminuire lo shock dovuto all'atto chirurgico [2].

Figura 1: Intervento di chirurgia mininvasiva

Una volta anestetizzato il paziente, si procede con un taglio iniziale nella zona da operare, attraverso il quale viene insufflata anidride carbonica per gonfiare tale cavità. Ciò permette al chirurgo una maggior visualizzazione del campo operatorio e uno spazio più vasto nel quale potersi muovere. Successivamente vengono eseguiti altri due piccoli tagli vicini al primo, per introdurre all'interno del corpo del paziente, tramite un dispositivo chiamato *trocar,* gli strumenti laparoscopici e il laparoscopio. Il laparoscopio è uno strumento contenente un insieme di fibre ottiche in grado di funzionare sia da sorgente luminosa che da telecamera [2]. Grazie a tale dispositivo possono essere visualizzate in tempo reale le immagini illuminate in modo che il chirurgo possa orientarsi al meglio all'interno del paziente [2].

I vantaggi ricavati dall'utilizzo della MIS sono molteplici, in *primis* vi è la rapida ripresa del paziente dovuta a una ferita di dimensioni notevolmente minori, con conseguente diminuzione del tempo di degenza e riduzione dei costi (sia intraoperatori che di ospedalizzazione) [2].
Inoltre, una ferita più piccola porta ad avere meno addome esposto durante l'intervento, ciò significa che diminuiscono anche il rischio di infezioni oltre alle complicazioni post-

operatorie. È stato infatti dimostrato che una minor esposizione di tessuti porti ad avere una miglior risposta immunitaria [3].

La chirurgia mininvasiva perciò, ha ormai sorpassato la chirurgia tradizionale sia in termini di costi che di maggior sicurezza per il paziente.

Tuttavia la chirurgia mininvasiva può portare ad avere complicazioni anche molto gravi se i chirurghi che la praticano non conoscono a fondo i vari procedimenti [2]. I chirurghi si trovano a dover operare avendo una visione bidimensionale dello spazio, senza prospettiva e senza poter neanche sfruttare le loro percezioni tattili [2].
Per ovviare a tali problemi è possibile utilizzare sistemi di apprendimento ed allenamento che sono costituiti principalmente da simulatori in grado di riprodurre ambienti da operare. Con tali allenamenti il chirurgo imparerà a sapere cosa fare, come e quando farlo.
Imparerà a manipolare oggetti attraverso l'uso di strumenti laparoscopici, differenziando i vari tessuti tra loro. Sarà in grado di scegliere punti anatomici di riferimento al fine di non perdere l'orientamento spaziale e sapere che ad ogni movimento ne corrisponderà uno uguale e contrario degli strumenti laparoscopici.
Capirà inoltre come fare a muoversi in maniera coordinata con entrambe le braccia senza lesionare tessuti innocenti ed imparerà come poter esporre solo il tessuto da tagliare gestendo i gradi di libertà a disposizione [2].

Esistono numerosi modelli di simulatori che possono essere classificati in simulatori fisici o simulatori virtuali. I simulatori fisici comprendono i *box trainer* e gli strumenti laparoscopici, mentre di quelli virtuali fanno parte i sistemi *computer-based* che utilizzano software di realtà virtuale [2]. Entrambi i tipi di simulatori verranno discussi più nel dettaglio nel paragrafo 1.3 Simulatori di realtà virtuale.

L'obiettivo preposto da questa tesi è proprio quello di migliorare le nuove tecnologie in grado di aiutare i neo-chirurghi a comprendere al meglio le procedure da utilizzare nella chirurgia mininvasiva.
In particolare verrà migliorato un sistema di valutazione chirurgica già esistente al fine di trarre una valutazione sulle abilità pratiche dei chirurghi.
Tale sistema è il sistema di monitoraggio EVA, il quale permette di estrarre i movimenti degli strumenti laparoscopici utilizzando una tecnica di visione artificiale[1] [4].
Il sistema è in grado di ottenere la posizione spaziale della punta dello strumento sulla base delle immagini 2D [4].

[1] L'insieme di processi che, partendo da immagini bidimensionali, mirano a costruire un modello 3D più simile al mondo reale

Il principale vantaggio di tale sistema è che per ottenere le posizioni 3D non richiede l'utilizzo di sensori esterni [4]. EVA necessita solamente di conoscere le proprietà geometriche degli strumenti laparoscopici ed i parametri intrinseci della fotocamera che si sta utilizzando [4]. Tale sistema di tracciamento può monitorare nello stesso momento un massimo di due strumenti laparoscopici [1].

EVA pertanto, viene considerato dai ricercatori come un sistema di tracciamento portatile ed a basso costo, capace di prevedere l'esperienza del chirurgo e formarlo nel caso in cui questa non fosse ritenuta del tutto adeguata [1].

Proprio tale sistema di monitoraggio EVA è stato utilizzato in questo progetto di tesi migrando l'intero codice dal linguaggio C++ al linguaggio Java, così da poterlo implementare in Android Studio e sfruttarlo anche da dispositivi mobili. È nostra intenzione aumentare la versatilità e la portabilità dei sistemi *box trainer*. È stata sviluppata un'applicazione per migliorare e facilitare l'uso di tale sistema, in quanto i dispositivi mobili come smartphone e tablet sono più accessibili e semplici da utilizzare da parte dei chirurghi.

Per motivi di riservatezza legati all'Università Politecnica di Madrid, in cui è stato svolto l'intero progetto di tesi, non è possibile mostrare nè i codici nè parte di questi.
Pertanto nel paragrafo 3.1.4 Codice calibrazione camera e nel
3.1.5 Codice *tracking* C++, nei quali verranno presentati i codici, si cercherà di spiegarli al meglio per quanto è concesso.

Il presente documento è strutturato come segue:

Nel Capitolo I verranno mostrati i risultati di una breve ricerca condotta sullo stato dell'arte, affrontando argomenti riguardanti l'utilizzo di dispositivi mobili durante interventi chirurgici e riguardo sistemi in grado di visualizzare e registrare il movimento degli strumenti laparoscopici.

Nel Capitolo II verrànno discussi gli obiettivi da raggiungere per portare a termine questo progetto di tesi. Si discuterà in particolare di ciò che si intende fare, ovvero sviluppare due applicazioni, una in grado di effettuare la calibrazione della camera sulla base dei marcatori e l'altra capace di monitorare al massimo due strumenti laparoscopici. Verrano affrontati brevemente anche gli obiettivi più specifici pianificati durante lo svolgimento del lavoro.

Nel Capitolo III verranno presentati tutti i materiali utilizzati partendo dal software Android Studio e concludendo con i sistemi hardware. Saranno elencate, in ordine cronologico, tutte le fasi di lavoro che ci hanno permesso di riuscire a sviluppare l'applicazione con la spiegazione dettagliata delle prove usate per verificarne la precisione.

Nel Capitolo IV si potranno visualizzare tramite immagini tutte le interfacce dell'applicazione risultante, ciascuna accompagnata da una breve spiegazione. Verranno anche discussi i risultati ottenuti dalle prove effettuate.

Nel Capitolo V saranno affrontate le conslusioni finali seguite dalle possibili ottimizzazioni future da apportare all'applicazione al fine di renderla il più completa possibile.

Nel Capitolo VI sarà possibile trovarvi la bibliografia contenente tutti i libri e gli articoli consultati.

In ultimo, verranno mostrati il glossario contenente i grafici che non sono stati posti nel capitolo dei risultati e il glossario contenente le abbreviazioni.

1. Stato dell'arte

In questo capitolo verrà esposta una breve ricerca sullo stato dell'arte circa:

- Le metriche di valutazione usate per ottenere feedback oggettivi sulle prestazioni pratiche di un chirurgo;
- Sistemi di monitoraggio degli strumenti laparoscopici, tra cui il sistema EVA;
- Principali simulatori di realtà virtuale (VR);
- Sistemi di monitoraggio con le tecnologie *Human Motion Tracking* (HMT);
- Sistemi di supporto per la valutazione chirurgica.

1.1 Metriche di valutazione

Per metriche di valutazione si intendono quell'insieme di parametri che vengono analizzati per ottenere delle valutazioni sulla base di compiti svolti.
Prima di poter scegliere i parametri adeguati da tenere sotto controllo, bisogna decidere, quali competenze del chirurgo devono essere valutate.
In seguito, sulla base di queste, si andranno a definire le attività da svolgere.
Le attività da svolgere sono esercizi volti a migliorare le abilità pratiche dei neo-chirurghi.
Sulla base delle abilità da migliorare dipendono i materiali di cui le attività sono costituite.

Le metriche a disposizione per la valutazione possono essere di efficienza o di qualità [5].
Le metriche di efficienza sono correlate a parametri fisici misurabili, come per esempio la velocità del movimento. Quelle di qualità si riferiscono alla definizione e all'esecuzione di un'attività.

I parametri più importanti riguardanti le metriche di qualità sono gli errori commessi durante l'esecuzione dell'attività [2].
Vediamo in Tabella 1 tutte le metriche di efficienza e di qualità presenti in letteratura.

Tabella 1:Metriche di valutazione [2]

Efficiency Metrics	Description
Time	*Total time to perform a task (s)*
Path length	*Total path followed by the laparoscopic instrument (m)*
Economy of movements	*Shortest distance possible to accomplish task/total distance (%)*
Economy of diathermy	*For diathermy: optimal burn time/excess burn time (%)*
Speed	*Rate of change of the instrument's position (m/s)*
Motion smoothness	*Abrupt changes in acceleration resulting in jerky movements of the instruments (m/s^3)*
Instrument orientation	*Amount of instrument rotation, measures the ability of correctly placing the instrument (rad)*
Depth	*Total path length travelled in the instrument 's axis direction (m)*
Angular path	*Sum of all angular paths about the instrument's pivot point (°)*
Angular area	*Area between the farthest positions occupied by the instrument in the camera plain (rad^2)*
Volume	*Angular area x Depth ($rad^2 \cdot m$)*
Force/torque	*Instrument – tissue force (N) and torsion (N·m) interactions*
Quality Metrics	**Description**
Outcome	*Final score of the task performed (end-product analysis). Task-dependent*
Task completion	*Indication of whether a trainee has completed a task or not*
Errors	*Errors performed during the task. Task dependent*
Idle states	*Time periods when instrument movements/interactions are minimal*
Task repetitions	*Number of trials required on a task/subtask before achieving satisfactory completion*
Collisions/tissue damage	*Detection of incorrect collisions and damage performed to background elements (e.g.: VR tissues)*

Tra tutte le metriche, le più importanti e significative sono il tempo e il percorso [6].
Si pensa che il percorso ideale sia quello costituito da una linea retta che congiunge il punto di partenza all'obiettivo finale. Tuttavia non è sempre così, in quanto il percorso ideale dipende anche dalla fluidità del movimento [2].
Un chirurgo più esperto, rispetto ad un tirocinante, sarà in grado di eseguire i compiti in un tempo minore, con movimenti più limitati e mostrando una marcata percezione dello spazio chirurgico a disposizione [7]. Inoltre, è stato osservato che i chirurghi più esperti utilizzano valori di forza maggiori durante le procedure di sutura [8].
Proprio in tali procedure vengono valutate le metriche relative alla forza che insieme al rilevamento degli stati di inattività possono portare a comprendere al meglio il livello di abilità pratiche dei neo-chirurghi.

I periodi più brevi di stati di inattività differenziano un chirurgo principiante da un esperto, questo perchè un chirurgo esperto impiega più tempo nel decidere l'azione successiva da compiere [2].
In ultimo, vengono valutati anche eventuali danni riportati ai tessuti sulla base di ferite ed urti da parte di strumenti laparoscopici, dovuti ad una non completa padronanza della percezione dello spazio [9].
La competenza del chirurgo è un parametro difficile da valutare perchè dipende da molti fattori, tra cui troviamo conoscenza e giudizio a cui è complicato attribuire un punteggio [2].
Il sistema di monitoraggio di strumenti laparoscopici EVA, da cui siamo partiti per il nostro progetto di tesi, usa tutte le metriche di valutazione citate prima per arrivare ad avere un giudizio finale sui compiti svolti dai chirurghi.

1.2 Sistemi di monitoraggio di strumenti laparoscopici

1.2.1 Sistema di monitoraggio EVA

Come accennato nel capitolo precedente, il sistema di monitoraggio EVA permette di estrarre i movimenti degli strumenti laparoscopici utilizzando una tecnica di visione artificiale [2].

Per ottenere la posizione spaziale della punta dello strumento laparoscopico sulla base delle immagini 2D EVA, ha bisogno di conoscere le proprietà geometriche degli strumenti laparoscopici ed i parametri intrinseci della fotocamera che si sta utilizzando [1], senza l'utilizzo di sensori esterni [2].

Il processo di validazione consisteva nel confrontarlo con il sistema, già ampiamente validato e utilizzato, TrEndo [4].
Sono stati selezionati 42 partecipanti in modo che eseguissero una prova da poterne ricavare una valutazione sulla base di dieci metriche relative al movimento, tra cui la lunghezza del percorso, la profondità, la velocità e l'accelerazione media [4].
Tutti i partecipanti non avevano alcuna fomazione preliminare relativa al compito proposto. La prova consisteva nel prendere, trasportare e inserire un oggetto di forma sferica all'interno di fori consecutivi utilizzando un solo strumento laparoscopico [4].
Il tutto veniva svolto all'interno di un simulatore, costruito sulla base di criteri dettati dalla gestione dell'ospedale a seconda degli obiettivi proposti [4].
È stato tutto monitorato e registrato a video, sia dal sistema Eva che da TrEndo, per elaborarlo e analizzarlo successivamente tramite il software Matlab.
Non è stato definito alcun tempo massimo per terminare la prova e non è stato richiesto il completamento forzato [4].

Tale confronto ha rilevato correlazioni tali da poter arrivare a dimostrare la validità di EVA [4].

1.2.2 Monitoraggio con dispositivo iPhone

Nell'ospedale pediatrico Federico Gómez in Messico, è stato creato un sistema di allenamento laparoscopico utilizzando un iPhone 5 ed una custodia in plastica per simulare la cavità addominale di un adulto [10].

Figura 2:Sistema di allenamento laparoscopico con iPhone5 e custodia in plastica [10]

La custodia in plastica è stata forata per permettere agli strumenti laparoscopici di entrarvi. Inoltre, è stata creata una finestra per appoggiare la fotocamera dello smartphone come è possibile vedere in Figura 2 [10].

Tale fotocamera dispone di 8 megapixel con registrazione video ad alta definizione, autofocus e tre zoom [10]. Una volta collegata ad uno schermo tramite un adattatore digitale, per poter dimostrarne l'efficacia, è stata validata confrontandola con un *trainer* laparoscopico già in uso. Come metrica di valutazione viene misurato il tempo necessario a terminare la prova [10]. Tale sistema, chiamato *iPhone Trainer*, ha dato risultati soddisfacenti per quanto riguarda l'allenamento della coordinazione occhio-mano, lo sviluppo delle abilità laparoscopiche di base e il miglioramento della percezione della profondità [10]. Risulta essere una novità in quanto è portatile, semplice da sviluppare e a basso costo.

1.3 Simulatori di realtà virtuale

Una volta definiti i parametri da valutare bisogna decidere in che modo registrare i suddetti valori.

Per la registrazione delle attività c'è bisogno di sistemi che permettano ai tirocinanti di potersi allenare in ambienti controllati, senza la presenza costante di un supervisore [2]. Esistono numerosi modelli di simulatori, i quali possono essere classificati in simulatori fisici e simulatori virtuali.

I simulatori fisici comprendono i *box trainer* e la strumentazione laparoscopica, mentre i simulatori virtuali utilizzano software di realtà virtuale e sono sistemi *computer-based* [11]. In Tabella 2 è possibile visualizzare i vantaggi e gli svantaggi dei principali simulatori esistenti.

Tabella 2:Caratteristiche dei vari simulatori [11]

Training Model	Advantages	Limitations	Effect
Box trainers	Inexpensive, provide realistic haptic feedback	Box-trainers lack objective assessment of performance	basic laparoscopic skill acquisition such as hand-eye coordination, depth perception, and knot-tying
Hybrid simulators	Provide objective metrics for laparoscopic task performance	Lack haptic feedback	Basic and advanced laparoscopic skill acquisition
Virtual reality simulators	Provide objective metrics for laparoscopic task performance; provide explanations of practiced tasks	Lack haptic feedback	Basic and advanced laparoscopic skill acquisition
Augmented reality simulators	Provide realistic haptic feedback and objective assessment of the performance of the trainee	Expensive	Basic and advanced laparoscopic skill acquisition; ideal for laparoscopic suturing training
Laboratory animals	Provide realistic operative training experience	Expensive, anatomical differences from human body	Advanced laparoscopic skill acquisition, such as dissection, cutting, coagulation, and stitching
Cadaver models	Provide perfect anatomy, normal tissue consistency and a realistic operative training experience	Limited number of available human cadavers	Advanced laparoscopic skill acquisition, such as dissection, cutting, coagulation, and stitching

Tra i principali simulatori esistenti vi troviamo i *Box Trainers* che sono simulatori fisici in grado di offrire agli utenti un feedback di tipo aptico, tralasciando la possibilità di avere valutazioni oggettive sulle performance dell'operato.
Quest'ultimi permettono di acquisire una conoscenza laparoscopica di base.
Al contrario, gli *Hybrid simulators* e i *Virtual reality simulators* forniscono oltre a una valutazione sulle metriche oggettive anche conoscenze laparoscopiche avanzate, pur non disponendo di un'interfaccia aptica.
Gli *Augmented reality simulators* invece, uniscono al feedback aptico le metriche di valutazione oggettive, arrivando così ad avere un'ottima conoscenza sulle pratiche usate in laparoscopia.
Una conoscenza ancora più approfondita risulta dai *Laboratory animals* e dai *Cadaver models* che permettono ai tirocinanti di allenarsi in un ambiente realistico così da riuscire a differenziare i diversi tessuti immergendoli in ciò che è la realtà [2].

Tra tutti i simulatori di realtà virtuale, esistono simulatori sviluppati appositamente per operazioni specifiche come artroscopiche di ginocchio e spalla, o anche per colonscopie e per procedure oftalmologiche [2]. Per poterli classificare sulla base delle loro prestazioni si può utilizzare la Tabella 3.

Tabella 3: Simulatori di realtà virtuale:modelli, caratteristiche e studi di validazione [8]

SIMULATOR	TASKS	ANATOMIC SCENARIO	HAPTIC FEEDBACK	VALIDITY
MIST-VR	Basic/Procedural	Both	Both	OR: Gallagher et al. 2004
LapMentor	Basic/Procedural	Both	Yes	ProMIS: Brinkamn et al. 2012 OR: Cohen et al. 2009
LapSim	Basic/Procedural/ Advanced	Both	Yes	BT: Newmark et al. 2007
CAE ProMIS	Procedural	Hybrid simulator	Realistic interation	LapSim: Botden et al. 2007 OR: Rutter et al. 2007
SIMENDO	Basic/Advanced	No	No	BT: Verdaasdonik et al.2006 TrEndo: Chmarra et al. 2008a
SINERGIA	Basic	Both	Yes	
CAE lap VR Sim	Basic/Procedural/ Advanced	Yes	Yes	
SEP	Basic/Procedural/ Advanced	Both	No	

Nella Tabella 3 vengono caratterizzati i diversi simulatori virtuali esistenti con riferimento alle prestazioni che sono in grado di offrire.

Viene specificato se i compiti forniti sono basici, procedurali o avanzati [2], se hanno a disposizione scenari realistici di ambienti anatomici ed infine, se sono in grado di offrire un feedback aptico al tirocinante. Nell'ultima colonna della Tabella 3 vengono definiti gli studi di validazione con le relative risorse bibliografiche a cui fanno riferimento specificandone se possono essere utilizzati in *Box Trainer* (BT) o direttamente nelle *Operating Room* (OR).

Analizziamo ora nel dettaglio le diverse caratteristiche dei principali simulatori riportati in Tabella 3:

- MIST-VR: fornisce una valutazione sulle abilità psicomotorie dei chirurghi prendendo come riferimento gli errori fatti e il tempo impiegato per portare a termine le diverse prove [12]. Effettuando vari test è stata verificata la qualità di tale simulatore, che nonostante non sia dotato di feedback tattili, è in grado non solo di aiutare a ridurre il numero di errori ma di ridurre anche il tempo impiegato per terminare la prova [12].

- LAP MENTOR: è un simulatore virtuale che contiene oltre 70 diverse attività di chirurgia generale, toracica, urologia ecc... con differenti livelli di difficoltà. Man mano che l'apprendista chirurgo fa progressi, i livelli aumentano di difficoltà per consentirgli di avere la miglior preparazione possibile [13]. Tale simulatore dispone anche di un'interfaccia aptica al fine di poter riceverne in risposta sensazioni tattili da rendere l'esperienza più realistica. Lap Mentor oltre ad essere stimolante e divertente fornisce una preparazione completa e sufficientemente adeguata [13].

- LAP SIM: si tratta di una piattaforma hardware aptica dotata di una grafica potente ed in grado di ricevere feedback di forza. Contiene una serie di corsi tra i quali scegliere, consentendovi di personalizzarli a seconda degli obiettivi che si vuole raggiungere [14]. Gli esercizi vengono monitorati fino al termine della prova, in modo da ottenere i valori delle metriche di valutazione più rilevanti. Vengono inoltre forniti ripetutamente feedback sulle competenze acquisite [14].

- CAE ProMIS: permette agli apprendisti chirurghi di imparare e migliorare le tecniche usate negli interventi laparoscopici più frequenti, come l'appendicectomia, la rimozione della cistifellea e l'occlusione delle tube. Tale simulatore può essere utilizzato da uno o più chirurghi nello stesso momento ed è possibile pianificare i corsi che si hanno a disposizione definendone il livello di difficoltà [15]. Una volta terminato l'esercizio, al fine di comprendere meglio ciò che è stato fatto, i tirocinanti vengono interrogati sugli eventuali errori commessi. Anche con questo tipo di simulatore è possibile ottenere feedback [15].

- SIMENDO: è un simulatore in grado di allenare i tirocinanti per tutte le specialità laparoscopiche, tra cui le più importanti sono la chirurgia generale, la ginecologia e l'urologia. È collegato ad una piattaforma online con la quale vengono calcolati i punteggi finali, potendo così confrontarsi con altri tirocinanti ed esperti chirurghi al fine di rendere più stimolante l'attività da svolgere [16].

- SINERGIA: è un simulatore usato per l'allenamento e la valutazione delle abilità psicomotorie della formazione chirurgica nelle sue prime fasi [17]. Comprende più attività con differenti livelli di difficoltà e produce costantemente feedback. É possibile ottenere confronti con altri tirocinanti e scambiarsi dati per ottenere un'analisi statistica tramite l'interfaccia utente. Inoltre, vengono inviati messaggi ogni qualvolta che vengono commessi degli errori durante l'esecuzione di un'attività. Vi è la possibilità di tenere sotto controllo il punteggio globale della pratica [17].

Di seguito nella Tabella 4 possiamo trovare raggruppate le principali caratteristiche dei simulatori analizzati.

Tabella 4: Principali simulatori virtuali [2]

LapMentor (Simbionix, Lod, Israel – Cleveland, USA) • Simple and advanced tasks, surgical procedures • Realistic scenarios • Force feedback	
LapSim (Surgical Science Ltd, Göteborg, Sweden) • Simple and advanced tasks, surgical procedures • Realistic scenarios • Optional force feedback	
MIST-VR (Mentice AB, Göteborg, Sweden) • Simple tasks • Non-realistic scenarios • Optional force feedback	
Promis (Haptica, Dublin, Ireland – Boston, USA) • Simple tasks, surgical procedures • Real scenarios (Hybrid Simulator) • Optional force feedback	
SIMENDO (DeltaTech, Delft, Netherlands) • Simple tasks • Non-realistic scenarios • No force feedback	

Nonostante i numerosi vantaggi ricavati dall'utilizzo di tali simulatori, non vengono ancora considerati del tutto indispensabili per la formazione pratica dei chirurghi. Ciò si pensa sia dovuto agli elevati costi dei simulatori e alla mancanza di tempo da parte dei tirocinanti impegnati nelle attività di studio teoriche [9].
Tra le atre limitazioni incontriamo la mancanza di realismo, questo perchè non si riesce ancora a pensare come una sorta di videogame possa offrire un valore didattico [2].
Nonostante tutte le problematiche, molti studi di validazione dimostrano che alcuni tra i simulatori virtuali visti fino ad ora, come ad esempio LapSim, siano in grado di far in modo che il chirurgo trasferisca le proprie conoscenze pratiche acquisite con i simulatori, sul campo reale [14].

1.4 Monitoraggio con le tecnologie HMT in ambienti reali

I sistemi HMT sono in grado di registrare le metriche di efficienza delle performance di un chirurgo direttamente in ambienti reali [8].
Tali sistemi, rispetto ai simulatori VR, offrono un'alternativa più economica per valutare le competenze chirurgiche sia in campi di allenamento che in sala operatoria [2].
Non è possibile ricavare le metriche di qualità poichè tale valutazione richiederebbe la presenza di un supervisore oppure, una registrazione dello svolgimento dell'attività permettendone una revisione futura [18].

Tabella 5:Sistemi HMT, Modelli principali, caratteristiche e studi di validazione [2]

HMT SYSTEM	Technological Base	Range of Application	Metrics Registered	Portability	Construct Validity
ICSAD	Electromagnetic	BT, OR*	Hand movements	Yes	(Brydges et al., 2006; Moorthy et al., 2004; Bann et al., 2003; Datta et al., 2001)
ARH	Electromagnetic	BT	Motion	Yes	(Pagador et al., 2012)
BlueDRAGON	Mechanical	BT, OR**	Motion/Force	No	(Rosen et al., 2006; Rosen et al., 2002b)
CELTS	Mechanical	BT, VR	Motion	Yes	(Maithel et al., 2006; Stylopoulos et al., 2004)
ADEPT	Mechanical	BT	Motion	No	(Francis et al., 2002)
Zebris	Ultrasound	BT	Motion	Yes	(Sokollik et al., 2004)
HUESAD	Optical	BT	Motion	-	(Egi et al., 2008)
TrEndo	Optical	BT, VR	Motion	Yes	(van Empel et al., 2012; Hiemstra et al., 2011a; Chmarra et al., 2010b)

Nella Tabella 5 vengono mostrate le principali caratteristiche di tali sistemi, viene specificato il tipo di tecnologia di base che può essere elettromagnetica, meccanica, ultrasonetica oppure di tipo ottico.
Viene definito il range di applicazione che può essere BT (utilizzato in sistemi di allenamento), VR (utilizzato in sala operatoria per allenamento), OR* (utilizzato in sala operatoria su pazienti reali) oppure OR**(utilizzato in sala operatoria su animali).
Come si può notare, solo una minima parte di questi sistemi vengono utilizzati in sala operatoria durante interventi su pazienti reali. Ciò accade perchè adattarli alla sala

operatoria è molto complicato, in quanto richiede l'introduzione di nuovi elementi modificandone così l'ergonomia.
Vengono mostrate inoltre le metriche registrate, tra cui vi troviamo le posizioni X, Y e Z con l'orientazione degli strumenti laparoscopici [2].
Alcuni tra questi sistemi, come l'ICSAD, anzichè seguire i movimenti degli strumenti chirurgici, seguono i movimenti delle mani del chirurgo. Viene inoltre mostrata la possibilità di portabilità o di non portabilità di ciascun sistema, accompagnata dalle risorse bibliografiche che sostengono i rispettivi studi di validazione.

1.5 Sistemi di supporto per la valutazione delle competenze chirurgiche

Grazie ai simulatori VR e ai sistemi HMT è possibile ottenere punteggi sulle performance dei chirurghi che hanno eseguito alcune tra le attività proposte.
Una volta ricavati tali punteggi per avere un riscontro formativo è necessario analizzarli e interpretarli correttamente. Per ottenere questo riscontro si potrebbe sfruttare la figura di un tutor, perdendo però i caratteri oggettivo ed immediato della prova [2].
Un'altra possibilità sarebbe quella di inviare messaggi in cui vengono forniti consigli e punteggi, potendo confrontare i propri risultati con quelli di altri tirocinanti ed esperti [8].

Per poter interpretare ogni punteggio ricavato attraverso una valutazione, bisogna passare tramite due fasi, una di addestramento ed una di classificazione.
Nella fase di addestramento verranno suddivise le caratteristiche più rappresentative dei chirurghi esperti, differenziandole da quelle dei tirocinanti. In questo modo si stabiliranno le diverse classi di uscita che determineranno poi il giudizio finale.
Nella fase di classificazione si deciderà in quali classi di uscita inserire le metriche valutate [2].

Sono stati ricercati alcuni modelli in grado di aiutarci nella valutazione chirurgica, le loro caratteristiche sono riassunte nella Tabella 6.

Tabella 6: Sistemi di valutazione di metriche [8]

	Tasks	***Metrics***	***Acc.(%)***
HMM	*Touch/Coordination/Suture*	*Acceleration/Force/Trajectory /Instrument position/Speed*	*>90*
MM	*Suture/Cholecystectomy*	*Force signatures/ Motion signatures/ Idle state*	*87.5*
BAYES	*Suture*	*Motion signatures*	*>90*
FUZZY	*Suture/Coordination/Peg Grasping/Peg Transfer*	*Time/Errors/Collisions/ Tissue Damage/Continuity of moviments/Speed*	*38*
LDA	*Peg Grasping/Peg Transfer /Bi-manual coordination /Cutting /Dissection/*	*Time/Path length / Depth/ Angular area/ Volume/Motion smoothness*	*74*
SVM	*Peg Grasping/Peg Transfer /Bi-manual coordination*	*Time/Path length/Volume/ Force*	*91.6*

Il modello di Markov nascosto (HMM) è costituito da processi doppiamente stocastici descritti sulla base del comportamento umano, infatti un processo comprende uno stato mentale nascosto e l'altro, lo stato fisico che corrisponde alla manifestazione di quello mentale [2]. Gli HMM valutano le funzioni probabilistiche di forza e movimenti impiegati. I risultati di accuratezza variano dal 90 al 100%.

Il modello di Markov (MM) interpreta sequenze di azioni come una serie di passaggi costituiti da un numero chiuso di stati e da probabilità di passare da uno stato all'altro. Gli stati vengono considerati come gli interventi chirurgici predefiniti, così da costruire diversi modelli per ogni livello di competenza [2]. Tale modello prevede di ottenere un'accuratezza di circa l'87.5% sulla base della probabilità delle azioni calcolate.

Il classificatore Bayes opera una stima sulle probabilità dell'esperienza chirurgica [2], ottenendo un'accuratezza superiore al 90% risultando così affidabile e compatto.

Il modello di Analisi Discriminante Lineare (LDA) assume che i dati ottenuti siano distribuiti in maniera gaussiana. Calcola i centroidi e la matrice di covarianza cumulata al fine di ottenere i coefficienti delle funzioni di classificazione [19].
Si arriva così ad avere un valore massimo di accuratezza pari al 74%.

I modelli *Support Vector Machines* (SVM) sono classificatori molto potenti che riconducono alla classificazione lineare ogni tipo di problema. Avendo a disposizione una serie di dati viene trovata una funzione in grado di classificarli nel miglior modo possibile trovando i

parametri caratteristici delle classi. Qesti modelli ci permettono così di avere un'accuratezza del 91,6%.

In ultimo è presente la logica fuzzy, la quale attribuisce a un certo dato una proprietà tramite un numero compreso tra 0 (proprietà falsa) e 1 (proprietà vera) [20].
Nonostante la potenza di tale classificatore i risultati ottenuti per determinare le competenze chirurgiche non sono stati soddisfacenti [2].

Le difficoltà maggiori non si incontrano nel ricercare il classificatore perfetto che ci dia un valore di accuratezza pari al 100%, cosa molto improbabile, ma si incontrano nel definire le competenze chirurgiche e le condizioni che circondano l'esperimento.
Per competenze chirurgiche si intendono per esempio le metriche di valutazione e i valori legati alle soglie di esperienza, mentre per condizioni che circondano l'esperimento si intendono il numero di partecipanti, la metodologia di validazione, l'architettura del classificatore, le condizioni esterne ecc... [2].

2. Obiettivi

In questo capitolo verranno presentati oltre agli obiettivi generali preposti anche quelli più specifici pianificati durante lo svolgimento del lavoro.

2.1 Obiettivi Generali

2.1.1 Obiettivo del Tracking

Il principale obiettivo di questa tesi è la migrazione di un codice scritto in linguaggio C++ al linguaggio Java al fine di poterlo implementare in Android Studio.
Si intende sviluppare così un applicazione per telefoni e tablet in grado di valutare le abilità motorie di neo-chirurghi per quanto riguarda la chirurgia mininvasiva.
Ciò è reso possibile grazie all'aiuto di algoritmi che ci permettono di effettuare un analisi video endoscopica in grado di monitorare i movimenti degli strumenti laparoscopici e poterli usare come principale fonte di informazione per poter eseguirne la valutazione.

2.1.2 Obiettivo calibrazione camera

Un altro obiettivo è stato quello di creare un applicazione, sempre sulla base di un codice già esistente in C++, in grado di calibrare la camera di un dispositivo tablet o smartphone, sulla base del colore dei marcatori utilizzati.
Quest'applicazione inoltre, viene creata in modo che si possano calibrare i colori dei marcatori indipendentemente dalla luce esterna e salvarne successivamente i relativi parametri di calibrazione.
L'obiettivo globale che ha finalizzato questo lavoro è sicuramente l'utilizzo di questa nuova tecnologia per fornire un'assistenza sanitaria migliore.

2.2 Obiettivi Specifici

- Ricerca sullo stato dell'arte, in particolar modo sul sistema di monitoraggio EVA;
- Acquisire familiarizzazione con l'ambiente di Android Studio e il linguaggio di programmazione Java;
- Sviluppo del design di entrambe le applicazioni;
- Migrazione dei codici;
- Adattare lo smartphone al simulatore in modo che si mantenga fermo e non entri una quantità di luce tale da interferire con la calibrazione;
- Effettuare prove per verificare la correttezza e la precisione dei risultati dell'applicazione;

3. Materiali e Metodi

In questo capitolo verrà fatta una panoramica su tutti i materiali utilizzati per portare a termine gli obiettivi preposti.
Verranno presentati il software Android Studio con le sue caratteristiche, il linguaggio Java, i due codici in C++ da migrare in Android Studio ed infine i dispositivi mobili utilizzati insieme al simulatore e agli strumenti laparoscopici.
È bene ricordare che per motivi di riservatezza legati all'Università Politecnica di Madrid, in cui è stato svolto l'intero progetto, non è possibile mostrare nè i codici nè parte di questi.
In questo capitolo verranno inoltre presentati i metodi utilizzati che ci hanno permesso di arrivare ad ottenere l'applicazione.

3.1 Materiali

3.1.1 Android Studio

Figura 3:Logo Android Studio

Android Studio è un sistema operativo che contiene librerie e strumenti utili alla creazione di applicazioni mobili che possano rispondere all'utente in maniera immediata [21].
Per applicazione mobile s'intende quel software che può essere installato su uno smartphone in modo da fornire ulteriori funzionalità che non vengono rese dal sistema operativo stesso.
Con Android studio si possono sviluppare applicazioni Android. Tali applicazioni possono essere scaricate solo da dispositivi costituiti da un sistema operativo di tipo Android.
È possibile fare il dowload di Android Studio sia su Mac, che su OS X, LINUX e Windows.
Diventando l'IDE primario di Google per la creazione di applicazioni Android Studio, sostituisce Eclipse.
Esso utilizza la macchina virtuale Dalvik Virtual Machine (DVM) in grado di eseguire il codice contenuto all'interno di un file di estensione .dex ottenuto a partire dal byte-code Java, poichè le applicazioni vengono scritte in linguaggio di programmazione Java [22].

Rispetto ad altri sistemi operativi per dispositivi mobili, Android è un sistema *open source* perchè:

- utilizza il *Kernel Linux*[2] che anch'esso è *open source* ed inoltre sfrutta le stesse API e le stesse librerie che vengono usate per la realizzazione di applicazioni mobili
- il codice sorgente può essere consultato e migliorato

[2] È un sistema operativo

- con la licenza *Open Source Apache* 2.0 di Android i produttori possono realizzare le proprie estensioni senza dover pagare *royality* per poter utilizzare Android sui loro dispositivi [23].

Negli ultimi anni Android ha superato tutti gli altri sistemi operativi diventando il più diffuso.

La versione di Android Studio utilizzata per portare a termine il progetto di tesi, è stata la versione 3.0.1.

3.1.1.1 Storia di Android Studio

Il 5 novembre 2007 nasce Android grazie alla *Open Handset Alliance*, si tratta di un consorzio di 84 aziende con a capo Google. Poco dopo vengono rilasciate le librerie, la documentazione, i tutorial, gli esempi di progetto, un emulatore del dispositivo e gli strumenti di sviluppo. Tutti questi sistemi sono inclusi nel paccheto *Software Development Kit* (SDK) [24].

Nel 2008 iniziarono ad esserci aggiornamenti per migliorare le prestazioni del sistema operativo e cercare di evitare alcuni problemi legati alla sicurezza. Tra le versioni più importanti del 2009 ci sono la 1.5 *CuPCake* che introduce una maggior integrazione con i servizi Google e aggiunge supporto ai *widget*[3] [25] ed infine, la 1.6 *Donut* che integra funzionalità a voce come la ricerca e la sintesi vocale [26].

Escono poi le versioni 2.3 *Gingerbread*, la 3.0 *Honeycomb*, la 4.0 *Ice Cream Sandwich* e la 4.1 con il nome di *Jelly Bean.*

Figura 4:Logo Android Studio versione 4.4

Come si può notare ad ogni versione di Android è stato dato un nome di un dessert questo perchè, come dichiarato da *Google* durante l'annuncio della versione 4.4 *KitKat*, gli smartphone ci rendono la vita più dolce [27].

Successivamente viene introdotta la versione 5.0 di nome *Lollipop* che, come miglioramenti, include notifiche a cui è possibile accedervi direttamente dalla schermata di blocco [28].

La versione successiva, la 6.0 chiamata *Marshmallow*, contiene nuove API per gli assistenti contestuali, un nuovo sistema di gestione della batteria, il riconoscimento delle impronte digitali e connettori USB *Type*-C, con la possibilità di poter migrare dati e app su una microSD [29].

La penultima versione, 7.0 *Nougat*, apporta consistenti modiche nel sistema operativo: si possono aprire più applicazioni nello stesso momento in una visualizzazione a schermo diviso e vi è una modifica per quanto riguarda il risparmio energetico che se attivato limita le funzionalità del dispositivo. Inoltre la piattaforma è passata a un ambiente Java basato su *OpenJDK* [30].

L'ultima versione del 21 agosto 2017 è la 8.0 e si chiama Oreo. Vengono migliorate ancora una volta le prestazioni con l'ottimizzazione dell'utilizzo della batteria, l'inserimento di

[3] È un componente grafico di un'interfaccia utente

Bluethoot 5, il raggruppamento notifiche, l'integrazione a livello di sistema con app VoIP e Wi-Fi consapevole. Android Oreo introduce anche due principali funzionalità della piattaforma, tra cui la distribuzione del software Android Go-a del sistema operativo per dispositivi di fascia bassa, nonchè il supporto per l'implementazione di un livello di astrazione hardware [31].

3.1.1.2 Strumenti essenziali di Android Studio

Per poter progammare e sviluppare app con Android Studio è necessario scaricare un JDK, ovvero un kit utile per poter programmare con Java, ed un IDE che consiste in un ambiente di sviluppo integrato contente tutti gli strumenti utili al programmatore (nel nostro caso si parla di Android Studio stesso) [32].

Un'altro elemento fondamentale è l'Android SDK cioè un pacchetto di strumenti costituito da programmi, emulatori e piattaforme per ogni versione di Android includendo anche molto altro in grado di essere cambiato in base alle necessità del programmatore.
Inoltre, un elemento utile è l'interfaccia JNI che consente di utilizzare le librerie C all'interno del codice sorgente in linguaggio Java e di conseguenza poter utilizzare servizi nativi del sistema operativo. Questo è vantaggioso in quanto il linguaggio Java pur essendo il linguaggio più utilizzato è comunque un linguaggio di basso livello se confrontato con il C e il C++ [32].

Troviamo infine il Native Development Kit (NDK) che è un set di strumenti che permette agli sviluppatori di poter implementare parti di codice di un applicazione utilizzando linguaggi nativi di Android, C, C++, le librerie libc (C library), libm (math library), libz (Zlib compression library-8), liblog e OpenCV [32]. Ciò permette di bypassare la macchina virtuale Dalvik[4] e accedere così direttamente al codice macchina sottostante, evitando tutti i controlli e le mediazioni della macchina virtuale di Android, potendo sfruttare più a fondo la potenza della CPU.
Questo pacchetto viene principalmente utilizzato quando si vuole sviluppare un'applicazione che usi al massimo la potenza della CPU e faccia un uso moderato della Ram [32].

3.1.1.3 Architettura di Android Studio

Come già accennato Android Studio comprende, un sistema operativo, un ambiente di esecuzione, librerie, *middleware*, servizi e applicazioni. L'architettura di Android Studio è possibile visualizzarla tramite la Figura 4.

[4] È una macchina virtuale ottimizzata per sfruttare la poca memoria presente nei dispositivi mobili e permette di eseguire più processi contemporaneamente [22]

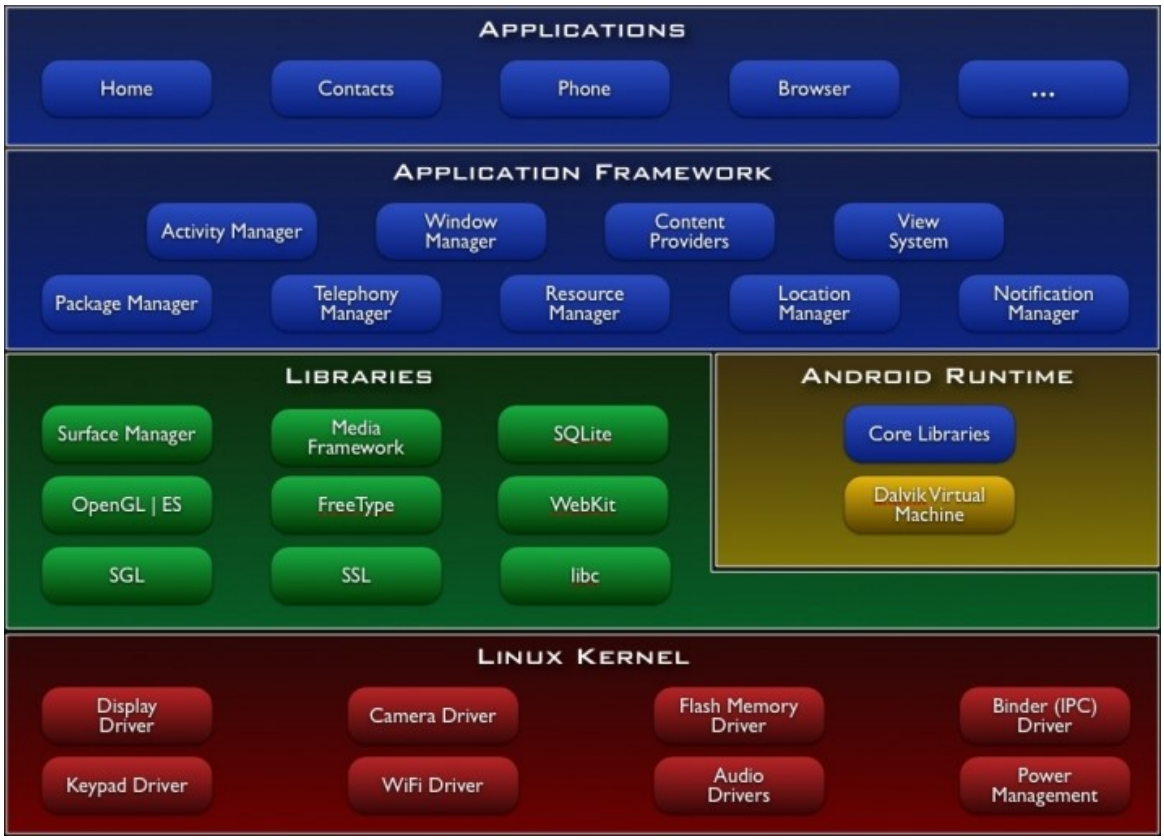

Figura 4: Architettura Android Studio [33]

Partendo dall'ultimo *layer* è possibile notare il *Linux Kernel* che ha il compito di fornire driver come il display, audio, fotocamera, tastiera e molti altri componenti fondamentali di un dispositivo per permettere il corretto funzionamento delle applicazioni [33].
Salendo di un *layer* vi troviamo l'Android *run-time* che traduce l'applicazione Android dal formato *bytecode* in un formato *Executable* and *Linkable* Format (ELF) tramite un processo denominato *Ahead-of-Time* (AOT). Questo nuovo formato ci permette di aumentare la velocità dell'applicazione e la durata della batteria del dispositivo [34].
Nello stesso *layer* possiamo incontrare l'insieme delle librerie Java specifiche per Android, ognuna delle quali consente una particolare funzione all'applicazione che viene creata.
Salendo ancora vediamo l'*Application Framework* che è costituito da un insieme di API[5] e da componenti che utilizzano le librerie native per l'esecuzione di funzionalità di base del sistema Android.
In cima vi è il *layer* delle Applicazioni che comprende sia le applicazioni native della versione di Android in uso sul dispositivo, sia le applicazioni di terze parti installate dall'utente sul dispositivo.

[5] Significa "interfacce di programmazione di un'applicazione". Indica ogni insieme di procedure disponibili al programmatore. Spesso, con tale termine si intendono le librerie software disponibili in un certo linguaggio di programmazione [22].

3.1.1.4 Componenti di un'applicazione

I blocchi costitutivi di un'applicazione Android sono cinque [35]:

- L'*Activity* è un interfaccia che permette l'interazione diretta tra gli utenti e l'applicazione stessa. Tra le *activity* di un applicazione troviamo la principale definita "*main*" che è quella che viene visualizzata per prima all'avvio dell'applicazione. Ogni *activity* può iniziarne una nuova per eseguire diverse azioni. Per poter creare e comprendere le *activity* è utile conoscerne il loro ciclo di vita.

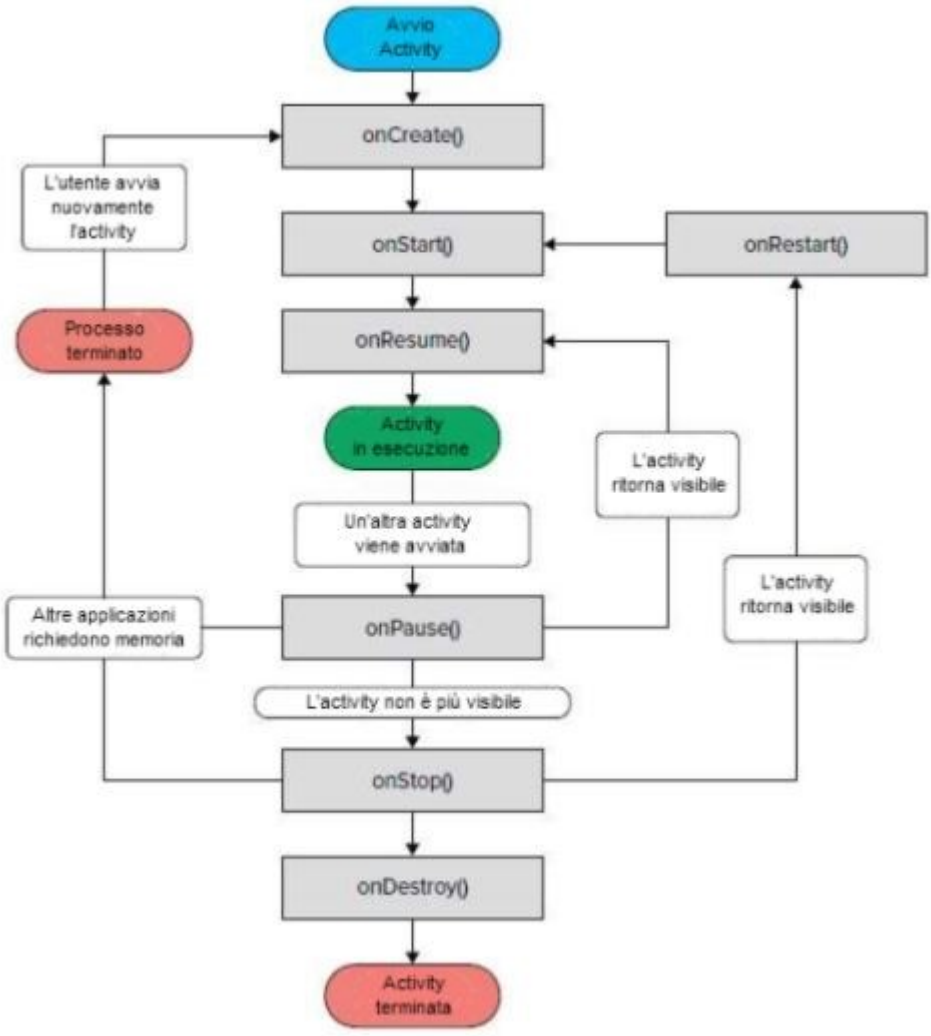

Figura 5:Ciclo di vita di un Activity

Quando un *activity* viene chiamata in esecuzione il sistema crea un'istanza della corrispondente classe invocando poi tre metodi, che sono :

1. *OnCreate* : viene chiamato nel momento in cui l'*activity* viene creata e permette di definire tutte quelle operazioni che devono essere eseguite una sola volta, come ad esempio il *layout* [33]
2. *OnStart* : viene chiamato quando l'*activity* sta per essere visualizzata e qui vengono implementate tutte le funzionalità che sono legate alla visualizzazione [33]

3. *OnResume* : viene chiamato quando l'*activity* viene visualizzata sullo schermo e perciò è pronta per essere utilizzata [33]

Anche quando viene cambiata *activity* si passa attraverso tre differenti step, che sono :

1. *OnPause* : viene chiamato per annunciare l'interruzione dell'*activity* e può essere seguito da *OnResume*() nel caso in cui l'*activity* ritorni a funzionare o da *OnStop*() in caso contrario [33]
2. *OnStop* : viene chiamato per segnalare che l'*activity* non è più visibile sullo schermo, può essere seguito da *onRestart*() se l'*activity* viene richiamata e quindi rivisualizzata o da *onDestroy*() nel caso in cui venga distrutta [33]
3. *OnDestroy* : viene chiamato prima che l'*activity* venga distrutta [33]

La struttura secondo cui è organizzato l'insieme delle *activity* è una struttura a pila dove in cima vi è quella attiva, ovvero l'*activity* attualmente in uso [33].

Un'*activity* deve permettere lo scorrimento della schermata, interazioni con i bottoni, la visualizzazione di video e animazioni ecc. Posso definirne il comportamento nel file .Java inserendo e modificando i metodi all'interno della classe *MainActivity*.

- L'*Intent* è un insieme di messaggi contenente informazioni astratte sull'operazione da eseguire e sui dati sui quali compiere azioni. Gli *intent* possono essere impliciti o espliciti. Sono impliciti quando il nome del componente obiettivo non si conosce e perciò vengono usati per attivare componenti di altre applicazioni, sono espliciti invece quando il componente obiettivo ha un nome conosciuto e sono usati per messaggi interni [33].
- Il *Broadcast Receiver* sono un insieme di messaggi che avvisano l'utente quando vi si presenta un evento esterno improvviso come per esempio una chiamata in arrivo, una connessione disponibile o non, l'arrivo di un messaggio ecc [33].
- I *Service* sono metodi che preparano i messaggi che vengono mostrati all'utente dalle activity [33].
- Il *Content Provider* è il mezzo che si occupa della gestione dei dati e ne permette la condivisione tra le applicazioni [33].

Per quanto riguarda invece le cartelle che si formano una volta creato un nuovo progetto Android (che sia esso *wearable* o *mobile*), troviamo :

- *MANIFEST* contiene il file AndroidManifest.xml il quale a sua volta contiene tutte le informazioni per poter eseguire il codice. Si occupa di dichiarare i permessi, di nominare le varie classi, di descrivere i componenti dell'applicazione, di dichiarare il livello minimo di API Android e infine vi è contenuta una lista delle librerie necessarie. I permessi possono essere additati come "Normali" o "Pericolosi". Quelli normali non mettono a rischio la riservatezza dell'utente, per esempio, possono

essere: l'accesso alla rete wifi e la connettività del bluethoot. Quelli pericolosi invece, si chiamano così perchè mettono in pericolo la riservatezza dell'utente. Questi riguardano: la connessione ai dati GPS, l'acceso a file, l'accesso ai contatti in rubrica e alle chiamate sia effettuate che ricevute [36].

- *RES* è costituito da tutti i file del progetto come i colori, le immagini, i file.xml, il menu,il layout ecc... [36].
- JAVA contiene tutto il codice dell'applicazione [36].
- *GRADLE* contiente un insieme di impostazioni per la configurazione che viene applicata ad ogni progetto creato in Android Studio. Consente inoltre, di poter fare il *build* anche su macchine senza Android Studio [36].

3.1.2 Libreria OpenCV

Figura 6: Logo Libreria OpenCV

OpenCV è una libreria *open-source* scritta in liguaggio C e C++ che supporta i sistemi operativi Linux, Windows e Mac OS X [37].
È dedicata alla *computer vision*, ovvero trattare le immagini o i video bidimensionali in modo da estrarne i dati rilevanti per ottenere una rappresentazione tridimensionale.
Tale libreria aiuta gli sviluppatori di applicazioni in quanto mette a disposizione più di 500 funzioni che possono essere usate per il trattamento di immagini, insieme a un pacchetto di algebra matriciale e funzioni matematiche ottimizzate [37].
Queste funzioni coprono molte aeree, come la robotica, sicurezza, immagini mediche, calibrazione camera. Vi è compresa inoltre, un'intera libreria per l'apprendimento automatico [37].
OpenCV è costituito da :

- *Core*: è formato dalle strutture dati principali come la gestione degli errori e la matematica di base;
- Improc: è un modulo usato per processare le immagini;
- Video: è un modulo per processare i video e contiene algoritmi per il tracciamento di oggetti;
- *highGUI*: è un'interfaccia che permette all'utente di gestire le riprese video
- *Machine Learning*: contiene funzioni di clustering, classificazione e data analisi; [37]

Tra tutti gli algoritmi presenti all'interno di queste librerie che si occupano di trattamento di immagini troviamo l'algoritmo di Canny, utilizzato in questo progetto di tesi, che ci permette di identificare e delimitare il contorno di oggetti per poterli riconoscere e localizzare [38].
Open CV comprende più versioni ed ogni versione include funzioni differenti. La versione utilizzata come principale fonte di librerie nel nostro caso specifico è stata la versione 3.2.

3.1.3 Java

Figura 7:Logo Java

Java è un linguaggio di programmazione che nasce all'inizio degli anni novanta grazie all'azienda SUN *Microsystems.* Viene realizzato da un gruppo di ricercatori dell'Università di Stanford con a capo il Professore James Gosling. Nel 1992, Java, viene presentato con il nome di *Oak* (in italiano quercia) ma ben presto a causa di problemi di *copyright* gli viene cambiato il nome [39].

Java è un linguaggio di programmazione orientato agli oggetti che raggruppa in classi i metodi e le strutture dati in modo da creare così un oggetto.

Un oggetto è costituito da uno stato, un comportamento e un'identità. Una classe invece è un insieme di più oggetti con caratteristiche in comune.

Java è correlato ai linguaggi di programmazione C e C++ [40].

James Gosling et alt. Mettono a disposizione il libro chiamato "*Java Language Specification & Virtual Machine*" [40] per aiutare le persone che vogliono imparare questo linguaggio.

Oracle Academy[6] inoltre, fornisce pacchetti contenenti corsi di formazione per docenti, tutorial, esempi di applicazioni, supporto e risorse di certificazione per le scuole.

Nel 1993 con l'esplosione di internet, Java si fece notare diventando così lo strumento per poter programmare.

Dal 2014 risulta che Java è tra i linguaggi di programmazione più usati al mondo, con una stima di nove milioni di sviluppatori. Infatti, lo ritroviamo nella maggior parte dei computer aziendali, dei telefoni cellulari, dei dispositivi TV, ed è senza dubbio la piattaforma di sviluppo preferita dagli sviluppatori. Java infatti, consente agli sviluppatori di applicazioni di "scrivere una volta ed eseguire ovunque" [41].

Questo perchè una volta che il codice viene compilato può essere eseguito su tutte le piattaforme che supportano Java, in quanto il prodotto compilato è definito in un formato *bytecode*.

Inoltre, Java ci permette di poter eseguire i programmi su *broswer* Web e poter accedere ai servizi Web disponibili, di creare app efficienti e robuste per qualsiasi dispositivo elettronico, per negozi, sondaggi, elaborazione di *form* HTML, forum di siti Web e molto altro. Si possono anche combinare applicazioni tra loro per avere servizi personalizzati [40].

Esistono più versioni di Java che vengono aggiornate di volta in volta migliorando la sicurezza delle applicazioni, le loro prestazioni e la loro stabilità. Attualmente la più recente è la versione 8.0.

[6] È una tra le più grandi società al mondo di software per le imprese

3.1.4 Codice calibrazione camera

Tra i codici che ci sono stati consegnati, al fine di migrarli al linguaggio di programmazione Java su Android Studio, vi è presente quello che permette di calibrare la camera del dispositivo sulla base dei marcatori utilizzati.

È strutturato in tre fasi:

- la prima fase corrisponde alla calibrazione del primo marcatore scelto a piacere tra i due a disposizione. Inizialmente, vengono visualizzate le istruzioni che chiariscono all'utente come fare. L'utente dovrà infatti cliccare con il mouse sul marcatore da calibrare in modo da salvarne i relativi valori *Hue Saturation Value* (HSV). Verranno mostrate in seguito, tre barre corrispondenti rispettivamente ai valori HSV da aggiustare in modo che il marcatore possa essere riconosciuto al meglio;
- la seconda fase corrisponde alla fase della calibrazione del secondo marcatore e consiste nell'eseguire il medesimo procedimento del marcatore precedente;
- la terza ed ultima fase consiste nella fase chiamata *shaft*. A questo punto l'utente dovrà modificare l'immagine in modo che il contrasto tra i due marcatori e lo sfondo risulti il maggiore possibile, al fine di poterli distinguere al meglio.

Successivamente vengono acquisiti e salvati automaticamente i dati ottenuti in un file.xml al fine di utilizzarli nel codice del *tracking,* il quale a sua volta, li userà per il monitoraggio degli strumenti laparoscopici.

Figura 8:Workflow app Calibrazione

3.1.5 Codice *tracking* C++

L'altro codice da migrare a Java relativo al monitoraggio degli strumenti laparoscopici, si articola in due parti quella del *Main* e quella delle *Processing Functions*.
Queste due parti collaborano tra di loro poichè nel main vengono richiamate alcune funzioni implemente nella seconda parte del codice.
È mostrato in Figura 9 il *workflow* con i principali blocchi logici che costituiscono il codice *main* del *tracking*.

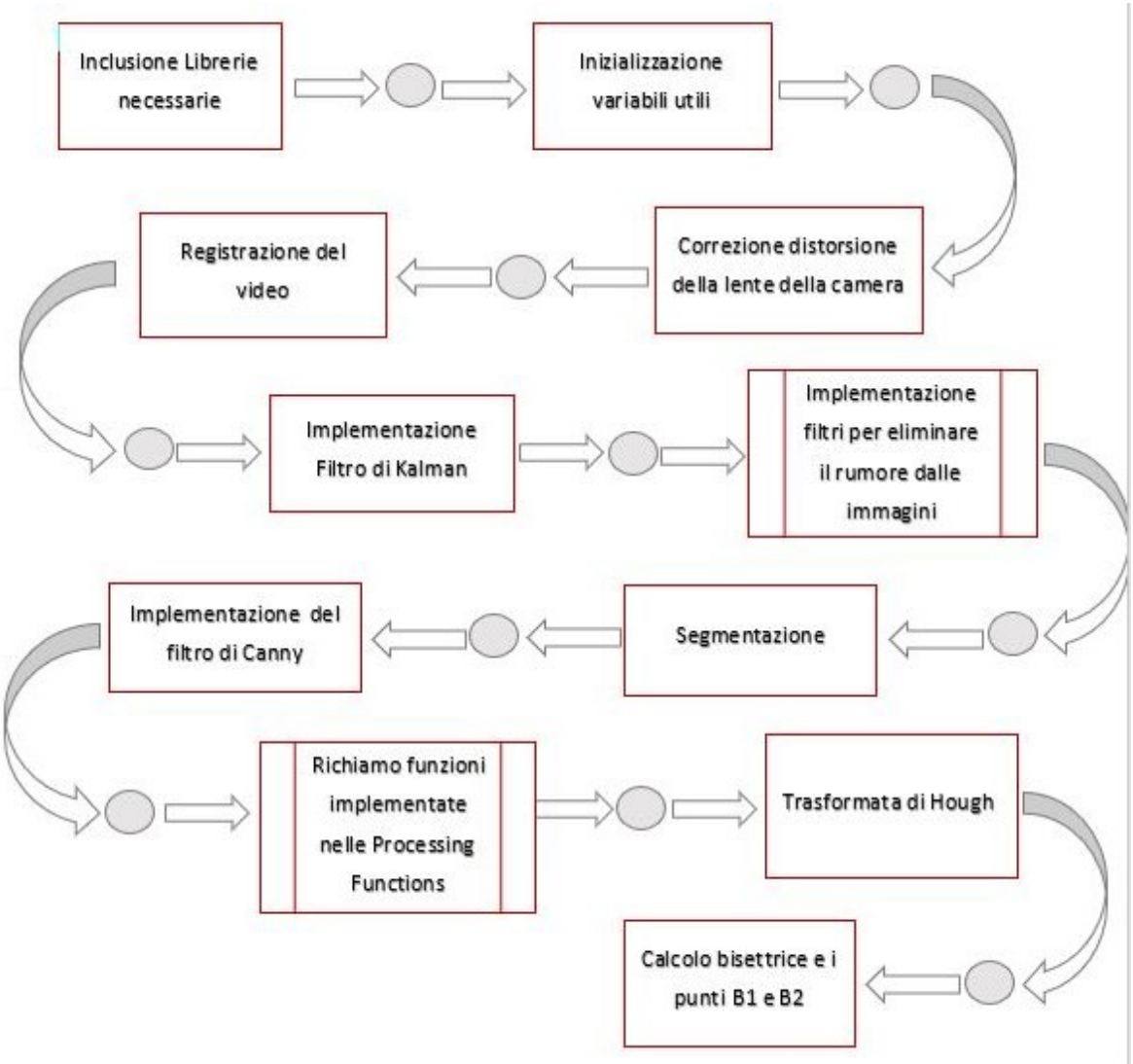

Figura 9:Workflow app Tracking

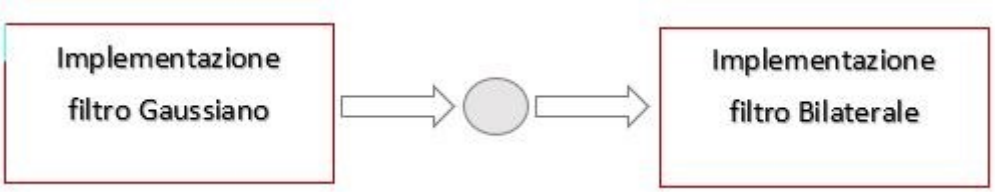

Figura 10:Workflow implementazione filtri per eliminare il rumore dalle immagini

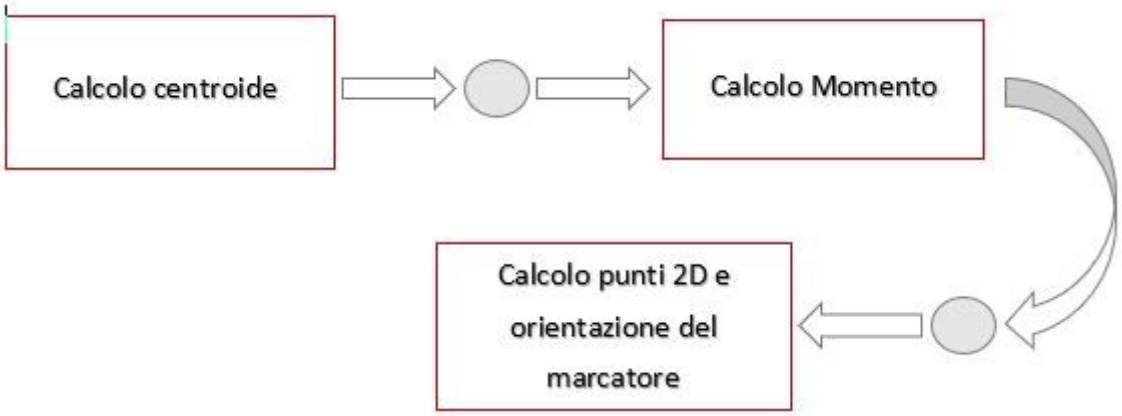

Figura 11:Workflow richiamo funzioni implementate nelle Processing Functions

- INCLUSIONE LIBRERIE NECESSARIE: Su entrambi i codici vengono incluse tutte le librerie necessarie compresa la libreria OpenCV, fonte essenziale del lavoro.
  Vi troviamo per esempio inserite la libreria #include <math.h>, che contiene le funzioni usate per le operazioni matematiche, e anche #include <opencv2/core/core.hpp> che ci permette di utilizzare determinate funzioni presenti nella libreria OpenCV;
- INIZIALIZZAZIONE VARIABILI: Sono state inizializzate e dichiarate tutte le variabili necessarie a permettere il funzionamento del codice. Tra le tante, vi troviamo quelle destinate a contenere le coordinate delle posizioni 3D degli strumenti laparoscopici;
- CORREZIONE DISTORSIONE DELLA LENTE DELLA CAMERA: Prima di iniziare il monitoraggio dei due strumenti laparoscopici è stata corretta la distorsione della lente della camera. Ciò è stato fatto perchè i sistemi di telecamere tendono a soffrire di un fenomeno noto come distorsione radiale o barile, a causa del quale le immagini vengono deformate dal centro ottico [2].
  Dato che il nostro sistema di tracciamento deve rilevare i contorni dritti dello strumento, è stata applicata una funzione di correzione in grado di garantirne il riconoscimento. La funzione utilizzata, direttamente dalla libreria OpenCV, è stata *Undistort Rectify Map*, la quale deforma la grigla dei pixel per poi mapparla sull'immagine di destinazione. Con il termine mappatura s'intende che per ogni pixel dell'immagine di destinazione, la funzione calcola le coordinate del corrispondente pixel dell'immagine sorgente e le copia nell'immagine finale [42];

- REGISTRAZIONE DEL VIDEO: In questo caso, differentemente da come è stato fatto per i dispositivi mobili, il filmato può essere ottenuto,oltre che in *real-time* anche tramite caricamento di un video già esistente. Pertanto grazie a tale opzione è possibile oltre a registrare video, anche effettuare il monitoraggio su un video già esistente;
- IMPLEMENTAZIONE FILTRO DI KALMAN: Viene in seguito applicato il filtro di Kalman, che ci permette di eliminare il rumore di cui è soggetto uno stato dinamico al fine di costruirne una stima che sia il più possibile simile a quella reale [43];
- IMPLEMENTAZIONE FILTRI PER ELIMINARE IL RUMORE DALLE IMMAGINI: Si può così iniziare a lavorare sui due strumenti laparoscopici.
  Vengono inizialmente applicati ad ogni frame il filtro Gaussiano e quello bilaterale, al fine di eliminare i rumori provenienti dalle immagini.
  Il filtro Gaussiano consiste nell'eseguire la convoluzione dell'immagine con una curva di Gauss centrata e normalizzata. Il filtro bilaterale è un filtro digitale[7] che preserva i contorni delle immagini, mantenendoli nitidi [44];
- SEGMENTAZIONE: Una volta ripulite le immagini, si effettua una segmentazione per poter classificare e raggruppare i pixel dell'immagine con caratteristiche in comune, come ad esempio l'intensità e il colore;
- IMPLEMENTAZIONE DEL FILTRO DI CANNY: Viene applicato il filtro di Canny che dall'immagine segmentata ci permette di riconoscere e salvare i contorni dei due strumenti laparoscopici.
  L'algoritmo è stato implementato per ricercare all'interno dell'immagine punti in cui la luminosità cambia improvvisamente. Tali punti sono considerati rumore. La funzione matematica che viene utilizzata è la derivata prima di una funzione gaussiana [45].
  Più nel dettaglio l'algoritmo presentato da Canny comprende quattro fasi:
  I. Si applica all'immagine un filtro gaussiano per eliminare gli effetti del rumore;
  II. Si calcola la derivata prima in funzione della luminosità dell'immagine;
  III. I punti corrispondenti al massimo della derivata prima sono considerati come punti di contorno, gli altri vengono eliminati;
  IV. Si applica la sogliatura adattiva tramite isteresi, con la quale si confrontano i punti restanti con due valori soglia mantenendo tutti i punti che hanno un valore maggiore del valore soglia più alto [45];
- RICHIAMO FUNZIONI IMPLEMENTATE NELLE PROCESSING FUNCTIONS: A questo punto inizia il vero e proprio processo del primo marcatore. Vengono richiamate alcune funzioni implementate nel codice delle *Processing Functions* che

[7] Opera su segnali discreti nel tempo

permettono di calcolare il centroide, il momento, i punti 2D e la pendenza dello strumento laparoscopico.
Viene inoltre richiamata una funzione che agisce nel caso in cui i due marcatori si sovrappongano oppure quando uno dei due esce dal campo di visualizzazione della camera, perciò o uno o entrambi non vengono riconosciuti in maniera adeguata. Tale funzione infatti si occupa di assumere come valori delle posizioni quelli dell'istante precedente;

- TRASFORMATA DI HOUGH: Successivamente si applica la trasformata di Hough, la quale si occupa del riconoscimento di oggetti all'interno di un'immagine [46]. Permette di stabilire se i pixel, ricavati con l'algoritmo di Canny, si trovano su una curva che costituisce i bordi dell'oggetto da riconoscere [46].
  Prima di poter essere utilizzata, la trasformata di Hough, necessita di un *preprocessing* con il quale si ricavano i contorni dell'oggetto da ricercare [46].
  In questo codice la fase di *preprocessing* è fornita dal filtro di Canny.
  In particolare, in questo codice, la trasformata di Hough, sulla base del coefficiente angolare calcolato precedentemente [46], ha il compito di identificare quali tra questi bordi rappresenta una linea retta.
  Per una miglior caratterizzazione dei bordi viene eseguito anche un miglioramento del contrasto dell'immagine.
  Tali bordi vengono poi marcati con delle linee colorate da poterli identificare meglio, com'è possibili visualizzare in Figura 45;
- CALCOLO BISETTRICE E PUNTI CHIAMATI B1 E B2: Per ottenere le posizioni 3D dellla punta dello strumento laparoscopico, vengono calcolate la bisettrice e i punti ricavati dall'intersezione tra il bordo dello strumento e la perpendicolare che passa per il suo centroide;
- Il medesimo procedimento viene eseguito per il secondo marcatore, a partire da quando vengono richiamate alcune funzioni (implementate nel codice delle Processing Functions) che permettono di calcolare il centroide, il momento, i punti 2D e la pendenza dello strumento laparoscopico;
- Vengono infine chiusi tutti i file rimasti aperti.

3.1.6 Simulatore e strumenti laparoscopici

Come già accennato nel capitolo Introduzione, il centro di chirurgia mininvasiva *Jesùs usòn* di Madrid ci ha fornito il simulatore che corrisponde al modello SIMULAP-R, insieme ad una serie di strumenti laparoscopici di prova.
Tale simulatore ha le dimensioni della cavità addominale di un paziente adulto ed è costituito da fori attraverso i quali è possibile inserire gli strumenti laparoscopici tramite *trocar*. Il simulatore può essere visualizzato in Figura 12.

Figura 12:Simulatore ambiente chirurgico (box trainer)

Gli strumenti laparoscopici hanno una forma allungata con un diametro di circa 5 mm, collegata a un'estremità con pinze metalliche usate per afferrare, dissezionare o eseguire azioni [47]. Nell'altra estremità vi è una maniglia che consente di eseguire e controllare il movimento delle pinze.

I movimenti possibili consistono nella rotazione sinistra e destra, rotazione su e giù, rotazione oraria e antioraria e inserimento e ritiro [47].

Il loro utilizzo prevede l'inserimento all'interno del corpo del paziente tramite piccole incisioni.

Alla base di questi strumenti laparoscopici sono stati solo posti dei marcatori realizzati con tubi termorestringenti di diversi colori, in modo da riconoscerli. I colori utilizzati in laboratorio durante le prove sono stati rosso, verde e blu.

Figura 13:Strumento laparoscopico

Figura 14:Strumenti laparoscopici con marcatori

3.1.7 Struttura di legno per attività all'interno del simulatore

All'interno del campo di visualizzazione verranno inserite delle strutture di legno contenenti alcune attività che devono essere svolte dai neo-chirurghi affinchè siano valutate al meglio le loro conoscenze pratiche.
A seconda degli obiettivi didattici variano sia le attività proposte che i materiali di cui esse sono costituite.
La struttura di cui si disponeva in laboratorio corrispondeva alla struttura *Coordinate Pulling* mostrata in Figura 15 ed in Figura 16.

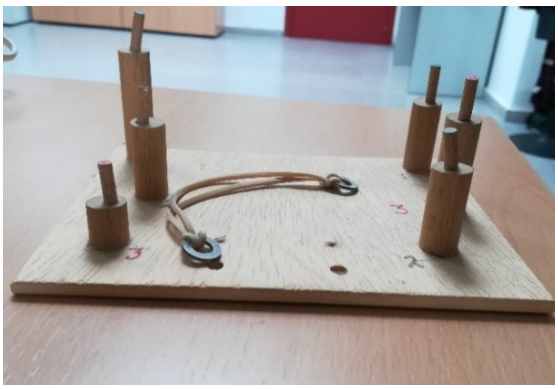

Figura 15:Coordinate Pulling vista laterale

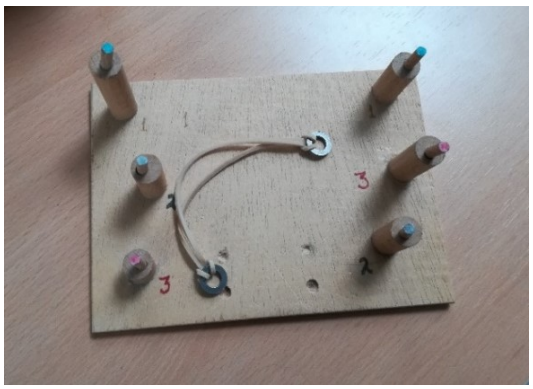

Figura 16:Coordinate Pulling vista dall'alto

Tale struttura è costituita da torri con differenti altezze ed inclinazioni.
L'obiettivo è posizionare sulla cima di queste torri, tre bande elastiche collegando le torri dello stesso numero tra loro.
L'intero compito può essere svolto potendo usufruire di entrambe le mani.

È possibile valutare in questo modo la coordinazione occhio-mano, la coordinazione bimanuale, l'afferrare e il trasferire degli oggetti [2].

3.1.8 Tablet e smartphone

Si è scelto di creare quest'applicazione appositamente per smartphone e tablet in quanto sono più accessibili, più comodi e più facili da usare da parte dei chirurghi.
Per consentire il corretto funzionamento dell'applicazione è necessario disporre di un dispositivo con un software di tipo Android con una API compresa tra 17 e 25, che corrisponde ad un sistema operativo Android compreso tra Android 4.2 ed Android 7.1.1.
Gli strumenti a disposizione in laboratorio per poter eseguire le prove sono stati un tablet Samsung S2 con un sistema operativo Android 5.0.2 *Lollipop* e uno smartphone Samsung S6 con Android 6.0.1 *Marshmallow*.
Non è necessario che il dispositivo di interesse disponga di una connessione ad internet, del bluethoot, del GPS e neanche del microfono. L'unica cosa essenziale è che sia provvisto di una fotocamera.
Inoltre, il dispositivo per essere utilizzato, deve contenere uno spazio di memoria tale da potervi salvare i dati relativi alle posizioni 3D risultanti.
Come da previsione l'applicazione ha funzionato sullo smartphone e sul tablet allo stesso modo, riportando gli stessi risultati e gli stessi errori durante le prove.
Si è preferito continuare ad usare solo lo smartphone per una questione di comodità, in quanto era molto più semplice sistemarlo all'interno della cassa con l'aiuto di un solo elastico.

3.2 Metodi

- Sviluppo applicazione della calibrazione della camera;
- Implementazione del codice del Tracking e creazione applicazione;
- Cambiamenti effettuati;
- Prove sperimentali;

3.2.1 Sviluppo applicazione della calibrazione della camera

Per raggiungere gli obiettivi preposti è stato necessario circa un mese di preparazione nell'*Universidad Politécnica de Madrid Escuela Técnica Superior de Ingenieros de Telecomunicación* (ETSIT), nella quale è stata presa confidenza, oltre che con l'argomento in generale, anche con il linguaggio di programmazione Java e con il software Android Studio.

Si è iniziato sviluppando piccole applicazioni Android di prova, la maggior parte delle quali erano incentrate sulla fotocamera e filtri in grado di modificare i colori, l'intensità, il contrasto ecc.
È stato preso di esempio il libro "*Android Application Programming with OpenCV* 3" di Joseph Howse [38], perchè si riuscisse a capire in maniera più approfondita come poter implementare un applicazione in Android Studio utilizzando la libreria OpenCV.
Si è passati a comprendere l'intero codice scritto in C++, studiandone il contenuto linea per linea. Esso conteneva la parte della calibrazione della camera e quella del monitoraggio degli strumenti laparoscopici.

Prima di migrare l'intero codice della calibrazione al linguaggio Java, sono state eseguite alcune prove di calibrazione con il software Matlab tramite l'app *Camera Calibrator* che mette a disposizione. Vi sono state caricate delle immagini precedentemente fatte ai marcatori in modo da ottenerne i rispettivi range dei valori *Red Green Blue* (RGB), al fine di inserirli direttamente nel codice Java del *tracking*. Le prove fatte sono risultate insufficienti in quanto:

- Si ottenevano i valori RGB delle immagini che non descrivevano in maniera esaustiva le caratteristiche dei due marcatori;
- Il codice del *tracking* richiedeva i valori *Hue Saturation Value* (HSV) e cambiarli direttamente all'interno del codice risultava troppo complicato;
- Ogni volta che si decideva di cambiare i marcatori o il dispositivo di prova era necessario rieseguire la calibrazione e ciò comportava un lungo processo ad opera di un esperto;
- Le immagini dalle quali venivano ricavati i range dei valori RGB su Matlab erano ottenute tramite schreenshot dal telefono. Ciò comportava una piccola distorsione dei colori, che non risultavano del tutto simili alla realtà, aumentando la difficoltà nel tracciare i due strumenti.

A causa di tutte queste motivazioni, si è preferito migrare direttamente il codice in modo da ottenere un'applicazione in grado di calibrare la camera all'istante, in maniera semplice ed intuitiva.
Linea per linea il codice è stato migrato al linguaggio di programmazione Java e si è sviluppata così l'applicazione della calibrazione che verrà spiegata nel dettaglio nel paragrafo 4.1 Applicazione mostrando le relative interfacce.
Tale codice migrato non presenta differenze logiche rispetto a quello in C++.

3.2.2 Implementazione del codice del tracking e creazione applicazione

La stessa cosa è stata fatta per il codice riguardante il monitoraggio degli strumenti laparoscopici.

Prima di arrivare a tale decisione, si è provato a mettere direttamente il codice in C++ in Android Studio, usando gli strumenti messi a disposizione da NDK e da JNI.
Si è notato che ciò portava a problemi difficili da risolvere, aumentando anche di molto i tempi computazionali.
È stato così deciso di riscriverlo interamente in linguaggio Java, in Android Studio.

Anche questo codice riscritto in Java, non presenta differenze logiche rispetto a quello scritto in C++. I filtri e le funzioni utilizzate, nonchè l'ordine con cui questi vengono applicati, rimangono gli stessi.
Avendo terminato di riscrivere il codice in Android Studio si è proseguito con la creazione dell'applicazione, facendo sì che fosse in grado di ottenere al meglio le traiettorie dei due marcatori.
Tale applicazione ci ha permesso così di monitorare i movimenti degli strumenti laparoscopici prelevandone e salvandone le posizioni 3D in documenti specifici.
Anche per quest'applicazione, nel paragrafo 4.1 Applicazione, verranno mostrate le interfacce per comprenderla al meglio.

3.2.3 Modifiche effettuate

I. Per rendere il tutto più scorrevole e immediato si è deciso di racchiudere queste due applicazioni in una sola, al contrario da come era stato fatto per l'applicazione originaria implementata per i computer.
   Nonostante non fosse stato deciso a priori, abbiamo ritenuto che unire le due applicazioni risultanti fosse di fondamentale importanza.
   In questo modo siamo stati in grado di semplificare il passaggio dei dati provenienti dall'applicazione della calibrazione a quella del *tracking*.
   Abbiamo fatto in modo che non fosse richiesta la presenza obbligatoria di una persona specializzata che si occupasse di passare i dati ottenuti dalla calibrazione all'applicazione del *tracking*.
   Così l'applicazione è risultata più semplice con un conseguente maggior utilizzo da parte dei chirurghi.
   Si è riusciti perciò a sviluppare un'unica applicazione completa a tutti gli effetti.
   L'organizzazione dei documenti di tale applicazione risultante verranno mostrati nel sottoparagrafo 3.2.4 Architettura file in Android Studio.

II. Sono state apportate piccole modifiche al simulatore durante l'avanzare del progetto, affinchè ci fossero le condizioni sufficienti e necessarie per poter eseguire le prove al meglio. Queste modifiche comprendevano:

- il rivestimento interno del simulatore con carta di colore nero in modo che non entrasse luce esterna da modificare i colori dei marcatori, interferendo in questo modo con la calibrazione;
- l'inserimento all'interno del simulatore con led in modo che il campo di allenamento fosse ben illuminato.

In seguito però, si è optato di inserire un'altra cassa all'interno del simulatore completamente bianca, dotata di luce propria e di un supporto per poggiare, con l'aiuto di un elastico, uno smartphone e tenerlo in posizione.

Figura 17:Cassa bianca vista frontale

Figura 18:Cassa bianca vista dall'alto

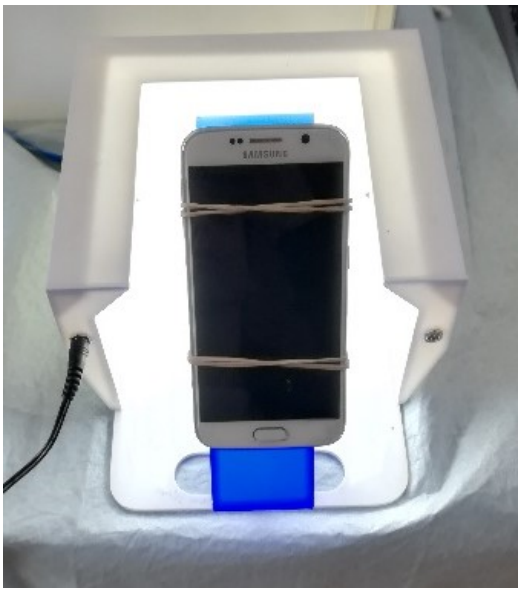

Figura 19:Cassa bianca con telefono di prova

Una volta inserita all'interno del simulatore, si ottiene il simulatore modificato come mostrato in Figura 20.

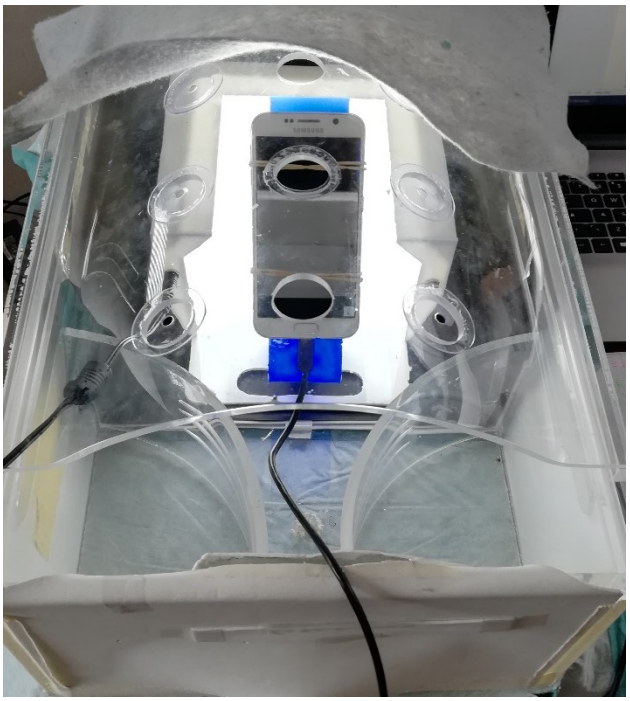

Figura 20:Simulatore modificato con cassa bianca all'interno

3.2.4 Architettura file in Android Studio

Nel passare i codici dal linguaggio di programmazione C++ a Java su Android Studio, abbiamo riscontrato alcune difficoltà dovute alle differenze nella gestione della memoria e nell'organizzazione delle classi.

Per quanto riguarda il C++, si può affermare senza alcun dubbio che è uno tra i cinque linguaggi di programmazione più utilizzati al mondo. Si possono realizzare progetti di grosse dimensioni perchè oltre ad essere stato pensato per la programmazione orientata agli oggetti, discende dal linguaggio C ereditandone la flessibilità, la potenza e la possibilità di programmare a basso livello.
Pertanto è l'unico linguaggio in grado di integrare i benefici della programmazione moderna ai vantaggi di quella precendente della programmazione strutturale. Inoltre, per tutti i sistemi operativi a disposizione troviamo compilatori C++ e ciò ne assicura una certa portabilità [48].

Anche Java, come è stato spiegato nel paragrafo precedente, è un linguaggio di programmazione orientato agli oggetti in grado di essere eseguito da qualsiasi piattaforma grazie all'uso della *Java Virtual Machine* (JVM).
In Android Studio il comportamento delle applicazioni viene definito da un codice Java, mentre le interfacce vengono descritte in file xml.

Le differenze tra C++ e Java possono essere di tipo sintattico, architetturale e strutturale. Le differenze di sintassi riguardano il modo in cui vengono definite le variabili, i vettori, le funzioni, i commenti e in ultimo le espressioni. La sintassi di base di Java viene mantenuta pressochè simile a quella del C++, però alcune delle caratteristiche più complesse come l'ereditarietà multipla, l'istruzione goto e l'aritmetica dei puntatori sono state modificate o addirittura completamente cancellate da rendere il passaggio tra i due linguaggi il più semplice e intuitivo possibile [40].

Nel linguaggio C++, a differenza del Java, sono utilizzati i puntatori che sono particolari tipi di variabili che permettono agli sviluppatori di poter accedere e modificare la memoria del calcolatore.
Per quanto riguarda la gestione della memoria in Java, l'utente non ha la possibilità di controllarla o gestirla perchè il tutto viene coordinato da un sistema automatico.
Infatti, quando un metodo viene eseguito, gli viene automaticamente assegnato uno spazio di memoria in grado di contenere i valori delle variabili che esso utilizza. Una volta che il metodo termina di essere utilizzato la memoria viene liberata.
Nel nostro caso particolare, i codici consegnatoci in C++ non erano orientati agli oggetti, perciò, in aggiunta, è stato modificato il paradigma dalla programmazione strutturata alla programmazione a oggetti.
Le differenze strutturali e architetturali incontrate sono state evidenti in quanto, nella programmazione orientata agli oggetti l'organizzazione dei file prevede l'utilizzo di un file per ogni classe contenente i metodi e gli attributi che ne definiscono le caratteristiche e le funzionalità. Le classi inoltre, possono essere riunite in *package* al fine di distinguere quelle

che hanno caratteristiche in comune da quelle che non ne hanno. Nella programmazione strutturata invece, vi è presente un minor numero di documenti contententi variabili e funzioni globali.

Possiamo vedere qui di seguito l'organizzazione dei documenti in Android Studio utilizzati per creare la nostra applicazione. Le cartelle principali da cui ricaviamo tutti i file necessari alla creazione dell'applicazione sono :

1. *App*
2. *OpenCVLibrary320*
3. *Gradle Scripts*
4. *External Build Files*

Per ognuna di queste cartelle andremo a spiegarne i relativi contenuti.

- Partendo dalla cartella "App" vi troviamo :

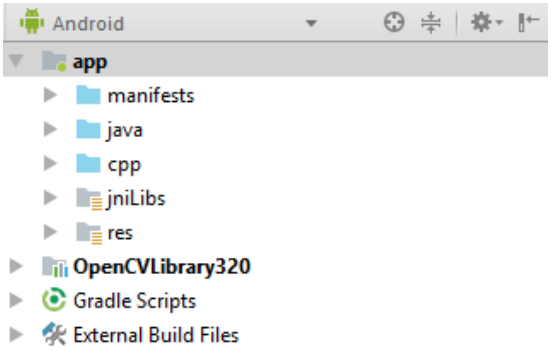

Figura 21:Architettura delle quattro cartelle principali in Android Studio

- *Manifest* : contiene le informazioni relative alla configurazione dell'applicazione. Si occupa di richiedere i permessi e definire le caratteristiche principali dell'applicazione, come ad esempio descrivere l'*activity* principale [36].
- Java : contiene l'intero codice sorgente dell'applicazione Android organizzato in pacchetti. Nel pacchetto principale ci sono le *activity*, negli altri vengono processate le immagini e gestiti i vari file.
- *Cpp* : contiene le librerie di *OpenCV* in quanto sono scritte in linguaggio C++. Inoltre, nel nostro caso specifico, vi troviamo anche l'intero codice del *tracking* implementato in C++, poichè precedentemente era stato sperimentato l'utilizzo dei sistemi JNI e SDK non ottenendo buoni risultati.
- *JniLibs* : contiene i file JNI necessari per poter utilizzare *OpenCV.*
- *Res* : è costituita da altre sottocartelle :
    - ❖ *Layout* : contenente l'architettura grafica dei componenti dell'interfaccia utente. I documenti sono in file.xml

- *Mipmap* : contiene l'icona scelta per la nostra applicazione mostrata in Figura 22.

Figura 22:Icona Applicazione

- *Values* : contiene stringhe, colori, dimensioni e altre tipologie di valori che verrano utilizzate nel codice Java.
- *Raw* : contiene tutti i file utili per lo sviluppo dell'applicazione che non sono presenti in Android Studio ma che vengono aggiunti manualmente dal programmatore. In particolare, vi troviamo tutti i file ricavati dalla calibrazione come ad esempio i parametri della lente.

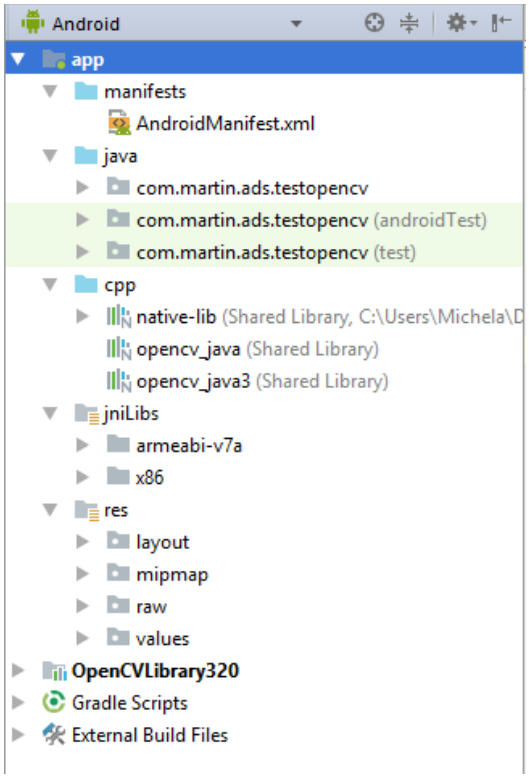

Figura 23:Architettura sottocartelle in App

- Successivamente troviamo la cartella *OpenCvLibrary320* in cui vi sono conservati tutti i documenti necessari per poter utilizzare le librerie che fanno parte del pacchetto di *OpenCV*.

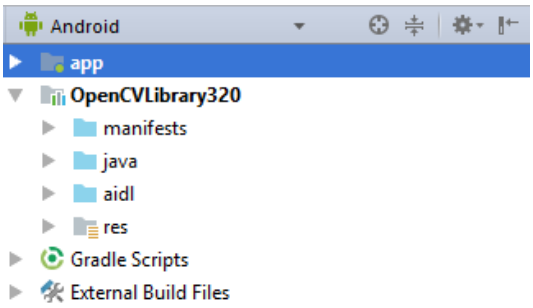

Figura 24:Architettura modulo OpenCV

- Nella cartella *Gradle Scripts* sono racchiusi i documenti relativi alla compilazione del progetto. In particolare troviamo i file *build.gradle* nei quali si specifica la versione del compilatore e quella del sistema operativo Android necessaria perchè il tutto funzioni.
  Nel nostro caso, come già accenato precedentemente, la versione del sistema operativo Android è al massimo di 25 API, ma si può utilizzare un qualunque dispositivo android che abbia una versione pari o superiore alla 17. Ciò è stato fatto per permettere la retrocompatibilità, ovvero per permettere a più dispositivi non attualizzati di poter installare e usare la nostra applicazione. In aggiunta vengono dichiarate le dipendenze, ovvero le librerie necessarie a cui il nostro progetto deve fare riferimento affinchè funzioni.

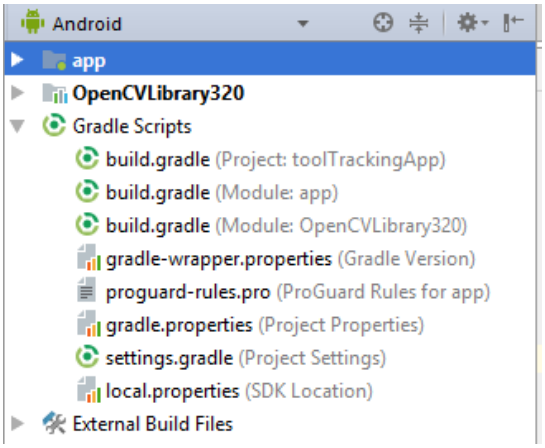

Figura 25:Architettura Gradle Scripts

- In ultimo troviamo la cartella *External Build File* che contiene il *Cmake*[8] con le istruzioni necessarie per poter compilare un codice in C++. Questa cartella è stata creata dopo aver configurato il progetto di Android per poter funzionare con librerie in C++.

[8] È un software libero multipiattaforma per l'automazione dello sviluppo, il cui nome è un'abbreviazione di *cross platform make*

3.2.5 Prove sperimentali

Una volta terminato l'intero progetto, sono state eseguite due prove statiche per poter verificare la correttezza e la precisione dei risultati ottenuti.
È stato preso come esempio l'articolo "*Feasibility of tracking laparoscopic instruments in a box trainer using a Leap Motion Controller*" [47], nel quale Oropesa I. Et al. spiegano come validare il sistema *Leap Motion*.
Basandoci su tale articolo, la prima prova consisteva nel valutare il comportamento dell'applicazione per un periodo di tempo prolungato (circa un'ora) rispetto al tempo realmente previsto.
Al contrario, nella seconda prova è stato valutato il comportamento dell'applicazione durante un periodo di 15 secondi.

Tali prove verranno spiegate nel dettaglio nei sottoparagrafi 3.2.5.1 Prova precisione a lungo termine e 3.2.5.2 Prova precisione a breve termine.
Per poter eseguire le prove è stata utilizzata una piattaforma di legno, creata appositamente per questi due esperimenti, affinchè potesse mantenere in posizione fissa gli strumenti laparoscopici.

Figura 26:Struttura in legno vista dall'alto

Figura 27:Struttura in legno vista laterale

La piattaforma è costituita da due altipiani forati, ogni foro dista dall'altro circa 25 mm.
I fori del primo altopiano hanno un diametro di 8 mm (corrispondente al diametro esterno di un trocar laparoscopico) invece, i diametri dei fori del secondo altopiano misurano 5 mm come il gambo dello strumento che verrà inserito all'interno.
Il risultato finale sarà come mostrato in Figura 28.

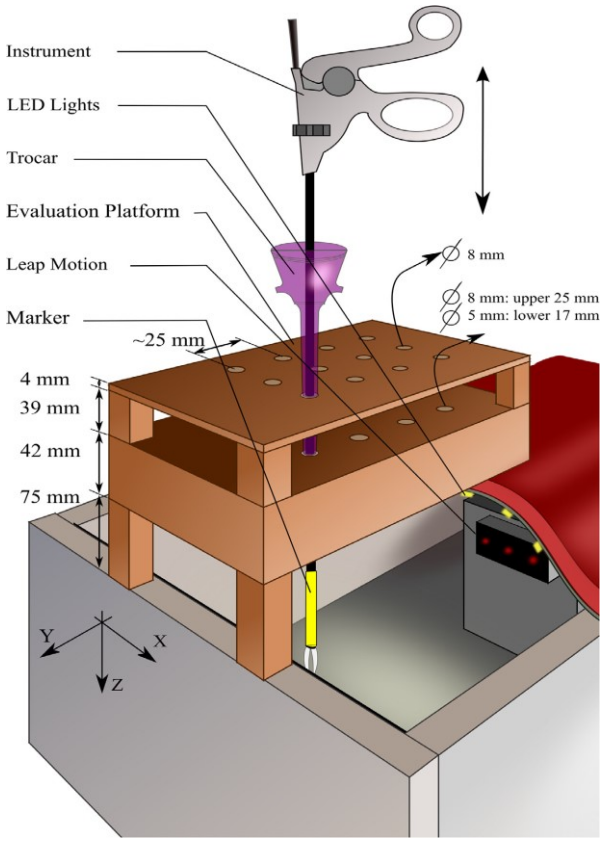

Figura 28:Sistema di prova utilizzato nell'articolo [47]

Entrambe le prove sono state eseguite utilizzando tre marcatori in modo da avere un confronto.
I marcatori con proprietà termorestringenti posti alla base degli strumenti laparoscopici hanno anch'essi un diametro di circa 5 mm.
Posizionando così la piattaforma sopra il simulatore, siamo stati in grado di mantenere in una posizione fissa lo strumento laparoscopico per tutto il tempo a noi necessario.

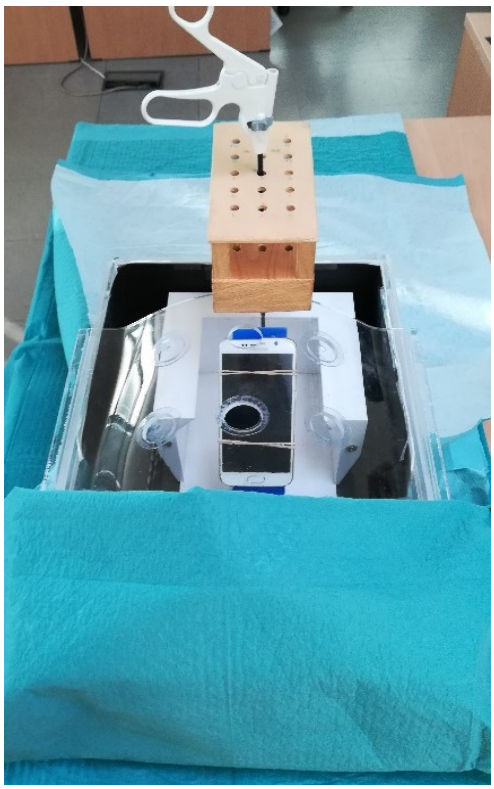

Figura 29:Esecuzione prova vista frontale

Figura 30:Esecuzione prova vista laterale

La distanza tra la fotocamera del dispositivo mobile e le prime tre posizioni della piattaforma è di circa 9 cm per l'asse y.
La distanza calcolata invece, tra la fotocamera del dispositivo e le ultime tre file della piattaforma è approssimativamente di 19 cm.

3.2.5.1 Prova precisione a lungo termine

La prova della precisione a lungo termine consisteva nel mantenere in posizione fissa lo strumento laparoscopico per circa un'ora, affinchè si potessero prelevare i dati delle posizioni 3D risultanti.
Tale prova è stata effettuata usando un marcatore alla volta tra quelli a disposizione. I colori a utilizzati sono stati rosso, verde e blu.
Lo strumento laparoscopico è stato sistemato in posizione verticale all'interno del simulatore tramite la struttura di legno a disposizione, come mostrato in Figura 30.
Precisamente lo strumento laparoscopico è stato sistemato in una posizione scelta in modo che non fosse nè troppo vicina nè troppo lontana rispetto alla fotocamera del dispositivo.
Inoltre, come si può notare in Figura 31, la posizione è stata scelta il più centrale possibile in modo da poterla visualizzare al meglio e da non avere grossi problemi di messa a fuoco.
Le ultime due file non sono state considerate, in quanto con l'aggiunta della cassa bianca all'interno del simulatore, non vi rimaneva spazio per inserire gli strumenti laparoscopici.

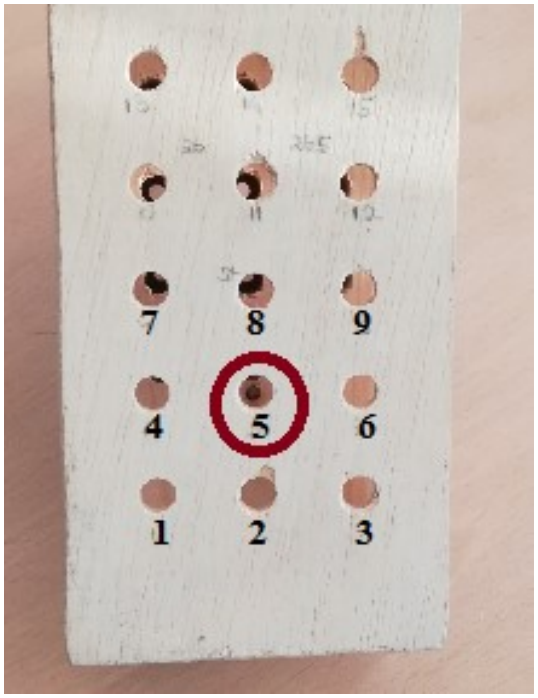

Figura 31:Piattaforma di legno con evidenziata la posizione utilizzata per la prova a lungo termine

Una volta inseritovi lo strumento laparoscopico con il marcatore, è stato tenuto fisso in modo che potesse essere seguito continuamente dall'applicazione per tutto il tempo necessario.

Successivamente i dati relativi alle posizioni 3D risultanti sono stati analizzati con Matlab calcolandone la media e la deviazione standard per ogni misura effettuata.
I grafici ottenuti vengono mostrati nel sottoparagrafo 4.2.1 Risultati prova a lungo termine.

3.2.5.2 Prova precisione a breve termine

La prova di precisione a breve termine prevedeva il tracciamento continuo dello strumento laparoscopico da parte dell'applicazione per un periodo di tempo più breve rispetto a quello previsto.
Lo strumento laparoscopico è stato così inserito all'interno del simulatore attraverso la struttura di legno, utilizzando solo nove delle quindici posizioni a disposizione.
Sono state utilizzate nove posizioni in quanto, avendo avuto la necessità di introdurre all'interno del simulatore la scatola bianca e illuminata, non vi rimaneva abbastanza spazio per inserirvi lo strumento nelle ultime due file.
In Figura 32 mostriamo le nove posizioni utilizzate.

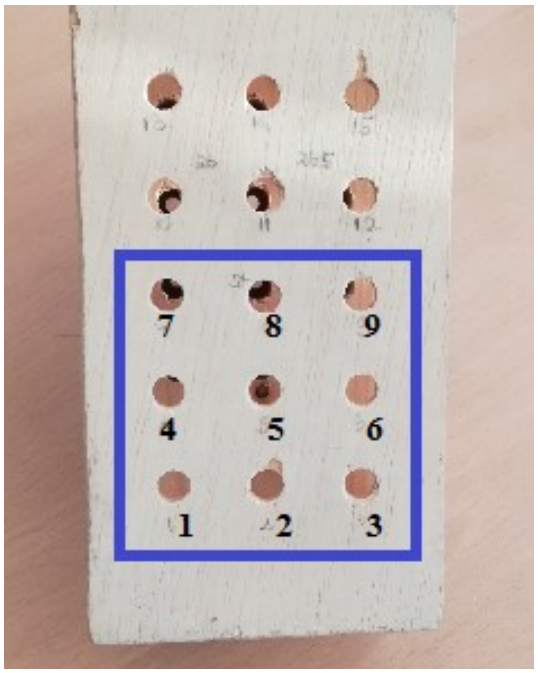

Figura 32:Piattaforma di legno con evidenziate le posizioni utilizzate per la prova di precisione a breve termine

Sono state effettuate 10 misurazioni, ognuna della durata di 15 secondi, per ciascuna delle nove posizioni, per un totale di 90 misurazioni per marcatore e 270 in totale.

Tutto ciò è stato eseguito per un marcatore alla volta, cercando di mantenere la massima precisione possibile nello spostare lo strumento laparoscopico da una posizione ad un'altra. Si è cercato inoltre di limitare al minimo i movimenti della struttura di legno perchè questi avrebbero portato ad errori nei risultati ottenuti.

Per far in modo che la struttura di legno risultasse più stabile è stata incollata al simulatore tramite del *Blu Tack*.

Anche per questo tipo di esperimento i dati ottenuti sono stati analizzati su Matlab calcolandone la media e la deviazione standard per ogni misurazione effettuata.
È possibile visualizzare i grafici nel sottoparagrafo 4.2.2 Risultati prova a breve termine.

4. Risultati

In questo capitolo verranno presentate tutte le interfacce dell'applicazione creata, insieme ad un breve cenno sui modelli utilizzati per compiere le attività che portano ad una successiva valutazione.
Successivamente verranno mostrati nel dettaglio i risultati delle prove sperimentali di precisione sia a lungo che a breve termine. Per capire al meglio, sono stati inseriti i grafici più rilevanti ottenuti avendo calcolato su Matlab la media e la deviazione standard dei dati ricavati dall'applicazione.
Tutti i grafici rimanenti, ritenuti non così rilevanti al fine dell'analisi, sono stati inseriti al fondo nel Glossario Grafici.

4.1 Applicazione

L'applicazione sviluppata risulta semplice e intuitiva.
Consiste in una prima parte di calibrazione, ed una seconda parte nella quale vengono salvati in un file .xml, i punti 3D ottenuti tracciando le traiettorie degli strumenti laparoscopici.
Vengono mostrate in questo capitolo alcune interfacce dell'applicazione in modo che si possa capire più nel dettaglio ciò che è stato fatto.
Se si accede all'applicazione per la prima volta, la prima interfaccia di visualizzazione sarà quella mostrata in Figura 33, nella quale vengono mostrati i simboli relativi all'*Universidad Politécnica de Madrid*, in cui è stato svolto l'intero progetto di tesi.

Figura 33:Prima interfaccia dell'applicazione con simboli dell'Universidad Politécnica di Madrid

Tramite un semplice *click* sullo schermo si passa all'interfaccia successiva che ci permette di eseguire la calibrazione dei due marcatori, premendo il tasto "*Calibrate*", o di accedere direttamente al monitoraggio degli strumenti laparoscopici, tramite il tasto "*Start*".
È possibile visualizzarla in Figura 34.

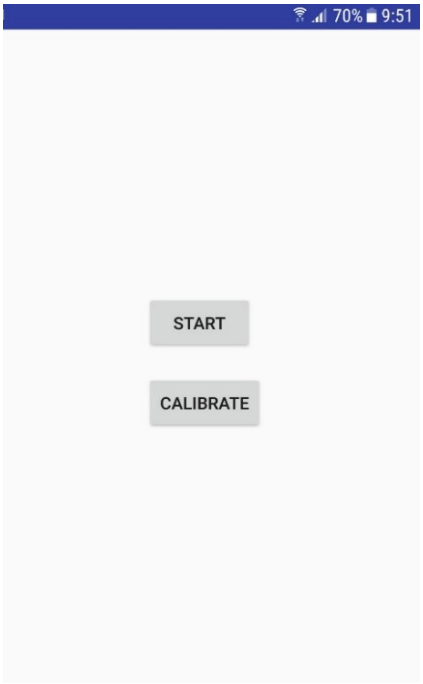

Figura 34:Schermata con la quale è possibile scegliere se eseguire la calibrazione o iniziare il monitoraggio degli strumenti laparoscopici

In questo caso, essendo la prima volta che si utilizza l'applicazione, senza aver calibrato i due marcatori non è consigliato accedere direttamente alla fase del monitoraggio in quanto l'applicazione non avrebbe nulla da ricercare.
Il chirurgo inizierà perciò la calibrazione dei due marcatori tramite il tasto "*Calibrate*".
Verranno visualizzati in automatico due messaggi. Nel primo messaggio verrà richiesto all'utente chirurgo di permettere all'applicazione di poter fare foto e registrare video(Figura 35).

Nel secondo messaggio verrà richiesto il permesso di accedere ai contenuti multimediali e tutti gli archivi del dispositivo al fine di utilizzarne le immagini (Figura 356).

Figura 35:Primo permesso richiesto

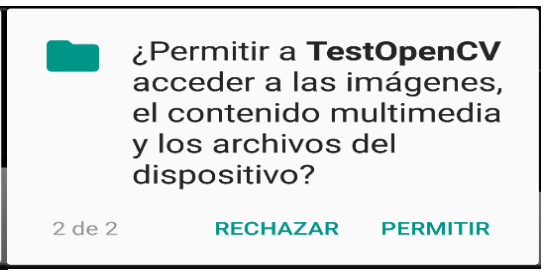

Figura 36:Secondo permesso richiesto

Una volta rilasciati i permessi non sarà più necessario che questi messaggi compaiano di nuovo. Pertanto tali messaggi, a partire dalla seconda volta che si accede all'applicazione, non verranno più mostrati.

Si visualizzerà in seguito la schermata in Figura 37, la quale ci permetterà di eseguire la calibrazione del primo marcatore (*marker* 1) che sarà scelto a piacere dall'utente chirurgo tra i due a disposizione.

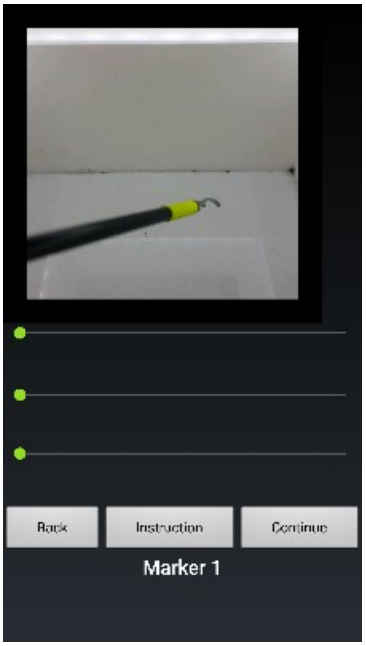

Figura 37:Prima schermata dell'app calibrazione camera

In Figura 37 possiamo vedere che l'interfaccia è divisa in tre parti:

- una parte superiore nella quale si vede ciò che la fotocamera del dispositivo sta riprendendo;
- una parte nel mezzo che mostra tre barre relative ai valori HSV che l'utente chirurgo andrà a modificare;
- l'ultima parte, nel basso, nella quale ci sono diversi bottoni creati per aiutare l'utente chirurgo.

Il bottone centrale, è il bottone delle "*Instruction*", che una volta premuto, mostra a video un messaggio con la spiegazione, come in Figura 38, che l'utente chirurgo dovrà seguire per poter calibrare al meglio il marcatore.

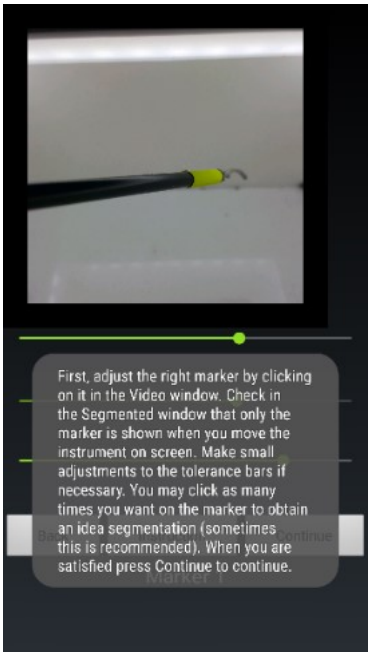

Figura 38:Schermata con le Istruzioni per calibrare il primo marcatore

La spiegazione che sarà visualizzata è la seguente: "Per prima cosa, ottieni i valori HSV del marcatore cliccandoci sopra nella schermata video.
Modifica se è necessario i valori delle tre barre al fine di vedere al meglio solo il marcatore che si sta calibrando.
Controlla che si veda a video solo il marcatore quando viene mosso all'interno del campo di allenamento. Puoi tornare indietro e ricalibrare il marcatore quante volte vuoi. Se invece sei soddisfatto premi il tasto *Continue* per continuare".
Una volta lette e capite le istruzioni si può passare alla fase della calibrazione vera e propria.

L'utente dovrà ora cliccare sul marcatore.
Se cliccando sbaglia e clicca da un'altra parte, con l'aiuto del tasto "*Back*" potrà tornare indietro e riprovare tutte le volte di cui ha bisogno.

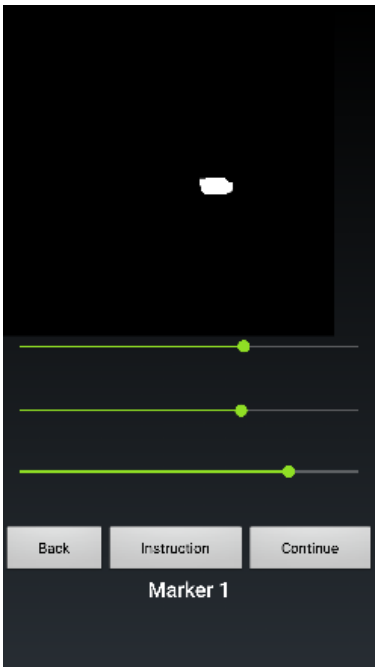

Figura 39:Schermata con la quale si scelgono i valori HSV per il primo marcatore

Una volta che l'utente ha cliccato sul marcatore si ottiene un'interfaccia nella quale si vede solo il marcatore, colorato interamente di bianco.
Nel caso in cui dovessero comparire altri pixel dell'immagine di colore bianco, l'immagine sarebbe soggetta a rumore, causando difficoltà nel monitorare gli strumenti laparoscopici.
Tale rumore perciò, andrebbe eliminato modificando i valori HSV attraverso le tre barre a disposizione.
L'immagine finale deve essere pulita come quella in figura 39.

Una volta che sono stati scelti i valori HSV ritenuti più esplicativi per quel marcatore, con il tasto "*Continue*" si passa alla calibrazione del secondo marcatore.

Le istruzioni del secondo marcatore, come si vede in Figura 40, sono le medesime che per il primo: "Per prima cosa, ottieni i valori HSV del marcatore cliccandoci sopra nella schermata video. Modifica se è necessario i valori delle tre barre al fine di vedere al meglio solo il marcatore che si sta calibrando. Controlla che si veda a video solo il marcatore quando viene mosso all'interno del campo di allenamento. Puoi tornare indietro e ricalibrare il marcatore quante volte vuoi. Se invece sei soddisfatto premi il tasto *Continue* per continuare".

Per ottenere i valori HSV infatti, basterà cliccare sul marcatore e modificare i valori delle tre barre (Figura 41).
Anche in questo caso se l'utente si sbaglia e non clicca sul marcatore avrà la possibilità di tornare indietro e riprovare.

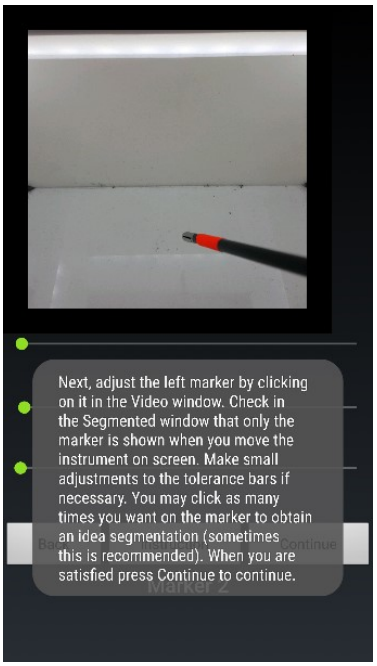

Figura 40:Schermata con le istruzioni per calibrare il secondo marcatore

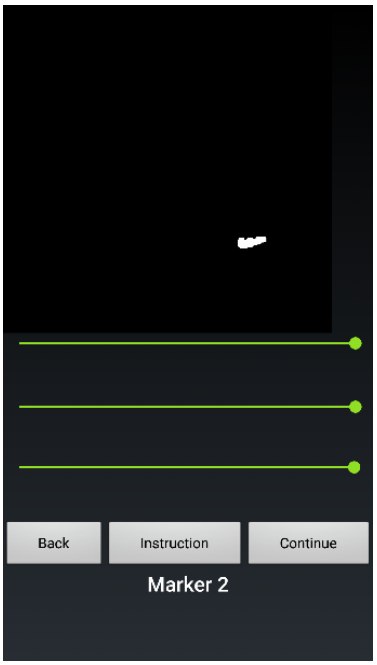

Figura 41:Schermata con la quale si scelgono i valori HSV per il secondo marcatore

Una volta eseguita la calibrazione anche per i secondo marcatore tramite il tasto "*Continue*" si accederà all'ultima fase.

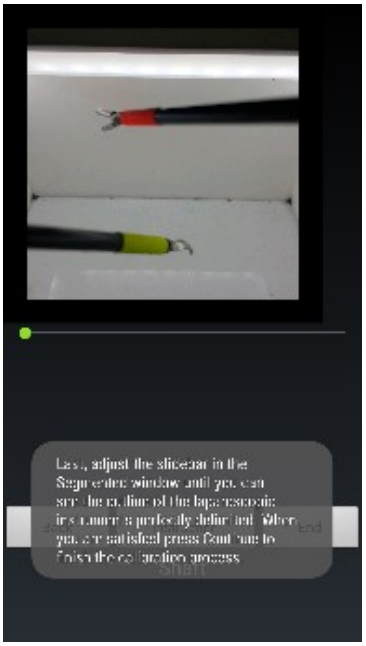

Figura 42:Schermata con le istruzioni per eseguire lo Shaft

L'ultima fase della calibrazione, corrisponde alla fase chiamata "Shaft".

Vi sono presenti, anche per questa fase, le istruzioni che spiegano all'utente come agire: "Infine, regola la barra di scorrimento finchè non vedi il contorno degli strumenti perfettamente delineato. Quando sarai soddisfatto, premi il tasto "*End*" per terminare il processo di calibrazione".

In quest'ultima fase bisognerà vedere il contorno del gambo dello strumento laparoscopico in modo da distinguerlo il più possibile dallo sfondo come in Figura 43.

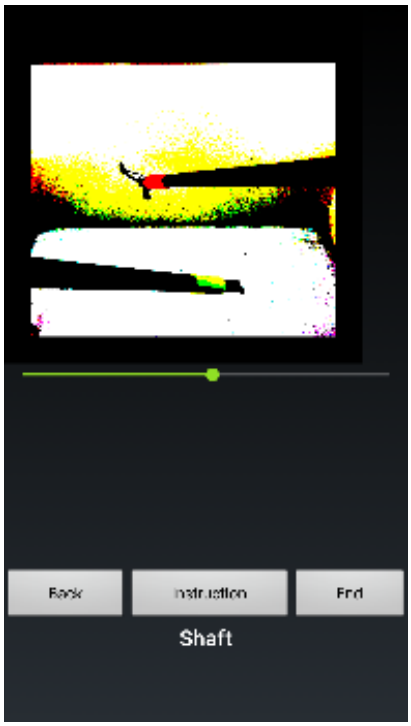

Figura 43:Schermata con la quale è possibile scegliere un contrasto adeguato

Si desidera aumentare il contrasto tra gli strumenti e lo sfondo così da poterli visualizzare al meglio.
Una volta terminato, tramite il bottone "*End*", sarà possibile uscire dall'applicazione della calibrazione e iniziare ad usare quella del *tracking*.

Una volta premuto il tasto "*End*" si genererà un file .xml con i dati ottenuti dalla calibrazione. Tali dati verranno automaticamete passati all'applicazione del tracking per riconoscere i marcatori. Come spiegato nel paragrafo 3.2.3 Modifiche effettuate, ciò rappresenta una modifica aggiuntiva del nostro lavoro che lo ha reso più completo ed efficace.

Successivamente si ritornerà all'interfaccia mostrata in Figura 34, che permetterà di iniziare ad usare l'applicazione del *tracking* tramite il bottone "*Start*", oppure, nel caso in cui non si fosse soddisfatti della calibrazione appena eseguita, tramite il tasto "*Calibrate*", è possibile ricalibrare di nuovo.

Premendo il tasto "Start" si arriva così all'applicazione del *tracking* che ci permette di monitorare i movimenti degli strumenti laparoscopici seguendo i marcatori appena calibrati.

Quest'applicazione, tramite l'implementazione di filtri, spiegati precedentemente, andrà a calcolare i bordi degli strumenti laparoscopici marcandoli con delle linee colorate come si vede in Figura 44, Figura 45 e Figura 46, evidenziando il centroide e i punti B1 e B2.

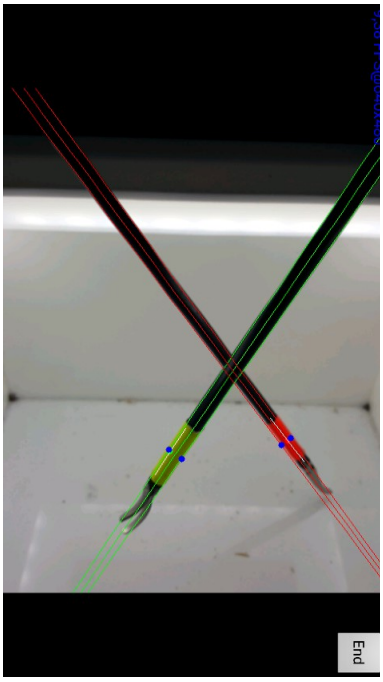

Figura 44:Risultati dell'applicazione del monitoraggio

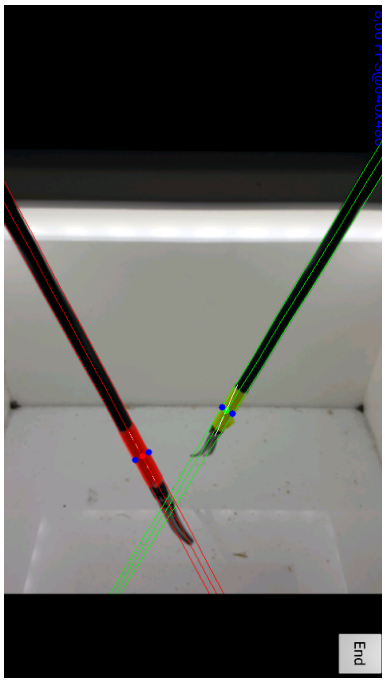

Figura 45:Risultati dell'applicazione del monitoraggio

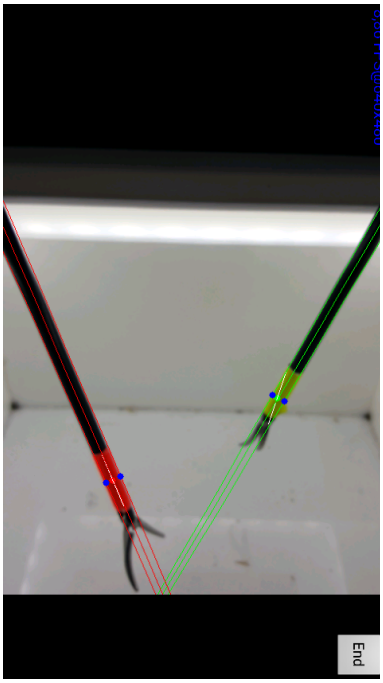

Figura 46:Risultati dell'applicazione del monitoraggio

È possibile notare in alto a sinistra della Figura 44, Figura 45 e Figura 46, una scritta di colore azzurro, nella quale sono riportati i valori dei *frame* per secondo e la risoluzione dello schermo[9] che è stata scelta di 640x480 pixel.

Per quanto riguarda invece i *frame* per secondo la nostra applicazione arriva a un massimo di 8. Si è provato ad aumentarli ma senza risultati.

Si pensa che ciò sia dovuto al fatto che il codice non è ottimizzato e l'applicazione nel processare le immagini fatichi molto. Inoltre, l'applicazione è stata utilizzata sempre in versione di debug. Tale versione ci permette di trovare eventuali errori andando ad analizzare linea per linea il nostro codice.

L'uso della versione di debug ci permette anche di visualizzare tutte le risorse che l'applicazione utilizza nel momento in cui viene lanciata. Tutti questi motivi portano ad abbassare il numero dei *frame* per secondo verticalmente.

Si suppone che se l'applicazione fosse stata utilizzata in modalità release invece che in modalità debug, i *frame* per secondo sarebbero aumentati di molto.

[9] Con risoluzione dello schermo si intende il numero dei pixel orizzontali e verticali presenti o sviluppabili in uno schermo

Nonostante il numero di *frame* per secondo basso, si riesce a tracciare i movimenti degli strumenti laparoscopici in *real-time* per tutto il tempo necessario.
Tramite l'implementazione dei filtri di Canny, Gaussiano, il bilaterale, Hough e Kalman è possibile ottenere una stima il più possibile simile a quella reale.

Al termine dell'utilizzo dell'applicazione, ogni volta che viene premuto il tasto "*End*" direttamente dall'interfaccia finale, come per esempio quella in Figura 46, si generano file contenenti le posizioni 3D monitorate. Un esempio di tali file è mostrato in Figura 47.

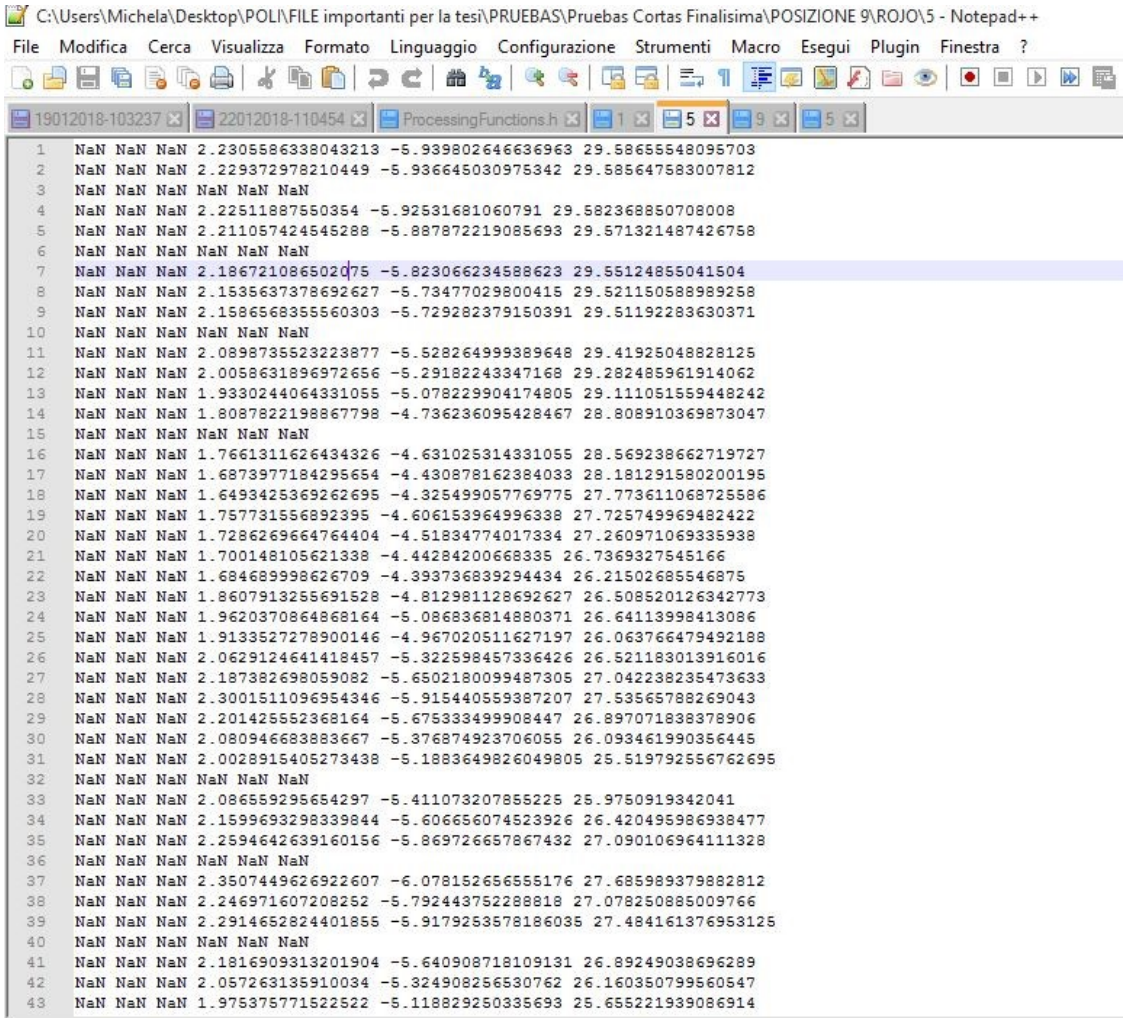

Figura 47:File contenente le posizioni 3D in centimetri di uno strumento laparoscopico

In Figura 47 vengono mostrate sei colonne, tre relative alle posizioni X, Y e Z del primo marcatore calibrato e tre per il secondo. Le posizioni X, Y e Z sono calcolate come distanze in centimetri tra il centroide del marcatore e l'obiettivo della fotocamera.
Come si nota in Figura 47, dalle prime tre colonne senza un valore, tale file è stato ricavato tracciando solo il secondo marcatore come mostrato in Figura 48.
È possibile che anche nelle ultime tre colonne, ci siano dei Not a Number (NaN) dovuti a problemi di messa a fuoco dell'applicazione.

Proprio questi file saranno utilizzati successivamente in Matlab come spiegato nel paragrafo 4.2 Risultati delle prove.

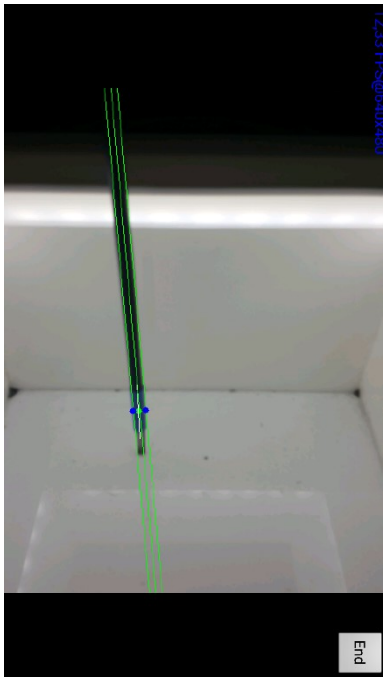

Figura 48:Tracciamento di solo uno strumento laparoscopico

4.2 Risultati delle prove

Effettuate le due prove con i tre marcatori a disposizione, siamo riusciti a ricavarne i dati relativi alle posizioni 3D della punta degli strumenti laparoscopici.
In generale i risultati sono stati soddisfacenti, poichè confrontando i valori ottenuti dall'applicazione con quelli misurati direttamente sul campo di allenamento, non abbiamo riscontrato grandi differenze.
In particolare verranno mostrati e discussi i grafici ottenuti per entrambe le prove effettuate, valutando per ogni marcatore la deviazione standard e la deviazione dalla media.

4.2.1 Risultati prova a lungo termine

Dopo aver seguito ininterrottamente per un'ora lo strumento laparoscopico nella posizione numero cinque, calcolando la deviazione standard dalla media per ciascuna misurazione, siamo riusciti ad ottenere l'andamento della dispersione temporale a lungo termine di tutti e tre i marcatori.

Raffigurando l'andamento della deviazione standard rispetto alla media, siamo in grado di osservare la dispersione delle posizioni ottenute intorno ad un indice di posizione, che nel nostro caso è la media aritmetica.

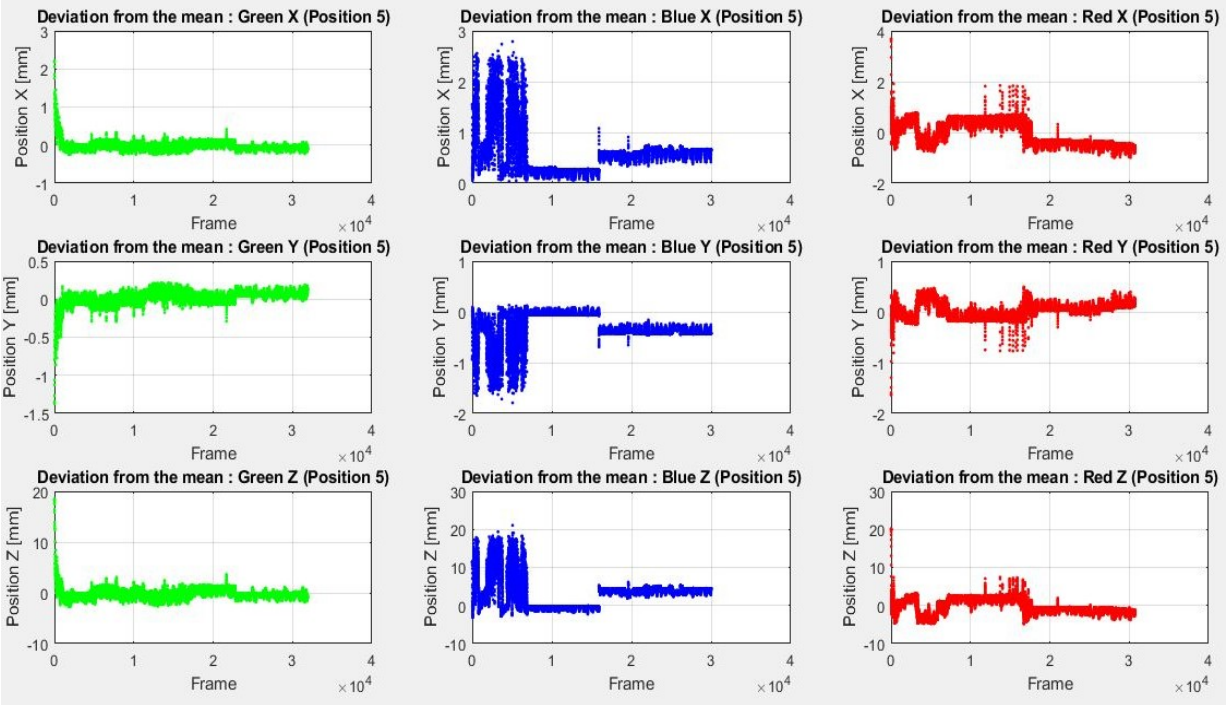

Grafico 1:Confronto deviazioni standard delle medie die tre marcatori per le posizioni X, Y e Z

Sull'asse delle ascisse vengono riportati tutti i frame ottenuti in un ora di tempo (che sono all'incirca 30800 per ogni marcatore), mentre sull'asse delle ordinate vi sono rappresentate le posizioni X,Y e Z in centimetri ricavate dalle prove eseguite.
Come si può notare il marcatore che ha dato risultati molto buoni è stato il marcatore di colore verde. Nonostante abbia avuto una singolare dispersione iniziale, successivamente si è ripreso mantenendosi costante fino al termine della prova.
Il marcatore che ha dato i peggiori risultati è stato il blu. Dai grafici risultanti infatti, si denota una grossa discontinuità durante i primi frame da non poter considerare validi i valori ottenuti. In seguito, è possibile notare un tratto costante durato pochi frame, dopo i quali viene perso il segnale per poi essere rispreso dopo qualche secondo.
Anche per il marcatore di colore rosso possiamo notare un'incertezza iniziale che viene successivamente sostituita da tratti costanti.

In laboratorio si è pensato che il marcatore blu essendo un colore più scuro degli altri due, è anche il più difficile da monitorare. Si pensa infatti, che possa essere confuso dall'applicazione con il gambo nero degli strumenti laparoscopici.

4.2.2 Risultati prova a breve termine

Una volta ottenute le 90 misurazioni per ogni marcatore, sono stati analizzati i valori delle posizioni con Matlab. Caricati tutti i file necessari, è stato creato un vettore contenente i valori medi di ciascuna misurazione.
A partire da questi vettori viene stimata la media totale dei valori di spostamento per ogni posizione.
Dalla media totale siamo stati in grado di calcolare la distanza statica tra le diverse posizioni, tramite l'Equazione 1.

$$Static\ Distance = \sqrt{(x_p - x_{ap})^2 + (y_p - y_{ap})^2 + (z_p - z_{ap})^2}$$

Equazione 1: Distanza statica

Con il termine:

- Xp indichiamo il valore della media totale nella posizione p per l'asse X;
- Xap indichiamo il valore della media totale per l'asse X nella posizione p+1, ovvero la posizione adiacente a quella precedentemente considerata;
- Yp indichiamo il valore della media totale nella posizione p per l'asse Y;
- Yap indichiamo il valore della media totale per l'asse Y nella posizione p+1, ovvero la posizione adiacente a quella precedentemente considerata;
- Zp indichiamo il valore della media totale nella posizione p per l'asse Z;
- Zap indichiamo il valore della media totale per l'asse Z nella posizione p+1, ovvero la posizione adiacente a quella precedentemente considerata.

I valori di distanza statica che possono essere considerati accettabili, sapendo che la distanza tra un foro e l'altro è di 2,5 cm, sono compresi tra 1,9 cm e 3,7 cm.
Analizzando i valori risultanti di distanza statica tra le diverse posizioni abbiamo riscontrato alcune anomalie.
I valori che possono essere considerati più attendibili, per tutti e tre i marcatori, li troviamo tra le posizioni 1-4, 4-7, 5-8, 3-6 ed infine la posizione 6-9.
Ricordando com'è fatta la struttura di legno utilizzata, queste posizioni rimangono l'una dietro l'altra e sono visualizzate dalla fotocamera in maniera sequenziale, com'è possibile visualizzare in Figura 49.

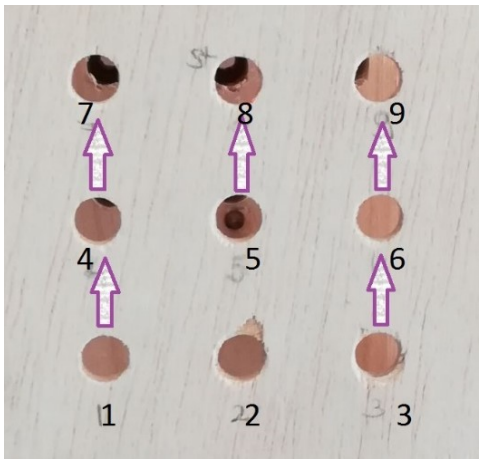

Figura 49:Posizioni con valori più accettabili di distanza statica

Valori meno precisi invece, li possiamo visualizzare tra le posizioni 7-8, 1-2, 8-9 ed in ultimo nella posizione 2-3. Tali posizioni vengono visualizzate dalla fotocamera non in maniera sequenziale ma in riga.

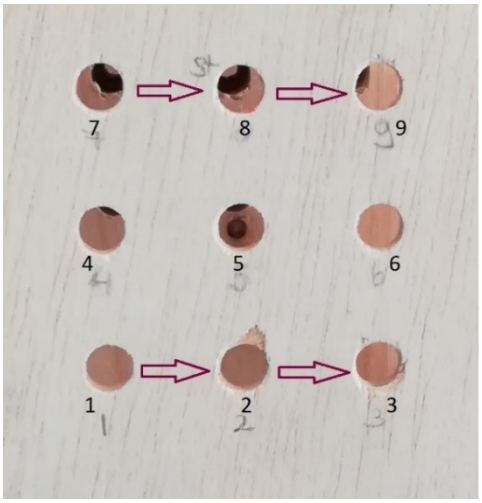

Figura 50:Posizioni con valori peggiori di distanza statica

Si pensa che ciò sia dovuto al fatto che le posizioni 1, 2 e 3 siano molto vicine all'obiettivo della fotocamera che fa fatica nel metterle a fuoco.
La stessa cosa vale per le posizioni dell'ultima fila che, essendo le più lontane possono creare difficoltà nell'identificare i marcatori al meglio.

Per quanto riguarda ciascun specifico marcatore, dai risultati notiamo che:

- il marcatore di colore blu è più riconoscibile e monitorabile nelle prime sei posizioni, nelle ultime tre invece, ricaviamo valori decisamente fuori range;
- Per il marcatore di colore verde la situazione si ribalta, in quanto valori più realistici delle distanze calcolate li troviamo nelle posizioni delle ultime due file;
- Infine, per quanto riguarda il marcatore di colore rosso, notiamo una certa omogeneità dei valori ricavati. Tale marcatore infatti, è quello che viene considerato

il più rappresentativo poichè in quasi tutte le posizioni della struttura otteniamo valori di distanza statica in grado di soddisfarci.

Successivamente è stata calcolata la deviazione standard come differenza tra il vettore contenente le medie di ciascuna posizione ed il valore della media totale. In questo modo vengono mostrati, tramite figure, gli andamenti della media e della deviazione standard per tutte le nove posizioni e per tutti e tre gli assi X, Y e Z.

Vediamo ora i grafici rappresentanti la media e la deviazione standard risultanti per le posizioni ritenute più rilevanti ai fini dell'analisi.

- Per quanto riguarda i grafici relativi alla media, sull'asse delle ascisse è stato posto il numero delle misurazioni effettuate (equivalente a dieci) mentre, sull'asse delle ordinate vi sono i vettori contenenti i valori delle medie in centimetri ricavate da ogni misurazione;
- Per i grafici riguardanti la deviazione standard, sull'asse delle ascisse è possibile visualizzare la posizione (che può essere X, Y o Z) ed il marcatore in questione. Nell'asse delle ordinate vi sono presenti i valori di deviazione standard calcolati in centimetri.

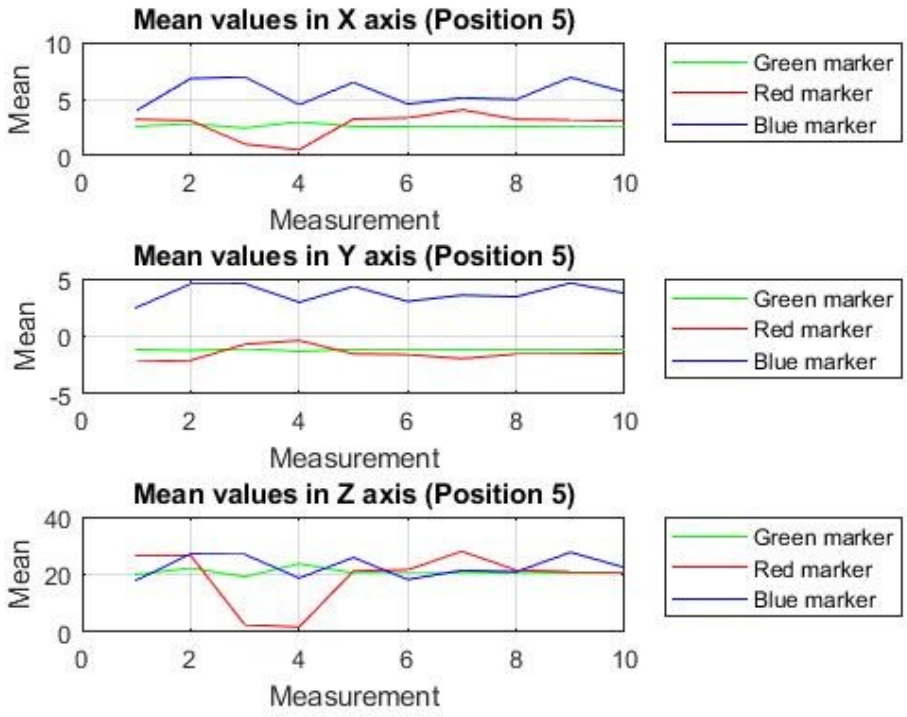

Grafico 2:Posizione5-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

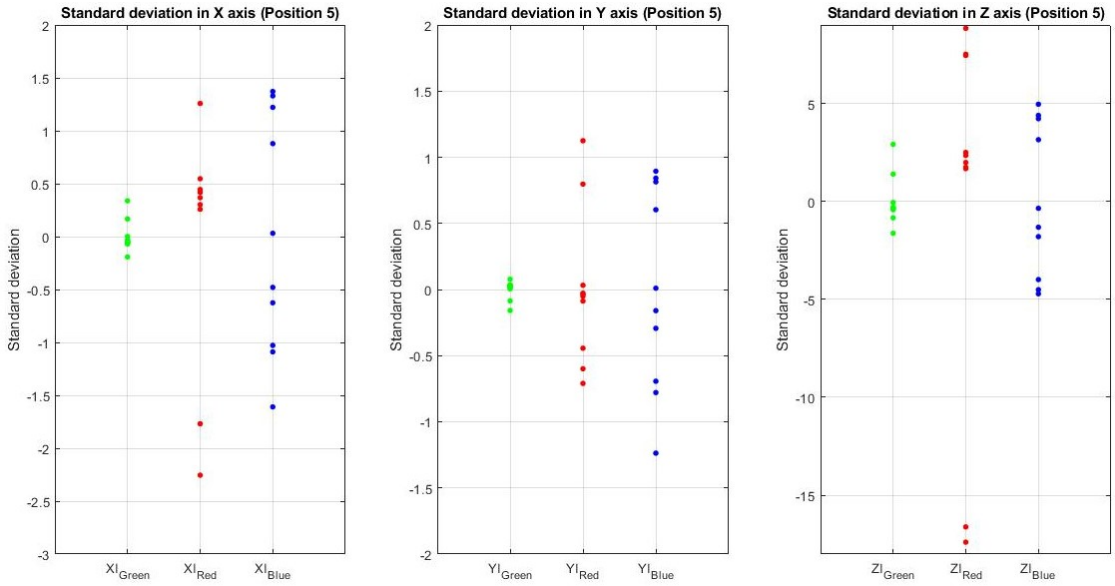

Grafico 3:Posizione5-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore.

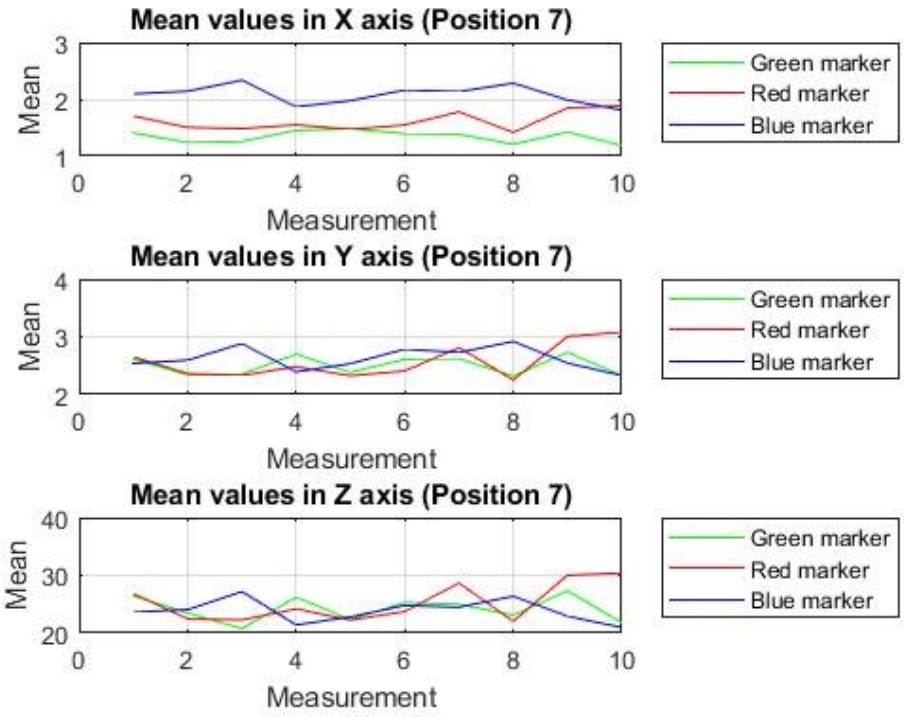

Grafico 4:Posizione7-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

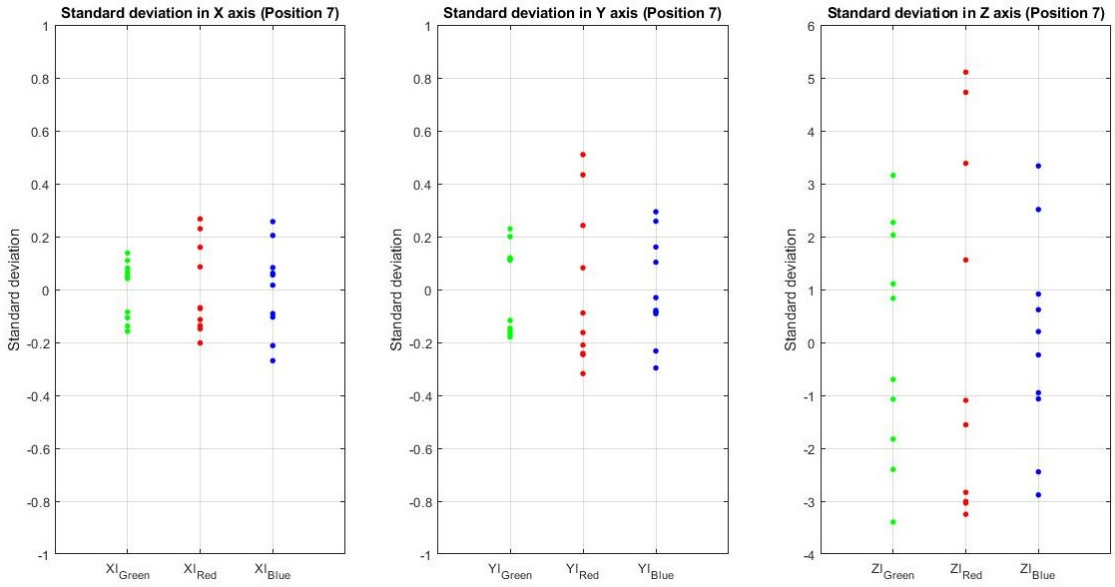

Grafico 5:Posizione7-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore

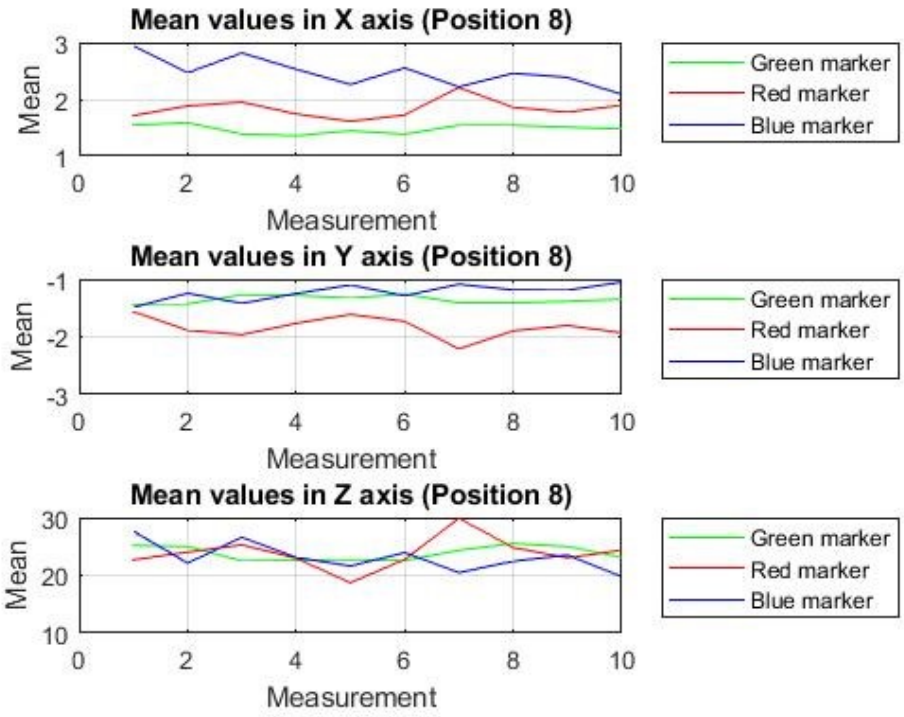

Grafico 6:Posizione8-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

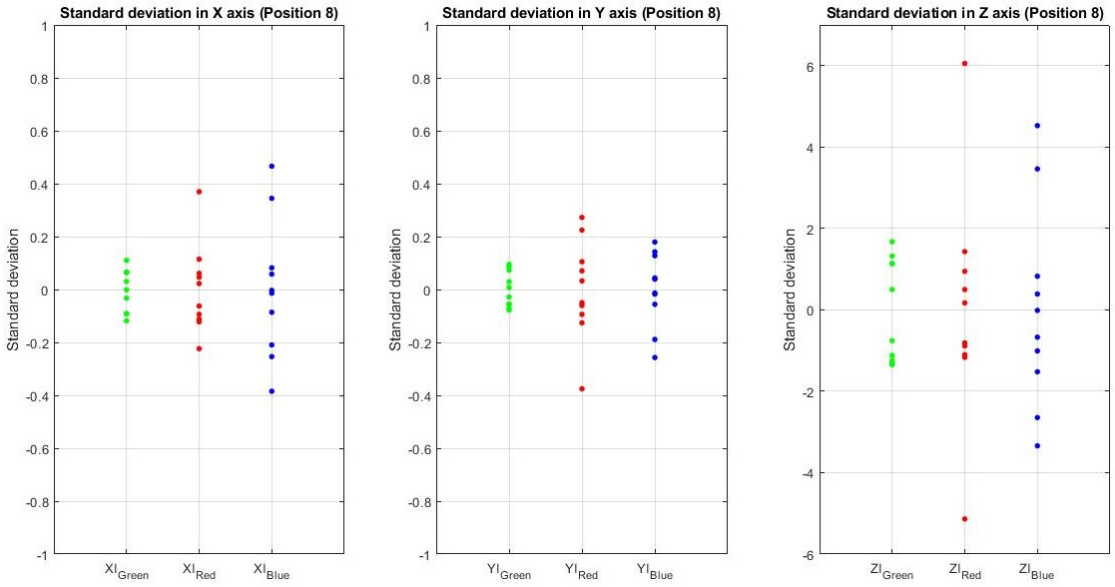

Grafico 7:Posizione8-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore

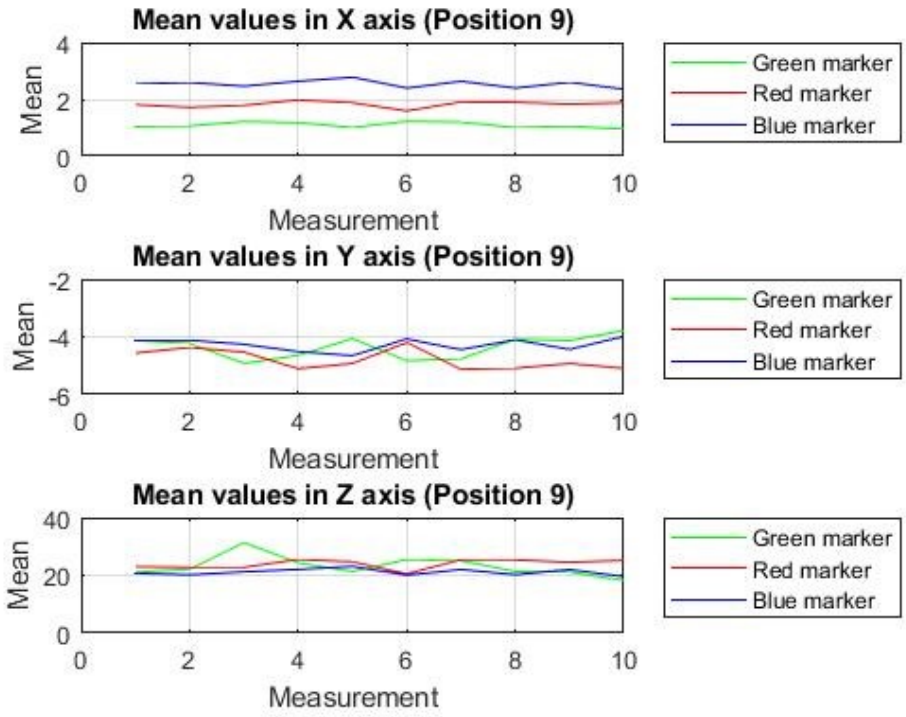

Grafico 8:Posizione9-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

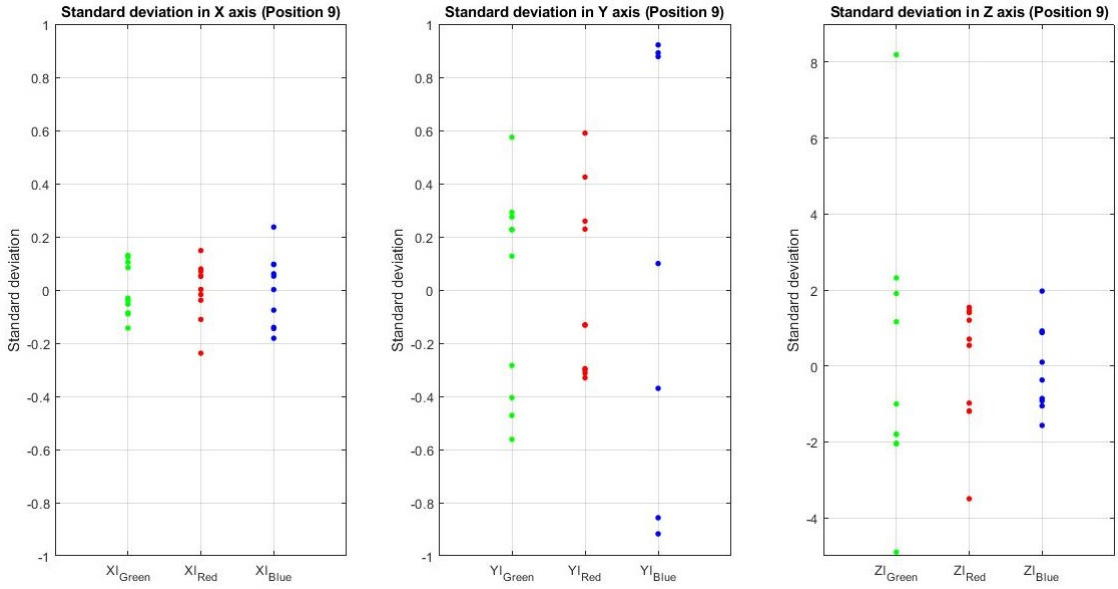

Grafico 9:Posizione9-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore

Una volta analizzati i risultati per ogni marcatore si osserva che:

- Il marcatore blu non dà esiti esaustivi in quanto si ottengono molti valori a cui la nostra applicazione assegna un Not a Number, ovvero non è in grado di assegnare alcun tipo di valore.
  Anche per questa prova di dispersione temporale a breve termine si pensa che ciò sia dovuto a problemi nella calibrazione, in quanto il colore blu essendo molto simile al colore nero, di cui sono fatti i gambi degli strumenti laparoscopici, può essere confuso dall'applicazione.
  Ciò si nota soprattutto in queste prove di precisione, in quanto l'applicazione non appena viene lanciata necessita di una serie di secondi per cercare il marcatore e acquisirne i relativi dati di posizione. Avendo però a disposizione solo quindici secondi per portare a termine la prova, l'applicazione non fa in tempo a captare il marcatore blu che già deve essere bloccata.
  Il marcatore di colore blu perciò è quello che presenta maggiori spostamenti in termini di media e deviazione standard.
  La scarsa efficacia di tale marcatore si può notare soprattutto nelle prime due file, mentre per l'ultima fila vediamo che almeno per quanto riguarda l'asse X migliora.

- Per il marcatore di colore verde dai dati risultanti notiamo una miglior rilevazione nelle ultime due file. Proprio in queste, infatti, dai grafici ricavati possiamo vedere una dispersione della deviazione standard minore soprattutto per gli assi X e Y. Tutte le deviazioni calcolate, eccetto che per gli assi Z, risultano con valori minori di circa 1 cm. Valori più piccoli è possibile trovarli lungo l'asse X. La posizione in cui ciò è più evidente è la posizione numero 5 che presenta un valore di deviazione standard minore di 0,5 cm.

- Per quanto riguarda il marcatore di colore rosso invece, si può notare che, eccetto in rare occasioni, presenta un andamento della media quasi sempre costante in tutte le posizione. Infatti, paragonato agli altri due marcatori, il rosso è costituito da minori deviazioni dalla media e da valori minori di deviazioni standard, come si può notare per la posizione 9.

In generale da tali grafici si evince che il marcatore che ha dato i migliori risultati è il marcatore di colore rosso.
La posizione in cui otteniamo valori molto buoni per tutti e tre i marcatori è la posizione numero 8, poichè è centrale rispetto alla fotocamera del dispositivo e costituisce parte dell'ultima fila che è priva di problemi di messa a fuoco.

È possibile visualizzare in fondo tutti gli altri grafici relativi alla media e alla deviazione standard risultanti delle posizioni mancanti. Da questi, notiamo che tra le altre posizioni quelle che ci hanno portato a ottenere risultati buoni sono state la numero 3, la 4 ed infine la 6, anche se per l'asse Z continuiamo ad avere valori non appaganti.
La posizione in cui invece ricaviamo dei valori che non possono essere considerati soddisfacenti è la numero 1, quella da cui otteniamo il maggior numero di Not a Number.

Si suppone che i valori anomali ottenuti per la deviazione standard rispetto all'asse Z, siano dovuti ad un problema nel filtro di Kalman, implementato nel precedente codice in C++.
Cercando di risolvere tale prolema, ci si è accorti che necessitavamo di una conoscenza più approfondita a riguardo e che il tempo a disposizione per acquisirla non era sufficiente.

Sulla base dell'analisi effettuata possiamo affermare che l'applicazione sviluppata funziona ed in generale produce risultati che possono essere considerati veritieri.
Risultati più precisi dipendono molto dal colore del marcatore, più chiaro e acceso è il colore e più l'applicazione funzionerà meglio.

5. Conclusioni e lavori futuri

In quest'ultima parte verranno presentate le conclusioni finali ricavate sulla base dei risultati ottenuti e si elencheranno i possibili lavori futuri da apportare all'applicazione al fine di renderla migliore e completa.

5.1 Conclusioni

In questo lavoro di tesi ci si era prefissati di sviluppare un applicativo per la piattaforma mobile di Android capace di eseguire il *tracking* di al massimo due strumenti laparoscopici. Per realizzare ciò, oltre all'utilizzo delle librerie OpenCV, i codici nativi in C++ a disposizione sono stati migrati su Android Studio.
L'applicazione prodotta ha raggiunto i risultati sperati.
Si è osservato come l'efficienza nell'inseguire i due strumenti laparoscopici, dipenda dalla velocità dei loro movimenti e dalla distanza presente rispetto alla fotocamera del dispositivo su cui si sta utilizzando l'applicazione.
Ciò è dimostrato dalla caduta rapida del numero di *frame* per secondo, dovuto al fatto che l'applicazione incontra maggiori difficoltà nel ricercare l'obiettivo da tracciare.
La soglia massima di distanza per cui l'applicazione produce risultati ancora soddisfacenti, calcolata sulla base delle prove effettuate, è all'incirca 20 cm. Ciò dipende anche dal colore del marcatore utilizzato, più il colore sarà vivace e più risulterà facile monitorarlo anche a distanze maggiori.
Lo smartphone utilizzato durante le prove disponeva di un sistema operativo di tipo Android 7.0, si spera perciò che con l'avanzare delle nuove tecnologie, in particolare dei nuovi smartphone, le problematiche legate al numero di *frame* per secondo possano essere superate.

5.2 Lavori futuri

Le modifiche e i miglioramenti che dovrebbero essere apportati a questo lavoro in futuro sono :

- Lo sviluppo di due supporti meccanici che in grado di mantenere in una posizione fissa qualsiasi tipo di smartphone e qualsiasi tipo di tablet, in quanto devono essere evitati possibili spostamenti indesiderati del dispositivo che si sta utilizzando durante l'uso dell'applicazione.
- Lo sviluppo di un sistema in grado di prelevare dal file .xml le posizioni 3D degli strumenti e mostrare direttamente i grafici una volta che si finisce di usare l'applicazione. Ciò sarebbe molto utile perchè non richiederebbe l'intervento di un

esperto nel prelevare i dati, inserirli su Matlab e visualizzare poi l'andamento della traiettoria dei due strumenti.

- La possibilità di salvare il video del monitoraggio degli strumenti per poterlo rivedere in un secondo momento. Questo diventerebbe utile per riguardare gli errori fatti precedentemente ed imparare da questi. Inoltre, si potrebbe utilizzare anche nella fase di valutazione dove si andrebbe a valutare non solo il risultato finale, ma anche il percorso fatto e i miglioramenti conseguiti.
- La possibilità di applicare l'algoritmo sviluppato ad un video già esistente. Si riuscirebbe in questo modo ad analizzare attività svolte in momenti precedenti e da dispositivi differenti.
- L'implementazione dell'algoritmo anche per dispositivi con sistema operativo iOS in modo da poterlo utilizzare anche su iPhone, iPod touch e iPad, così da poter essere adoperata su qualsiasi tipo di dispositivo attualmente in commercio.
- L'aggiunta di una realtà virtuale in modo che si possano visualizzare le azioni da compiere direttamente sul dispositivo, senza il bisogno di inserire all'interno del simulatore strutture in legno. Ciò renderebbe quest'applicazione più simile ad un videogioco, così da stimolare maggiormente l'interesse dei neo-chirurghi. Inoltre, non sarebbe più necessario l'uso di un simulatore ma basterebbe avere a disposizione qualsiasi tipo di contenitore rivestito di cartone di colore bianco e privo di fori così da non far entrare la luce esterna.
- L'attualizzazione e il miglioramento della versione esistente creata per i computer in modo che abbia interfacce più intuitive e più simili a quelle presenti della nostra applicazione. Inoltre, per facilitarne l'uso, bisognerebbe unire anche per tale versione, l'applicazione della calibrazione e quella del monitoraggio.
- L'ottimizzazione del codice del *tracking* in modo da poter aumentare i *frame* per secondo e per fare ciò, si potrebbe migliorare il sistema hardware del dispositivo che si sta utilizzando ovvero, aumentare la memoria RAM e la velocità del processore. In questo modo si pensa di arrivare all'incirca ad un valore di *frame* per secondo di 30. In aggiunta, si dovrebbe utilizzare la versione *release* dell'applicazione invece che la *debug* utilizzata fino ad ora.
- L'aggiunta di un video introduttivo che spieghi come deve essere svolta l'attività all'interno del simulatore e quali sono i traguardi da raggiungere. Ciò renderebbe l'applicazione più completa. Inoltre, per poter migliorare il processo di apprendimento potrebbero essere aggiunti dei feedback in tempo reale, cosicché il neo-chirurgo abbia la possibilità di correggersi al momento.

Bibliografia

[1] I. Oropesa, «Controlling virtual scenarios for minimally invasive surgery training using the EVA Tracking System,» in *XXXIII Congreso anual de la sociedad espanola de ingeneria biomedica*, Madrid, 2015.

[2] I. Oropesa, Conceptual framework for the design, implementation and validation of psychomotor skills'assessment systems in minimally invasive surgery, Madrid, 2012.

[3] V. Prandini, «Laparoscopia addominale,» [Online]. Available: http://www.poliambulanza.it/dipartimenti/dipartimento-di-chirurgia/chirurgia-generale/unita-chirurgia-laparoscopica-chirurgia. [Consultato il giorno 10 Ottobre 2017].

[4] I. Oropesa, P. Sánchez-González, M. k. Chmarra, P. Lamata, A. Fernández e J. A. Sánchez-Maragallo, «EVA: Laparoscopic Instrument Tracking Based on Endoscopic Video Analysis for Psychomotor Skills Assessment,» 6 Ottobre 2012.

[5] G. M. Fried e F. L. S, Objective assessment of technical performance, World Journal of Surgery 32, 2008.

[6] P. J. Van Empel, L. B. Van Rijssen, J. P. Commandeur, M. G. E. Verdam, J. A. Huirne, F. Scheele, J. B. H e W. Jeroen Meijerink, «Validation of a new box trainer-related tracking device: the TrEndo,» 2012.

[7] S. D. Bann, B. M, S. B, K. M. S, B. M, B. Ch, A. W. Darzi e D. M, «Measurament of Surgical Dexterity Using Motion Analysis of simple Bench Tasks,» 2003.

[8] I. Oropesa, S.-G. Patricia, P. Lamata, M. K. Chmarra, J. B. Pagador, J. A. Sánchez-Maragallo, F. M. Sánchez-Maragallo e E. J. Gómez, «Methods and Tools for Objective Assessment of psychomotor skills in laparoscopic surgery,» 2011.

[9] P. Lamata, Methodologies for the analysis, design and evaluation of laparoscopic surgical simulators, Madrid, 2006.

[10] F. P. Escamirosa, R. O. Flores e A. M. Martínez, «Construction and Validation of a Low Cost Surgical Trainer based on iPhone Technology for Training Laparoscopic Skills,» Città del Messico, 2015.

[11] E. Yiannakopoulou, N. Nikiteas, D. Perrea e C. Tsigris, «Virtual reality simulators and training in laparoscopic surgery,» 2015.

[12] A. Chaudhry, C. Sutton, J. Wood, R. Stone e R. McCloy, «Learning rate for laparoscopic surgical skills on MIST VR, a virtual reality simulator: quality of human-computer interface».

[13] «LAP Mentor Simbionix,» [Online]. Available: http://simbionix.com/simulators/lap-mentor/. [Consultato il giorno 10 Ottobre 2017].

[14] K. W. Van Dongen, E. Tournoij, D. C. Van der Zee, M. P. Schijven e I. A. M. J. Broeders, «Construct Validity of the LapSim: Can the LapSim virtual reality simulator distinguish between novices and experts?,» 2007.

[15] «CAE LapVR Laparoscopic,» 2014. [Online]. Available: https://caehealthcare.com/surgical-simulation/lapvr. [Consultato il giorno 15 Ottobre 2017].

[16] «Simendo laparoscopy,» [Online]. Available: http://www.simendo.eu/index.php/simendo-laparoscopy/. [Consultato il giorno 15 Ottobre 2017].

[17] P. Sánchez-González, I. Oropesa, F. Sánchez-Maragallo, P. Lamata e E. Gómez, A virtual reality simulator-based system for MIS objective evaluation., Madrid, 2010.

[18] A. Dosis, R. Aggarwal, F. Bello, K. Moorthy, Y. Munz, D. Gillies e A. Darzi, Synchronized Video and Motion Analysis for the Assessment of Procedures in the Operating Theater, Archives of Surgery, 2005.

[19] M. K. Chmarra, S. Klein, J. C. F. de Winter, F.-W. Jansen e J. Dankelman, «Objective classification of residents based on their psychomotor laparoscopic skills,» 14 Novembre 2009.

[20] M. Riojas, C. Feng, A. Hamilton e J. Rozenblit, «Knowledge elicitation for performance assessment in a computerized surgical training system,» 24 Febbraio 2011.

[21] «developer.android.com,» [Online]. Available: https://developer.android.com/studio/index.html. [Consultato il giorno 5 Settembre 2017].

[22] T. Bray, «What Android is,» 14 Novembre 2010. [Online]. Available: http://www.tbray.org/ongoing/When/201x/2010/11/14/What-Android-Is. [Consultato il giorno 5 Settembre 2017].

[23] «Android Open source project License,» 14 Novembre 2017. [Online]. Available: https://source.android.com/setup/licenses. [Consultato il giorno 20 Dicembre 2017].

[24] «T-Mobile Unveils the T-Mobile G1-- The first Phone Powered by Android,» 23 Settembre 2008. [Online]. Available: https://web.archive.org/web/20110712230204/http://www.htc.com/www/press.aspx?id=66338&lang=1033. [Consultato il giorno 10 Novembre 2017].

[25] «Android 1.5 Platform Highlights,» Aprile 2009. [Online]. Available: https://developer.android.com/about/versions/android-1.5-highlights.html. [Consultato il giorno 12 Ottobre 2017].

[26] «Android 1.6 Platform Highlights,» [Online]. Available: https://developer.android.com/about/versions/android-1.6-highlights.html. [Consultato il giorno 10 Novembre 2017].

[27] L. Ceridono, «Google annuncia kitkat,» 4 Settembre 2013. [Online]. Available: https://it.softonic.com/articoli/google-annuncia-kitkat. [Consultato il giorno 10 Novembre 2017].

[28] «Android Lollipop,» [Online]. Available: https://developer.android.com/about/versions/lollipop.html. [Consultato il giorno 10 Novembre 2017].

[29] «Android 6.0 Changes,» [Online]. Available: https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html. [Consultato il giorno 10 Novembre 2017].

[30] «Android 7.0 for Developers,» [Online]. Available: https://developer.android.com/about/versions/nougat/android-7.0.html). [Consultato il giorno 10 Novembre 2017].

[31] «Android 8.0 Features and APIs,» [Online]. Available: https://developer.android.com/about/versions/oreo/android-8.0.html. [Consultato il giorno 15 Dicembre 2017].

[32] M. Carli, «Android 3: Guida per lo sviluppatore,» Aprile 2011. [Online]. Available: https://books.google.it/books?id=wDUSkiIBaQC&printsec=frontcover&hl=it&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false. [Consultato il giorno 10 Novembre 2017].

[33] M. Carli, «Android 4: guida per lo sviluppatore,» 2013. [Online]. Available: https://books.google.es/books?id=GYO-AAAAQBAJ&printsec=frontcover&hl=it#v=onepage&q&f=false. [Consultato il giorno 10 Novembre 2017].

[34] G. Abatemarco, «Art sostituisce dalvik in android,» 17 Ottobre 2014. [Online]. Available: https://www.chimerarevo.com/android/art-sostituisce-dalvik-in-android-succede-alle-apps-178458/. [Consultato il giorno 10 Novembre 2017].

[35] G. Maggi, «Gli elementi e il funzionamento di base di un'applicazione,» Maggio 2014. [Online]. Available: http://www.html.it/pag/48525/gli-elementi-e-il-funzionamento-di-base-di-unapplicazione/. [Consultato il giorno 11 Novembre 2017].

[36] G. Maggi, «La prima app Android con Android Studio,» luglio 2017. [Online]. Available: http://www.html.it/pag/52901/hello-world-con-android-studio/. [Consultato il giorno 10 Settembre 2017].

[37] G. Bradsky e A. Kaehler, Learning OpenCV, vol. 1, M. Loukides, A cura di, Sebastopol, California, 2008, p. 571.

[38] J. Howse, Android Application programming with OpenCV 3, 2 a cura di, vol. 2, V. Pagare, V. Anantharaman, P. Khedekar e R. Kochery, A cura di, Birmingham, England, 2015, p. 190.

[39] K. Murphy, «so why they decide to call it Java,» 4 Ottobre 1996. [Online]. Available: https://www.javaworld.com/article/2077264/core-java/so-why-did-they-decide-to-call-it-java-.html. [Consultato il giorno 10 Settembre 2017].

[40] J. Gosling, B. Joy, G. Steele, G. Brancha e A. Buckley, The Java Language Specification, 8 a cura di, Redwood City, California, 2015, p. 788.

[41] «Write once, run anywhere?,» 27 Luglio 2009. [Online]. Available: http://www.computerweekly.com/feature/Write-once-run-anywhere. [Consultato il giorno 10 Settembre 2017].

[42] «Geometric image transformations,» [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html. [Consultato il giorno 20 Gennaio 2018].

[43] P. S. Maybeck, Stochastic models, estimation and control Volume 1, New York, 1979.

[44] E. Yuan, «Bilateral Filtering,» 5 Ottobre 2013. [Online]. Available: http://eric-yuan.me/bilateral-filtering/.

[45] A. Antonielli, «La libreria OpenCV-4-Edge detection con l'Algoritmo di Canny,» 1 Novembre 2012. [Online]. Available: http://www.allafinedelpalo.it/la-libreria-opencv-4-edge-detection-con-lalgoritmo-di-canny/. [Consultato il giorno 20 Ottore 2017].

[46] F. Becattini, «La libreria OpenCV-5- Trasformata di Hough,» 8 Novembre 2012. [Online]. Available: http://www.allafinedelpalo.it/la-libreria-opencv-5-trasformata-di-hough/. [Consultato il giorno 20 Ottobre 2017].

[47] I. Oropesa, d. J. T. L, P. Sánchez-González, J. Dankelman e E. J. Gómez, «Feasibility of tracking laparoscopic instruments in a box trainer using a Leap Motion Controller,» p. 10, 2 Dicembre 2015.

[48] G. Giordano, «C++, un'introduzione,» 19 Settembre 2016. [Online]. Available: http://www.html.it/pag/15473/introduzione17/. [Consultato il giorno 10 Settembre 2017].

Glossario Grafici

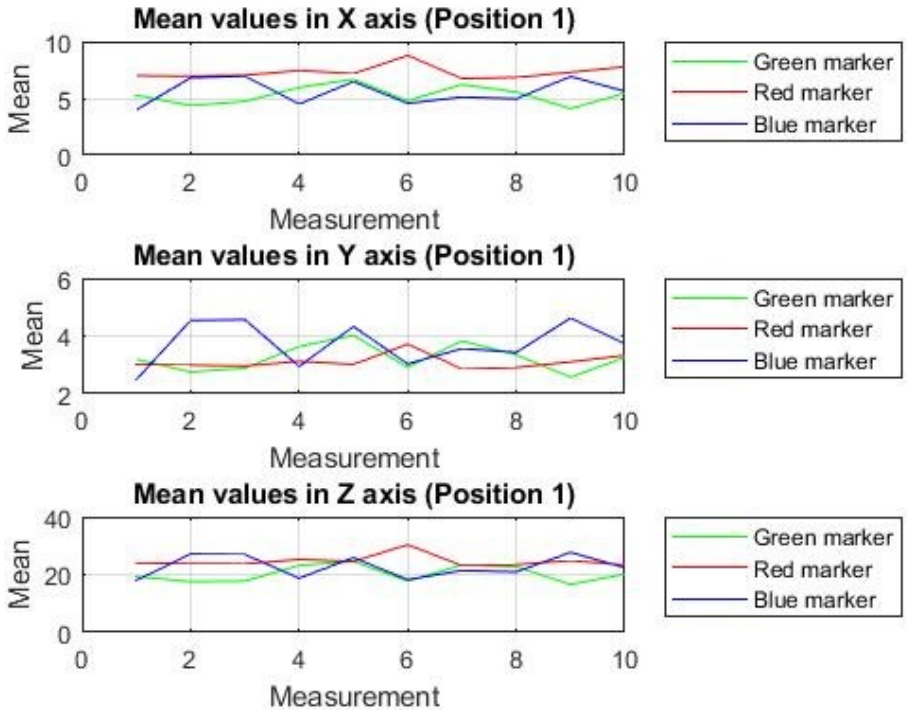

Grafico 10:Posizione1-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

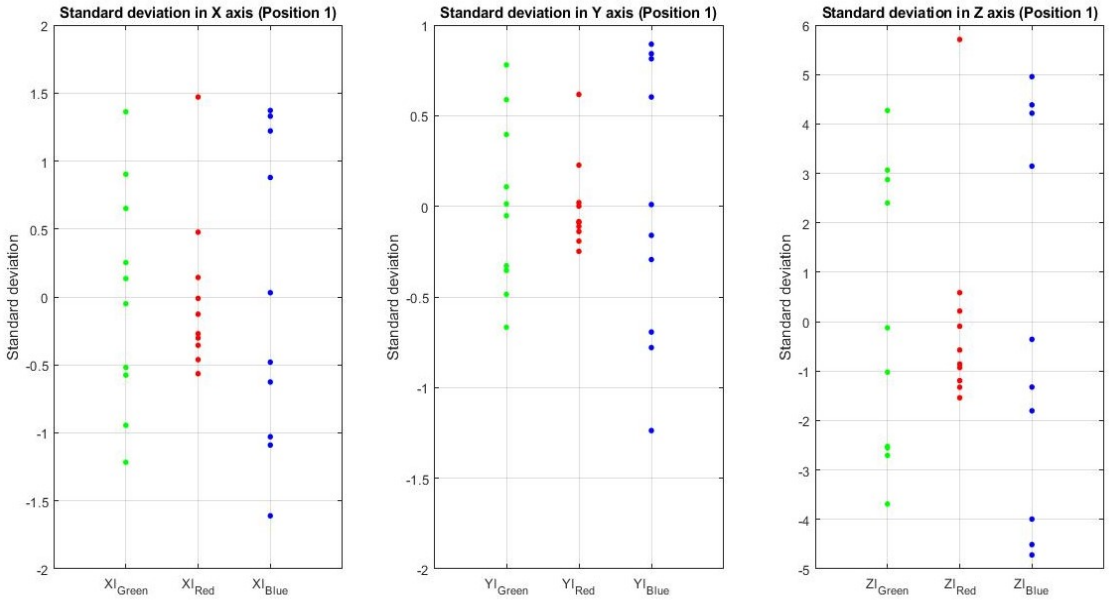

Grafico 11:Posizione1-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore

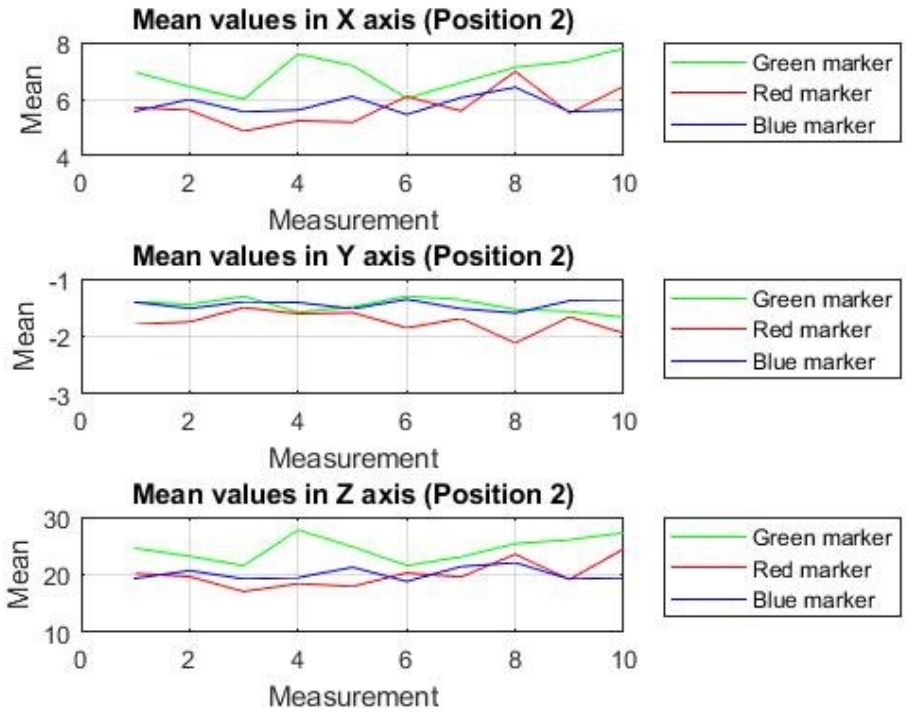

Grafico 12:Posizione2-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

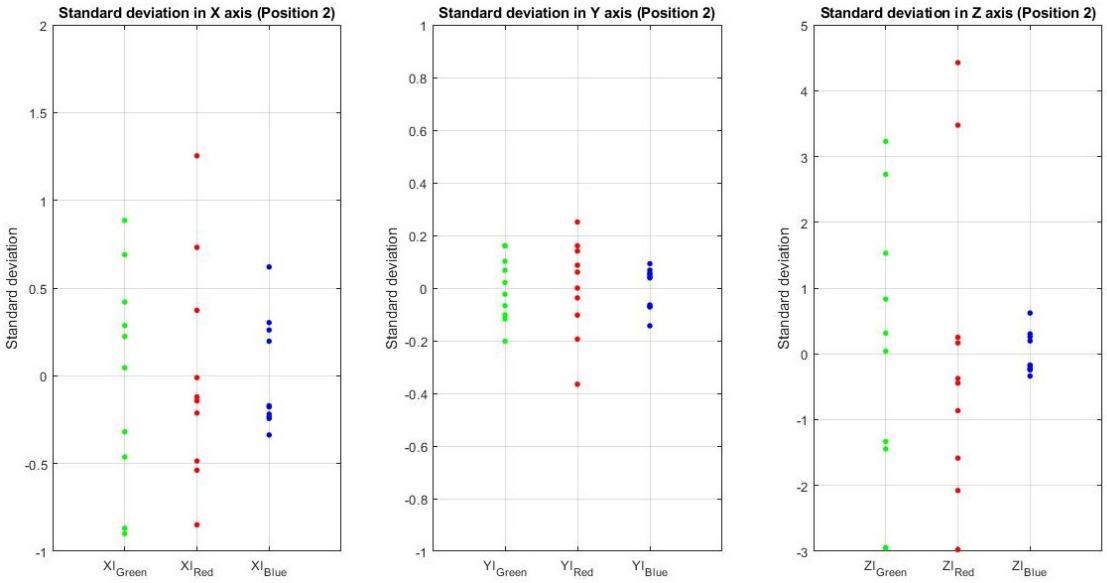

Grafico 13:Posizione2-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore

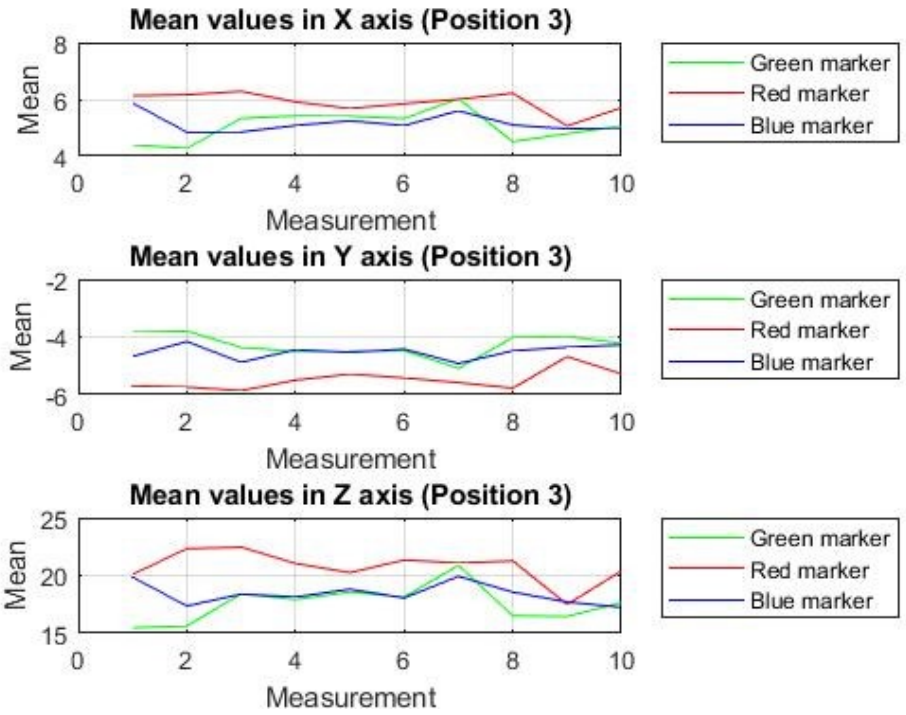

Grafico 14:Posizione3-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

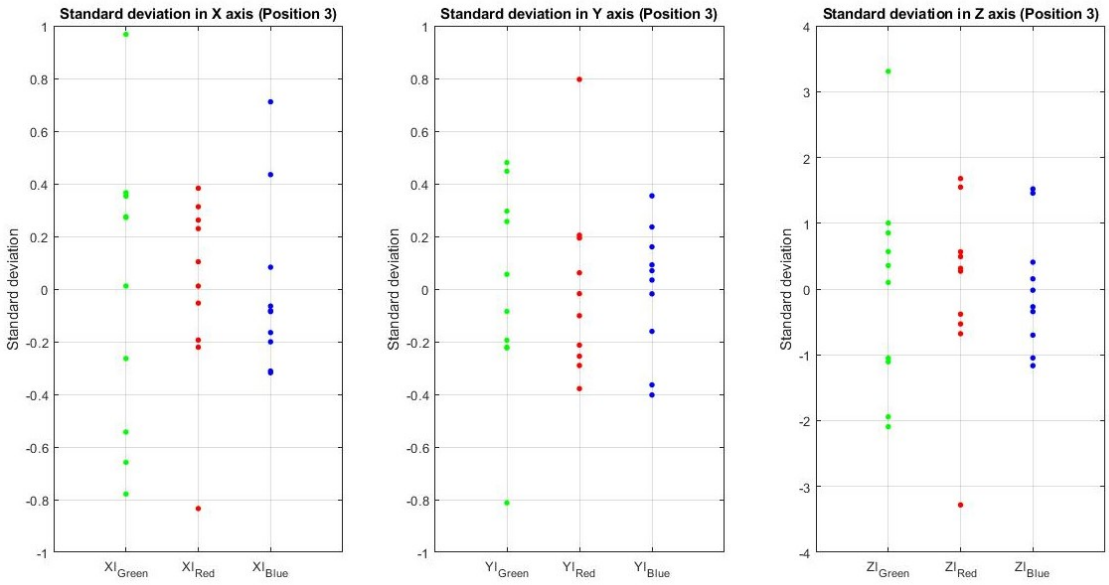

Grafico 15:Posizione3-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore

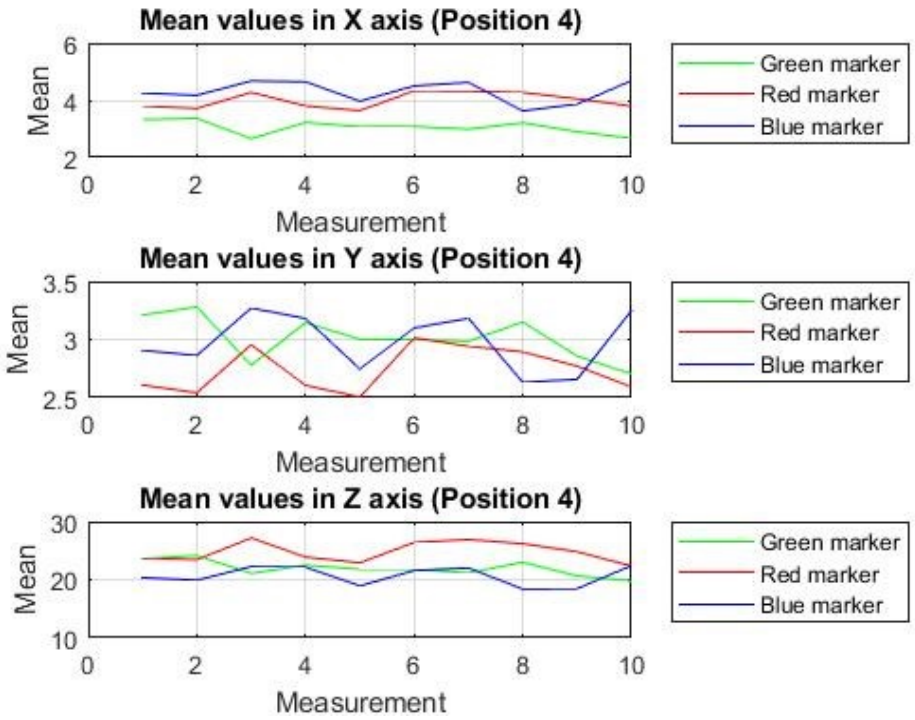

Grafico 16:Posizione4-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

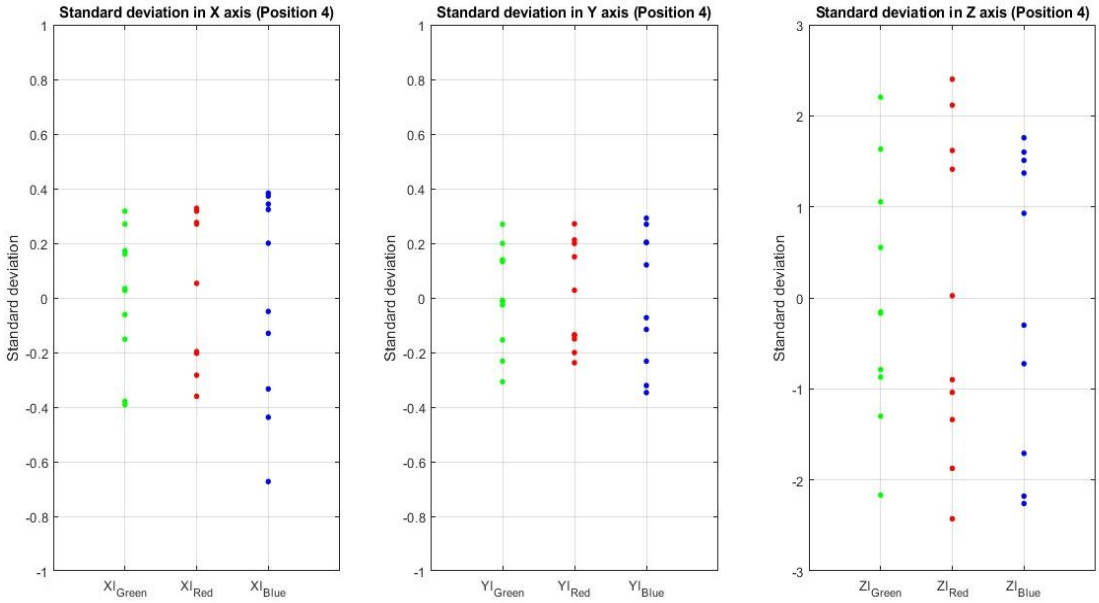

Grafico 17:Posizione4-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore

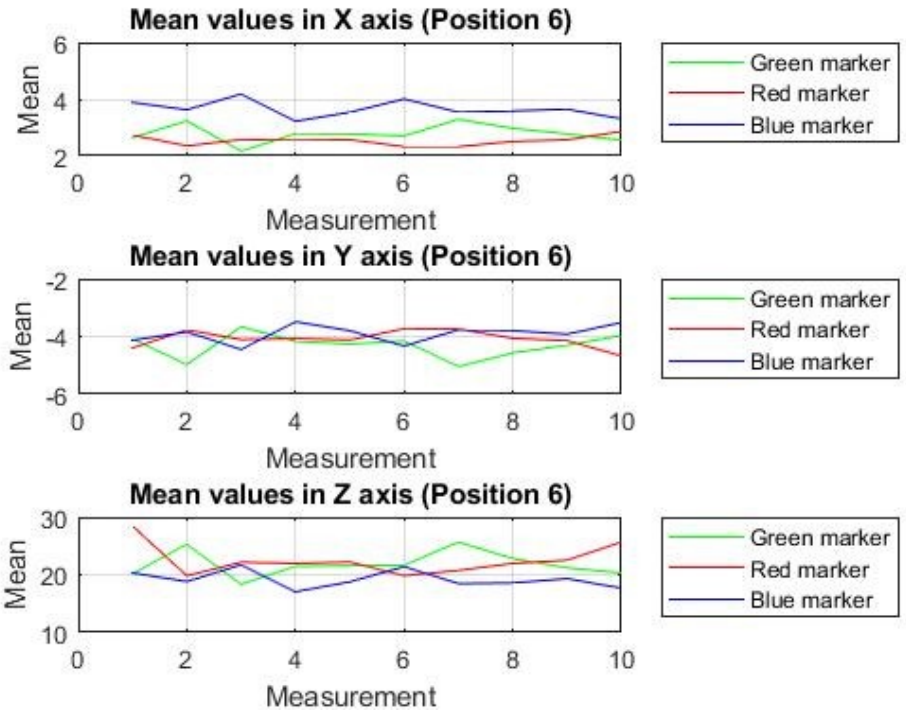

Grafico 18:Posizione6-Valori medi in centimetri per gli assi x, y e z per ciascun marcatore

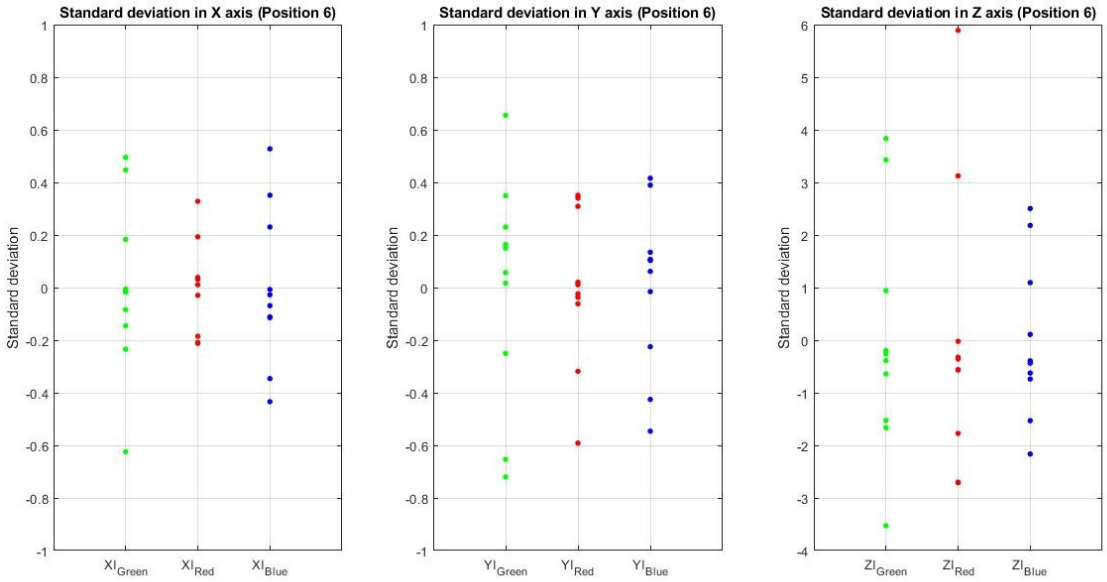

Grafico 19:Posizione6-Deviazioni standard in centimetri per gli assi x, y e z per ciascun marcatore

Glossario abbreviazioni

ABBREVIAZIONI	SIGNIFICATO	CAPITOLI IN CUI COMPAIONO
API	Android Programming Interface	Materiali
BT	Box Trainer	Stato dell'arte
CP	Coordinate Pulling	Materiali
DVM	Dalvik Virtual Machine	Materiali
ETSIT	Escuela Técnica Superiod de Ingenieros de Telecomunicaciòn	Metodi
EVA	Endoscopic Video Analysis	Introduzione
HMM	Hidden models'Markov	Stato dell'arte
HMT	Human Motion Tracking	Stato dell'arte
HSV	Hue Saturation Value	Materiali
IDE	Integrated Development enviroment	Materiali
JDK	Java Development Kit	Materiali
JNI	Java Native Interface	Materiali
JVM	Java virtual Machine	Materiali
MIS	Minimally Invasive Surgery	Introduzione
MM	Models'Markov	Stato dell'arte
NDK	Native Development Kit	Materiali
OR	Operating Room	Stato dell'arte
RGD	Red Green Blue	Materiali
SDK	System Development Kit	Materiali
VR	Virtual Reality	Stato dell'arte