STIMA DELLA TRAIETTORIA DI UNA MANO A PARTIRE DA DATI 3D

RELATORE: Ch.mo Prof. Guido Maria Cortelazzo
CORRELATORE: Ing. Carlo Dal Mutto
CONTRORELATORE: Prof. Pietro Zanuttigh
LAUREANDO: Dott. Fabio Dominio

Corso di laurea magistrale in Ingegneria Informatica

A.A. 2010-2011

UNIVERSITÀ DEGLI STUDI DI PADOVA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

TESI DI LAUREA

STIMA DELLA TRAIETTORIA DI UNA MANO A PARTIRE DA DATI 3D

RELATORE: Prof. Guido Maria Cortelazzo

CORRELATORE: Ing. Carlo Dal Mutto

CONTRORELATORE: Prof. Pietro Zanuttigh

LAUREANDO: Dott. Fabio Dominio

A.A. 2010-2011

In memoria di mio padre.

Indice

Sommario

Il lavoro descritto in questa tesi è stato svolto nell'ambito di un progetto più ampio, il cui obiettivo è la realizzazione di un'interfaccia uomo-macchina in grado di riconoscere con accuratezza i gesti di una mano compiuti da un generico utente. Le possibili applicazioni di tale interfaccia sono molteplici e non sono limitate al campo informatico, ma possono estendersi ad altri campi come la medicina. A differenza dei sistemi adottati per il *motion-capture* del corpo umano che, in genere, hanno un costo elevato e richiedono l'impiego di complessi algoritmi di analisi di dati 2D (nella fattispecie immagini o flussi video) o pseudo-3D, il sistema descritto in questa sede è progettato per elaborare unicamente dati 3D. Nell'elaborato è descritta la realizzazione del sistema sopracitato, riportando i risultati, i pregi e i difetti delle scelte effettuate. In particolare, nel Capitolo 1 è illustrato il sistema sperimentale d'acquisizione, nel Capitolo 2 è discusso il problema della segmentazione e una sua possibile risoluzione, mentre il Capitolo 3 tratta la costruzione di un *modello della mano*, necessario alla fase di riconoscimento della posa della mano descritta nel Capitolo 4. In quest'ultimo sono anche riportati i risultati di alcune verifiche sperimentali sul funzionamento dell'algoritmo di stima della posa, e i relativi grafici sono raccolti nell'Appendice. Infine, sono tratte le conclusioni. Data la complessità del progetto, il riconoscimento finale dei gesti non è contemplato nella tesi, e sarà eventualmente oggetto di un proseguimento del lavoro.

Introduzione

Dall'avvento del Personal Computer negli Anni '80 ai giorni nostri, il campo delle interfacce uomo-macchina ha subito profondi cambiamenti e inversioni di tendenza. Inizialmente la tastiera era l'unico dispositivo d'interazione con le scarne interfacce testuali offerte dai primi sistemi operativi, mentre in seguito, con l'introduzione del mouse da parte di Apple, l'evoluzione di questi ultimi accelerò sensibilmente grazie a interfacce grafiche sempre più ricche e intuitive. Per molto tempo i dispositivi sopracitati dominarono il mercato, non essendoci valide alternative per una loro sostituzione.
L'innovazione più importante fu l'apparizione nel mercato, qualche anno fa, dei primi monitor touch-screen, i quali cambiarono radicalmente il modo di interagire con le macchine. Grazie alla miniaturizzazione sempre più spinta e all'abbattimento dei costi della tecnologia, la maggior parte dei dispositivi elettronici, anche consumer, sembra vertere alle tecnologie touch. Oramai, infatti, è arduo trovare in commercio uno smartphone o un registratore di cassa sprovvisti di touch-screen, e i tastierini cedono il proprio spazio all'estensione della superficie del display. Osservando l'andamento del mercato, sembrerebbe che tra non molto mouse e tastiere diverranno obsoleti e saranno soppiantati dalle tecnologie touch.
In realtà, il notevole successo attribuito a nuovi dispositivi per l'interazione introdotti nel campo ludico, quali Microsoft Kinect™e l'emule Playstation Moove, è indice di un profondo interesse del pubblico nei riguardi di nuove tecnologie evolutivamente superiori ai touch-screen, tecnologie in grado di offrire un maggiore coinvolgimento all'utente permettendogli l'uso del proprio corpo come controller. L'interesse per Microsoft Kinect™, in particolare, ha spinto l'azienda a rilasciare alcune librerie ufficiali con cui interfacciare il dispositivo a un computer, permettendo agli sviluppatori di progettare i primi software che sostituiscano Kinect al comune mouse. Fino a non molto tempo fa, gli unici altri dispositivi per l'interazione con le console erano i joystick, la loro evoluzione seguì quella dei mouse e delle tastiere, abbandonando in certi casi l'impiego di cavi di collegamento.

Allo stesso tempo, il successo del film in stereoscopia[1] Avatar determinò la diffusione in molte sale cinematografiche di proiettori digitali e visori personali per la visione stereoscopica delle nuove pellicole, rafforzando il desiderio delle persone di un maggiore coinvolgimento che solo le "tecnologie in 3D" possono dare. Un'ulteriore conferma di quanto affermato deriva dall'introduzione nel mercato della *consumer-electronics* dei primi televisori e lettori blu-ray in grado di visualizzare filmati in 3D, nonché di schede video[2] corredate di occhialini stereoscopici attivi e di monitor con elevato framerate (almeno 120Hz) che permettono anche all"'utente medio" la fruizione di contenuti multimediali e di alcuni videogiochi a tre dimensioni.
L'uso delle nuove tecnologie per la stereoscopia non è, tuttavia, limitato al campo ludico o, più in generale, nel tempo libero. Alcuni ricercatori, infatti, sono interessati alla realizzazione di interfacce uomo-macchina tridimensionali che utilizzino la mano stessa dell'operatore come controller per la navigazione al loro interno, senza che questa sia vincolata da altri dispositivi o da sensori ausiliari al tracciamento. L'applicazione pratica più importante di tali interfacce la navigazione all'interno di un desktop tridimensionale in cui la mano assolve il ruolo di un mouse evoluto. A dire il vero, usare la mano come un semplice mouse è riduttivo: associando un'azione a un determinato *gesto* della mano, è possibile definire un numero virtualmente illimitato di azioni eseguibili, oltre a quelle canoniche (spostamento del puntatore, click, drag&drop) permesse da un mouse 2D attuale. Un'azione complessa ed eseguibile unicamente in uno spazio tridimensionale potrebbe essere, per esempio, la roto-traslazione di un oggetto *afferrato*. Un'interfaccia di questo tipo è adattabile senza modifiche sostanziali anche ad applicazioni *mediche*, come la rotazione di una TAC (Tomografia Assistita al Computer) da parte di un chirurgo senza dover toccare apparecchiature di dubbia asetticità, o potrebbe facilitare il lavoro degli operatori di un centro per la gestione delle catastrofi.
In questa tesi si vuole dimostrare come le interfacce uomo-macchina che impieghino la mano nuda come controller non siano appannaggio di film di fantascienza come Minority Report e Iron Man, ma come la loro realizzazione sia possibile in un futuro prossimo. Il cuore delle interfacce sopracitate è il *riconoscimento dei gesti della mano*, un problema ancora in fase di studio, anche se alcune aziende stanno rilasciando i primi prototipi delle loro soluzioni. Una volta risolto

[1] Non bisogna confondere *stereoscopia* con *spazio tridimensionale*: la prima è solo un'illusione ottica che permette la percezione della terza dimensione di un'immagine o un filmato che, per definizione, sono bidimensionali.

[2] I più grandi produttori sono NVIDIA™e ATI™

efficacemente il riconoscimento, la successiva costruzione di un'interfaccia grafica tridimensionale è semplificata dalle tecnologie avanzate per la 3D-vision e dall'esperienza pregressa nella realizzazione di motori grafici 3D per videogiochi.
La dimostrazione non è solo teorica: il lavoro illustrato nella tesi è stato svolto nell'ambito di un ampio progetto universitario di Computer Vision finalizzato al riconoscimento dei gesti di una mano, in vista di un possibile impiego della mano come controller di un'interfaccia tridimensionale. Data la vastità e la complessità del progetto, l'elaborato riguarda solamente la prima parte di quest'ultimo, vale a dire il riconoscimento della posa e dell'orientamento della mano in movimento.
Il problema del riconoscimento dei gesti è un problema di *classificazione* che richiede come ingresso una *stima attendibile* delle traiettorie delle componenti di interesse della mano, vale a dire le dita. Le traiettorie, a loro volta, sono stimate dalla posizione e dall"orientamento della mano in corrispondenza di ogni frame acquisito dal sistema.
Il progetto è innovativo nel suo complesso, poiché molti lavori precedenti riguardano la modellazione del corpo umano[3, 27] e non la mano nello specifico. Tali risultati non sono immediatamente applicabili alla mano; infatti, essa ha dimensioni ridotte e un'articolarità superiore a quella del corpo umano. La prima caratteristica comporta errori elevati nel caso di un'approssimazione non curata, mentre la seconda comporta un maggior numero di gradi di libertà, con un conseguente vertiginoso aumento della complessità del problema. Quasi tutte le pubblicazioni precedenti, inoltre, unite alle poche sulla modellazione della mano, impiegano complessi modelli probabilistici, classificatori o lavorano su dati bidimensionali. In particolare, fino a qualche anno fa il *motion capture* di un corpo, necessario per il tracking, era condotto unicamente tramite ingombranti tute di sensori che minavano la naturalità dei movimenti, o mediante l'applicazione di cluster di marker la cui luce riflessa era catturata da un set sperimentale costituito da diverse telecamere e i movimenti ricostruiti da complessi algoritmi di tracking. Il costo di tali sistemi e le lunghe e difficoltose procedure di calibrazione erano proibitivi al di fuori dell'ambito accademico, medico o di facoltosi enti di ricerca. Da qualche tempo alcuni ricercatori sono impegnati nello studio di sistemi d'acquisizione marker-less che, però, fanno ancora uso di telecamere per l'individuazione di alcune *features* da cui ricostruire i movimenti a partire da immagini e video. In questo progetto, invece, come descritto nel Capitolo 1, sono impiegate recenti tecnologie in grado di fornire dati tridimensionali che descrivono la realtà d'interesse, senza prima passare per una rappresentazione bidimensionale e senza l'ausilio di marker, tute di sensori e telecamere. Durante la progettazione si è

tenuto conto di quattro requisiti:

a) **Semplicità:** il sistema deve essere per quanto possibile semplice sia per aumentare l'efficienza di calcolo, sia per facilitare la comprensione della teoria sottostante. Tale semplicità è cercata anche lavorando su dati tridimensionali al posto d'immagini, raccolti tramite sensori attualmente costosi ma che in futuro si prevede saranno alla portata di tutti.

b) **Economicità:** i costi di progettazione e i costi dell'hardware devono essere, per quanto possibile, ridotti, in modo da favorire la ripetibilità dei risultati anche in altre sedi.

c) **Portabilità:** il sistema deve essere smontabile e il suo ingombro minimo, sia a causa dello spazio limitato in laboratorio, sia per poterlo eventualmente trasportare e rimontare altrove.

d) **Velocità:** il sistema deve essere in grado di elaborare i dati in *real-time* nel minor tempo possibile, in modo da catturare anche gli spostamenti rapidi della mano. Tale requisito non è, tuttavia, stringente nell'ambito della tesi; data la complessità del problema da trattare e l'inesperienza, assume maggiore importanza l'aspetto didattico che il raggiungimento degli obiettivi iniziali.

Lo schema a blocchi in Figura 1 riporta la filiera di passi da seguire per ottenere una stima della traiettoria della mano di un utente, o meglio la stima delle traiettorie delle ossa associate al palmo e alle falangi. L'affermazione è motivata all'inizio del Capitolo 3.

Figura 1: Filiera per la stima della traiettoria della mano.

La stima per entrambe le mani necessaria alla *multi-hand gesture recognition* non è contemplata in questa sede, e potrebbe essere oggetto di uno studio futuro. Alcuni blocchi, come quello riguardante la costruzione del modello, sono in realtà macro-blocchi contenenti altre filiere da seguire. Nonostante l'apparente sequenzialità indotta dagli schemi e adottata per chiarezza, alcuni passi vanno eseguiti obbligatoriamente in sequenza, mentre altri possono essere eseguiti parallelamente o con un diverso ordine. Quando ritenuto opportuno, all'inizio delle successive sezioni sarà riportato uno schema a blocchi del macro-blocco in esame.

Capitolo 1

Sistema d'acquisizione

Il primo passo affrontato nella realizzazione del sistema è stato la scelta dell'hardware; essa è di vitale importanza, in quanto influisce profondamente sul costo totale del sistema, sul tipo di dati da elaborare e la conseguente complessità dei metodi adottati nelle fasi progettuali successive. La progettazione ha seguito i passi indicati in Figura 1.1

Figura 1.1: Progettazione del sistema d'acquisizione.

Il sistema, schematizzato in Figura 1.2, è costituito solamente da un sensore Time-of-Flight di tipo Mesa SR4000[23] collegato a un computer tramite porta USB; quest'ultimo è omesso nell'immagine per semplicità rappresentativa. Benché tale sistema possa sembrare scarno e sia lecito muovere una critica nei confronti della ridotta velocità offerta dall'interfaccia USB, quando normalmente le telecamere impiegate in altri sistemi sono provviste d'interfacce Fire-Wire ad alta velocità, le prestazioni in fase d'acquisizione sono comunque elevate. Infatti, la maggior parte della computazione è assolta dal microprocessore interno al sensore, e la taglia dei dati restituiti ai software d'acquisizione è esigua: meno di un MB per frame al posto di taglie di svariati ordini di grandezza di dati RAW provenienti dalle immagini a colori.
Un altro grande vantaggio del sensore sono le sue dimensioni ridotte, che lo rendono adatto a essere trasportato o integrato in maniera pressoché trasparente in altri sistemi. Esso non è, tuttavia, privo di difetti: il suo costo elevato (nell'ordi-

ne di €7000) e l'ingente sviluppo di calore dovuto agli alti consumi energetici in fase operativa precludono momentaneamente il suo impiego in possibili interfacce commerciali per l'utente medio. Si ipotizza che, come per quasi tutti i nuovi dispositivi tecnologici, anche il suo costo sarà abbattuto entro alcuni anni; una valida alternativa, coinvolge il ben più economico (meno di €200) Microsoft Kinect™, purché si accetti una sensibile perdita di precisione. L'alternativa non è stata vagliata nel progetto, ma non si esclude che il dispositivo possa sostituire il sensore in un lavoro futuro.

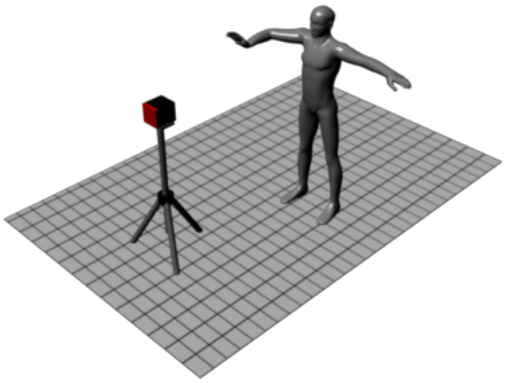


Figura 1.2: Rappresentazione del sistema d'acquisizione della mano.

Prima di riportare, per chiarezza, i principali dati di targa del sensore, è opportuno accennare brevemente al principio Time-of-Flight su cui si basa quest'ultimo. Per "Time-of-Flight", in questo ambito, si intende il tempo richiesto dagli impulsi luminosi generati da una corona di diodi led agli infrarossi, di cui è corredato il sensore, per viaggiare dalla sorgente luminosa, colpire un oggetto ed essere in parte riflessi per poi essere rilevati da un elemento fotosensibile del sensore. Immaginando, solo ai fini della semplicità, la luce come composta da un'unica sinusoide, la distanza attraversata dalla radiazione luminosa è ricavabile indirettamente dal ritardo di fase della sinusoide riflessa rispetto a quella emessa. Una descrizione approfondita di tale principio è presente nel manuale del sensore[24].

Le principali specifiche del sensore sono:

Frequenza di modulazione degli impulsi infrarossi: 30Mhz

Profondità massima rilevabile senza ambiguità: 5m

Distanza calibrata: da 0.8m a 5m

Errore assoluto: +/-10mm

Lunghezza d'onda illuminatore IR: 850nm

Framerate massimo: 50fps

Dimensioni depthmap: 176x144 pixel (QCIF)

Angoli di visione: 34.6° verticale, 43.6° orizzontale [1]

Lunghezza focale: 10mm

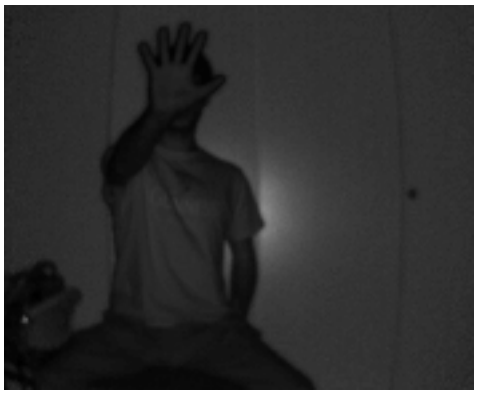
Lo SR4000 restituisce per ogni frame acquisito quattro diverse mappe:

- **Mappa di profondità:** indica, per ogni elemento fotosensibile, la profondità misurata del punto della scena colpito dalla radiazione luminosa. A dire il vero, la profondità è riferita a un *volumetto* della scena associato all'elemento fotosensibile, ma per semplicità spesso si confonde il volume con un punto 3D.
- **Mappa d'intensità:** indica, per ogni elemento fotosensibile, il valore d'intensità della radiazione luminosa riflessa dalla scena e assorbita dallo stesso in un certo tempo. Quando l'intensità è eccessiva, l'elemento *satura* e la misura di profondità non è più valida.
- **Mappa di confidenza:** riporta, per ogni elemento fotosensibile, un valore proporzionale alla probabilità che la sua misura di profondità sia corretta. Il valore dipende dalla mappa d'intensità del frame corrente e da quella del frame immediatamente precedente, e aiuta a decidere se conviene accettare o rifiutare la misura.

[1] Da notare che il campo visivo è un tronco di cono. Per chiarimenti si rimanda al manuale del sensore[24].

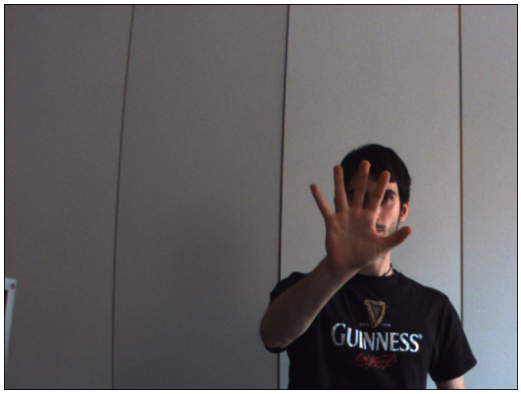
Un esempio di mappe restituite dal sensore e rappresentate come immagini, e di un fotogramma di riferimento prelevato tramite una telecamera posta in prossimità dello stesso, è riportato in Figura 1.3. Opzionalmente, per quanto riguarda la mappa di profondità, lo SR4000 è in grado di restituire anche la misura delle due rimanenti dimensioni; in questo modo la descrizione geometrica tridimensionale della scena ripresa è completa e non richiede la ricostruzione stereoscopica a partire da immagini. La "doppia mano" in Figura 1.3.c è dovuta probabilmente a un ridotto framerate in fase d'acquisizione, e la mancata sovrapponibilità del fotogramma di riferimento con le immagini precedenti è riconducibile al differente orientamento e ai diversi parametri intrinseci della telecamera rispetto all'ottica dello SR4000.

a) mappa di profondità B/W



b) mappa d'intensità B/W

c) mappa di confidenza



d) fotogramma di riferimento

Figura 1.3: Esempio di mappe restituite dallo SR4000.

La Figura 1.4, invece, riporta il sistema di riferimento secondo cui sono espresse le misure delle distanze. La specifica è essenziale nell'interpretazione dei dati acquisiti.

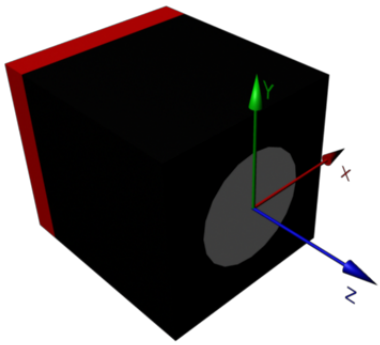


Figura 1.4: Sistema di riferimento dello SR 4000.

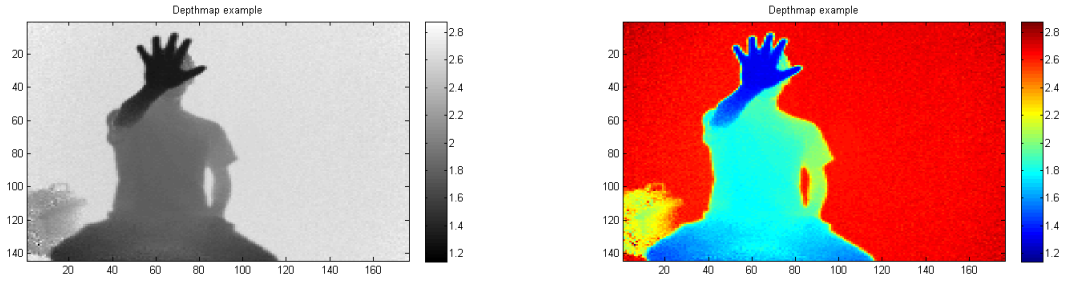
Infine, un altro contributo a una corretta interpretazione dei dati, è la specifica del sistema di riferimento del piano immagine del sensore con coordinate in pixel riportata in Figura 1.5.

a) Matlab/SR4000 b) openCV/openGL

Figura 1.5: Confronto sistemi di riferimento dell'immagine.

In particolare, la specifica del sistema di riferimento dell'immagine permette di capire l'ordinamento con cui sono fornite le misure delle distanze, intensità o confidenze, in modo da evitare ribaltamenti verticali o orizzontali nella visualizzazione delle mappe in software come Matlab, ampiamente impiegato nel progetto nello studio dei risultati intermedi. I pixel delle immagini in Matlab hanno, infatti indici crescenti a partire dall'angolo superiore sinistro dell'immagine. Bisogna prestare particolare attenzione durante l'importazione delle mappe di profondità dallo SR4000 e l'esportazione in un file dati o in un'immagine: importando la

mappa in un array bidimensionale e associando un pixel a ogni locazione di memoria, i suoi indici di riga e colonna sono compatibili con il sistema di riferimento delle immagini in Matlab. Le depthmap ottenute dallo SR sono, quindi, visualizzabili immediatamente come immagini all'interno del software, e la prova è la Figura 1.6.



a) immagine B/W b) immagine a colori

Figura 1.6: Esempio di depthmap importata in Matlab con due diverse scale di colore.

In openCV e openGL, invece, il riferimento dell'immagine è posto nell'angolo inferiore sinistro, perciò, senza ulteriori accorgimenti, una depthmap importata come immagine tramite le due librerie ai fini di successive elaborazioni appare rovesciata verticalmente perdendo, così, il suo contenuto informativo. Tuttavia, il problema è facilmente risolvibile applicando l'Equazione 1.1 L'invertibilità delle due equazioni permette di trasformare senza ambiguità le coordinate in pixel da un sistema di riferimento all'altro.

$$\begin{gathered} u_{Matlab} = u_{openGL/CV} + 1 \\ v_{Matlab} = 176 - v_{openGL/CV} + 1 \\ \text{con } (v_{Matlab}, u_{Matlab}) \text{ coordinate dei pixel nelle immagini in Matlab e} \\ (v_{openGL/CV}, u_{openGL/CV}) \text{ coordinate dei pixel nelle immagini in openGL/CV.} \end{gathered} \tag{1.1}$$

I dati sono acquisibili tramite i software demo in corredo al sensore o tramite un programma sviluppato internamente in LTTM, grazie alla fornitura di un'interfaccia con lo SR4000 offerta da MESA Imaging. Entrambi i software restituiscono le mappe di profondità di un numero prefissato di frame, ma al momento la prima soluzione è preferibile, benché il programma LTTM metta a disposizione maggiori funzionalità.

1.1 Modello del sensore

Le specifiche del sensore, di per sé, non sono sufficienti a caratterizzare il dispositivo e a estrarre informazione utile dalle mappe di profondità; è necessario, pertanto, definire un *modello del sensore* che ne descriva il funzionamento. Il modello, inoltre, come sarà accennato a inizio Capitolo 3 e ripreso nel Capitolo 4, è indispensabile ai fini della stima della posa della mano. Il modello del sensore adottato, tipico in Computer Vision, è il modello *pin-hole* di una telecamera o di una fotocamera ideale. La sua specifica coinvolge diversi sistemi di riferimento:

- sistemi di riferimento locali agli oggetti (local-space)
- sistema di riferimento globale (world-space)
- sistema di riferimento centrato sulla telecamera (camera-space)
- sistema di riferimento del piano immagine (metrico)
- sistema di riferimento dell'immagine (pixel)

Per la teoria sottostante, si rimanda alla letteratura citata[7, 10]. Dato che la definizione dei vari sistemi non segue regole precise, si riportano ai fini di una maggiore chiarezza uno schema del modello in Figura 1.7 e la struttura delle matrici di trasformazione da un sistema di riferimento all'altro nella Formula 1.2.

In questo progetto, per uniformità con le specifiche dello SR4000, si è deciso di impostare il sistema di riferimento della telecamera con una terna destrorsa avente l'asse Z uscente dal piano immagine e l'asse Y rivolto verso l'alto, come in Figura 1.4. Inoltre, avendo fatto coincidere il sistema globale con il sistema centrato sulla telecamera, non è stato necessario valutare i parametri estrinseci del sensore; equivalentemente, la matrice dei *parametri estrinseci* della telecamera coincide in questo caso con la matrice identità.

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad P = P_{openCV} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

dove f_x, f_y sono le distanze focali espresse in pixel e $[c_x, c_y]$ la posizione in pixel del *punto principale* nel sistema di riferimento dell'immagine.

Da notare che la *matrice di proiezione* non è una matrice 3x3 usata per le trasformazioni affini in $\mathbb{R}^3$. La proiezione non è, infatti, una trasformazione affine, né tantomeno lineare, e per essere espressa per via matriciale richiede il passaggio agli spazi proiettivi in cui vigono le *coordinate omogenee*[12]. Queste ultime sono ampiamente utilizzate anche nel codice non solo per via delle proiezioni, ma per la compattezza notazionale che esse introducono: ogni trasformazione affine in $\mathbb{R}^3$ che, normalmente, è descritta da prodotti di matrici 3x3 e somme di vettori, è esprimibile in $\mathbb{R}^4$ tramite una singola matrice 4x4.

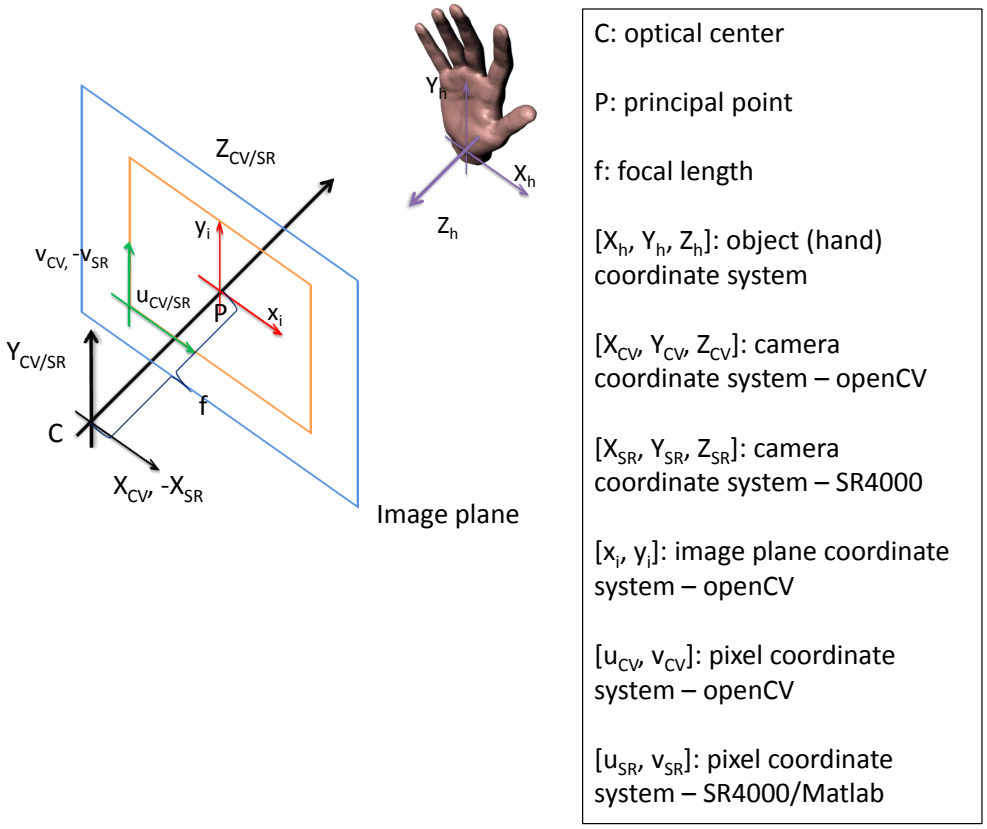


Figura 1.7: Sistemi di riferimento nel modello pin-hole.

1.2 Calibrazione del sistema d'acquisizione

Il modello pinhole è indubbiamente la descrizione più semplice del principio del funzionamento di una telecamera ma, essendo ideale, non è in grado di modellare correttamente una telecamera reale. Esso, infatti, assume che ogni punto dell'immagine sia sempre a *fuoco*, il *piano immagine* abbia estensione infinita e le sue coordinate assumano valori continui. Lo SR4000, in realtà, condividendo parte delle componenti di cui è costituito con l'hardware delle ordinarie telecamere, ha un cono visuale limitato e un numero discreto e finito di elementi fotosensibili. Inoltre, il sensore è affetto dagli stessi problemi di cui sono affetti i normali obiettivi: le lenti per correggere la messa a fuoco sono affette da distorsioni radiali, tangenziali e aberrazioni cromatiche, anche nelle apparecchiature di elevata qualità. Senza ulteriori accorgimenti, le misure delle distanze effettuate dal sensore non rispecchiano le distanze reali, e la stima della posa non risulta attendibile.
L'*accuratezza* di 1cm citata dai costruttori del sensore, ovvero che le misure riportate dallo SR4000 sono considerate attendibili entro un errore di un centimetro, non è un dato accettabile come *upper bound* dell'errore di misura. Va seguita, invece, una lunga e severa procedura di calibrazione che, oltre a restituire i *parametri intrinseci* del sensore, restituisce i parametri di correzione della distorsione.
La calibrazione del sensore è stata eseguita tramite un tool software sviluppato internamente in LTTM che funge da interfaccia con le librerie openCV. Le librerie comprendono alcuni metodi preposti alla calibrazione di una qualsiasi telecamera o fotocamera e al calcolo dei valori dei parametri per la correzione della distorsione, ma non sono direttamente applicabili a una telecamera Time-of-Flight. Una procedura di calibrazione ad-hoc è descritta in [25].
I parametri intrinseci ed estrinseci, unitamente ai dati di targa e alla specifica precedente dei sistemi di riferimento, sono sufficienti a definire ai fini del progetto un "buon" modello del sensore. In altri casi, si rende opportuna la definizione anche di un *modello di diffusione dell'errore*.
La calibrazione è ripresa nella prima parte del Capitolo 4, mentre si riporta a fine Sezione la Formula 1.3 che indica come calcolare le coordinate in pixel antidistorte.

$$u_{undist} = u_{dist}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 u_{dist} v_{dist} + p_2(r^2 + 2u_{dist}^2)$$
$$v_{undist} = v_{dist}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y_{dist}^2) + 2p_2 u_{dist} v_{dist}$$

con (v_{dist}, u_{dist}) e (v_{undist}, u_{undist}) coordinate di un pixel dell'immagine distorta e antidistorta rispettivamente, $r^2 = u_{distorted}^2 + v_{distorted}^2$, k_1, k_2, k_3 coefficienti di distorsione radiale e p_1, p_2 coefficienti di distorsione tangenziale. (1.3)

Esaminando la Formula 1.3, emerge un altro problema: le coordinate in pixel sono, per definizione, intere, mentre le coordinate antidistorte possono non esserlo, e la mappa di antidistorsione non è solitamente biiettiva[2]. Può, infatti, succedere che alcuni pixel dell'immagine distorta siano mappati nello stesso pixel dell'immagine antidistorta e altri non siano mappati in alcun pixel. Prendendo in esame un pixel dell'immagine distorta le cui coordinate antidistorte sono interne a un reticolo di 4 pixel in quest'ultima, uno dei metodi offerti da openCV per l'antidistorsione risolve il problema applicando un'interpolazione a scelta tra *nearest neighbour, bilineare o bicubica*.

[2]Si intende il fatto che non esiste in genere una corrispondenza biunivoca tra i pixel dell'immagine distorta e quelli dell'immagine antidistorta

Capitolo 2

Rilevamento e segmentazione della mano

Il sistema descritto nel Capitolo 1 è progettato per fornire un flusso di depthmap con un framerate elevato[1], e in seguito alla calibrazione la loro distorsione può essere efficacemente corretta in modo da ottenere misure di profondità attendibili. Solitamente, la scena descritta da una depthmap è popolata da molteplici oggetti, oltre alla mano di cui si vuole stimarne la traiettoria. La loro presenza è fonte di disturbo, se non di errore, in particolar modo quando sono vicini alla mano o la occludono, anche parzialmente . In una depthmap, quindi, la profondità memorizzata in alcuni pixel può non essere riferita alla mano. Senza un adeguato filtraggio che isoli i pixel della mano da quelli che non le appartengono, la stima della traiettoria può essere compromessa dagli oggetti della scena.
Il rilevamento e la segmentazione della mano sono due istanze dello stesso problema sopra scritto: il rilevamento si occupa di individuare nella scena un'area, se si hanno a disposizione dati bidimensionali (immagini, video), o un volume, se si dispone di dati tridimensionali (es. depthmap), in cui la mano è presente con un'elevata probabilità. La segmentazione, invece, si occupa di isolare i pixel della depthmap associati alla mano da quelli che non le appartengono, a partire da una prima "scrematura" effettuata dal rilevamento. Per quanto riguarda il rilevamento, nel caso di dati bidimensionali in genere si accetta un *bounding box* rettangolare, mentre nel caso di dati tridimensionali esso viene esteso a un parallelepipedo.

[1] A patto di usare il software demo fornito assieme al sensore

Il rilevamento è importante per altri due motivi:

1. la posizione del bounding box dà una buona stima iniziale per la successiva e precisa segmentazione;
2. la restrizione della regione di interesse analizzata spesso riduce sensibilmente il carico computazionale nelle fasi successive.

Fino a poco tempo fa, quando i sensori Time-of-Flight non erano ancora stati inventati, gli unici dati disponibili erano le immagini o i video acquisiti tramite telecamere, o a volte anche alcune informazioni sulla geometria sintetizzate tramite complicati algoritmi di ricostruzione stereoscopica. In letteratura sono descritti diversi tipi di rilevatori:

- rilevatori basati solo sul colore
- rilevatori basati sul movimento
- rilevatori basati sulle silhouette
- rilevatori ibridi

I primi discriminano i pixel della mano da quelli dello sfondo basandosi solamente sul loro colore; i pixel il cui colore è incluso in un range di colori associati alla pelle sono considerati interni alla mano, mentre tutti gli altri sono considerati appartenenti allo sfondo. Il rilevamento basato su questa euristica è complesso e fortemente soggetto a errori; per esempio, il colore della pelle differisce sia tra i diversi gruppi etnici, sia all'interno dello stesso gruppo, e allo stesso modo non è uniforme anche tra i pixel della mano di uno stesso individuo. Inoltre, i colori sono influenzati dall'illuminazione della scena, e quest'ultima può contenere altre parti del corpo denudate (es. le braccia e il volto) che possono "confondere" l'algoritmo di rilevazione. In alcuni casi più complessi, la rilevazione è compromessa da altri oggetti della scena che condividono i propri colori con quelli della pelle. I rilevatori basati sulle silhouette, invece, confrontano la forma degli oggetti con le possibili forme associate a una mano, scartando la regione in esame se la sua forma non è compatibile con quella di una mano. I rilevatori ibridi combinano sapientemente le informazioni sul colore con informazioni sulla forma o sul movimento, ottenendo un'accuratezza e prestazioni superiori.

La segmentazione, come accennato a inizio capitolo, consiste nel raffinare il rilevamento scartando i dati rimanenti che non riguardano la mano. Analogamente al rilevamento, in letteratura sono descritti vari algoritmi di segmentazione:

- segmentazione basata solo sul colore
- segmentazione basata sui bordi
- segmentazione basata sulla connettività, in termini di grafi, dei pixel con le informazioni sulla geometria della scena
- segmentazione ibrida

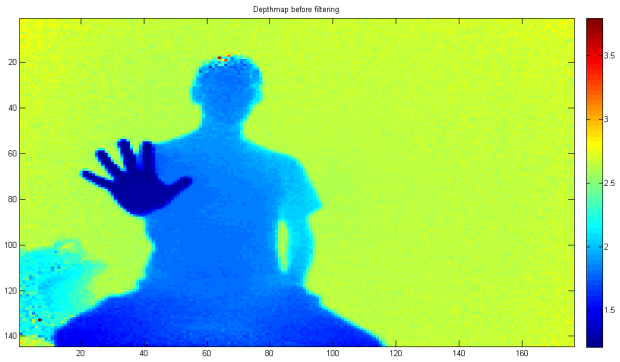
Oltre a ridurre, se non annullare, l'interferenza degli altri oggetti della scena, la segmentazione ha il pregio di favorire la ricerca di un "buon" punto da cui iniziare la stima della traiettoria. Tale affermazione è motivata nella Sezione 4.4. Avendo a disposizione dati 3D e a patto che la mano sia l'oggetto più vicino all'obiettivo (secondo il sistema di riferimento del sensore), il rilevamento e la segmentazione sono riconducibili a un semplice e immediato filtraggio sulla profondità, un'operazione che è eseguibile in due modi:

a) **Filtraggio statico:** eseguito direttamente dal sensore impostando due soglie, ovvero la profondità minima e quella massima;

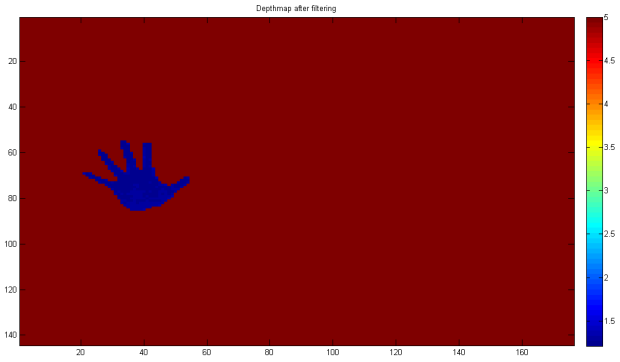
b) **Filtraggio dinamico:** eseguito per via software, impostando la differenza tra la soglia di massima profondità e quella di minima profondità.

Nel filtraggio statico, tutti gli oggetti al di fuori delle due soglie sono considerati fuori campo e i pixel della depthmap a essi riferiti sono impostati alla massima profondità. Una soglia immutabile ha il vantaggio di dover essere impostata una sola volta, ma obbliga gli oggetti o la mano a non superarla; è un limite spesso stringente. Alternativamente, il filtraggio dinamico calcola di volta in volta due soglie adattive con distanza relativa fissa, un'operazione che può essere maggiormente onerosa computazionalmente ma che non vincola la posizione della mano all'interno di un dato volume.
Quando la mano, invece, non è l'oggetto più vicino all'obiettivo ed eventualmente condivide il proprio range di profondità con quello di altri oggetti, o è occlusa parzialmente da essi, il puro filtraggio non è sufficiente a discriminare la mano dallo sfondo. In questo caso si rendono necessarie altre informazioni di supporto, come il colore, che aiutano il rilevamento a spese di una riduzione delle prestazioni dell'algoritmo.

In Figura 2.1 è riportato un esempio di rilevamento e segmentazione tramite puro filtraggio impostando le soglie di profondità minima e massima staticamente, comprensivo di fotogramma di riferimento.



a) depthmap prima del filtraggio



b) depthmap dopo il filtraggio

c) fotogramma di riferimento

Figura 2.1: Esempio di rilevamento e segmentazione tramite filtraggio.

Capitolo 3

Modello della mano

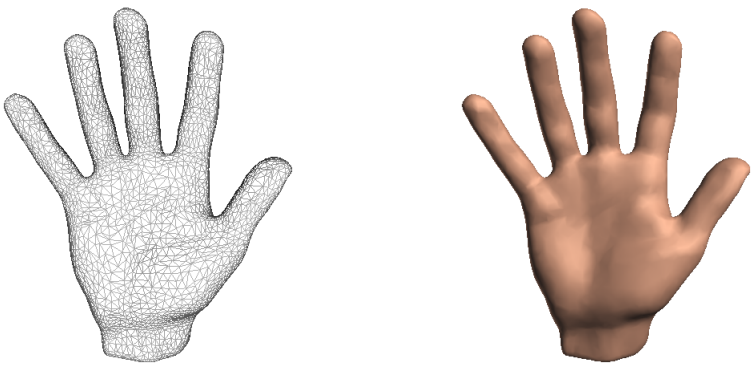
Nel Capitolo 1 è stato descritto un possibile sistema sperimentale d'acquisizione dei movimenti della mano, specificando il tipo di dati restituiti e la modalità con cui assicurarsi che questi siano attendibili, nonché una procedura per correggere la distorsione introdotta dall'ottica del sensore. Nel Capitolo 2, invece, è stato proposto un modo efficiente per epurare i dati dal rumore introdotto dagli altri oggetti presenti nella scena. Pertanto, da qui in poi, sono considerate solamente le depthmap risultanti dalla fase di segmentazione, e conseguentemente i dati elaborati sono riferiti unicamente alla mano.
Partendo da questo presupposto, l'idea alla base della stima della posa della mano in un frame è confrontare la depthmap acquisita dallo SR4000 con una depthmap *sintetica* generata da un sensore Time-of-Flight *virtuale* avente le stesse caratteristiche dello SR4000. Il sensore virtuale non misura, ovviamente, le profondità dei punti della mano dell'utente, ma le profondità dei punti di una mano sintetica che descriva al meglio quest'ultima. Più specificatamente, si tratta di definire un *modello deformabile* della mano in grado di riprodurne in maniera accettabile i movimenti e le deformazioni della pelle. Per "accettabile" si intende che il grado di realismo delle deformazioni deve essere sufficientemente elevato ai fini della successiva stima della posa.
Se, ipoteticamente, il modello definito fosse una riproduzione *perfetta* della mano reale, e il sensore virtuale avesse lo "stesso comportamento" dello SR4000, il primo obiettivo da raggiungere sarebbe spostare e deformare il modello affinché la depthmap reale e quella sintetica coincidano. In queste condizioni, infatti, il modello è posizionato e orientato allo stesso modo della mano dell'utente. Non è, tuttavia, sufficiente: per ottenere la stima della posa, è indispensabile che dal modello deformato siano estraibili i valori assunti dai parametri che descrivono la sua posa assunta nell'istante di acquisizione del frame in esame.

Il modello proposto nel progetto è analogo al modello impiegato nelle tecniche di *skinning* adottate in Computer Graphics, ed è composto da:

- una *mesh* rappresentante il soggetto da animare;
- uno *scheletro* associato al soggetto nella sua "posa base";
- una serie di *pesi* che quantifichino l'influenza delle componenti dello scheletro sui vertici della mesh;
- un *algoritmo* per calcolare la deformazione dei vertici della mesh nello spazio a seconda della particolare posa dello scheletro assegnato.

3.1 Ricostruzione della mano

Informalmente, in questo ambito per "mesh" si intende una superficie chiusa e connessa costituita da punti nello spazio euclideo, detti *vertici*, e da triangoli, detti *facce*[1], con il vincolo che ogni lato sia condiviso da al più due facce e che queste non si intersechino. A seconda della presenza o dell'assenza di uno *shading*, è possibile evidenziare la struttura della mesh (wireframe) o il suo aspetto in presenza di una fonte di illuminazione. Un esempio di mesh è riportato in Figura 3.1.



a) mesh in wireframe b) mesh con skin-shading

Figura 3.1: Esempio di mesh.

[1] Al posto dei triangoli è possibile impiegare altri poligoni, solitamente quadrilateri.

A differenza di quanto avviene spesso in animazione, la superficie della mano[2] non viene sintetizzata da zero, ma ricostruita a partire da dati tridimensionali ottenuti dalla scansione di una mano reale tramite un *laser scanner*. La Figura 3.2 riassume la sequenza dei passi da seguire per il raggiungimento dello scopo.

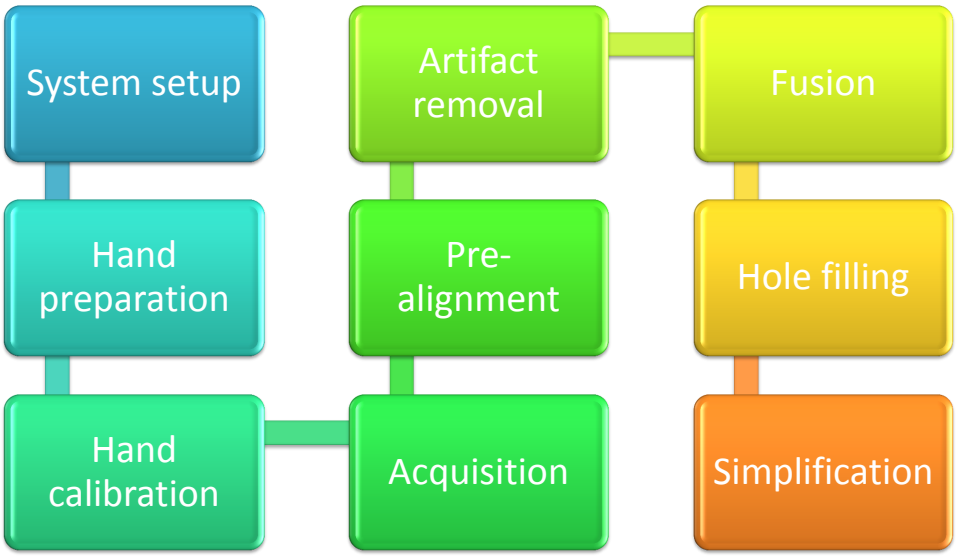


Figura 3.2: Filiera per la ricostruzione della superficie della mano.

3.1.1 Sistema di ricostruzione della mesh

A causa dello spazio limitato in laboratorio e non avendo a disposizione il costoso full body scanner usato dagli autori di SCAPE[3] per la generazione di una mesh completa (o quasi) in un unico passaggio, si è resa necessaria la realizzazione di un framework di acquisizione portatile a (relativamente) basso costo. Il sistema, schematizzato in Figura 3.3, è basato su NextEngine™3D, un laser scanner di costo inferiore ai $3000 adottato da molte industrie nella progettazione di prototipi, dagli esperti di animazione e nel campo medico delle protesi. Il corredo di NextEngine comprende una base rotante per la ricostruzione automatica della superficie di piccoli oggetti e il software d'acquisizione ScanStudio, l'unico in grado di interfacciarsi allo scanner.

[2]Da qui in poi, per brevità, sarà usato il termine *mesh*

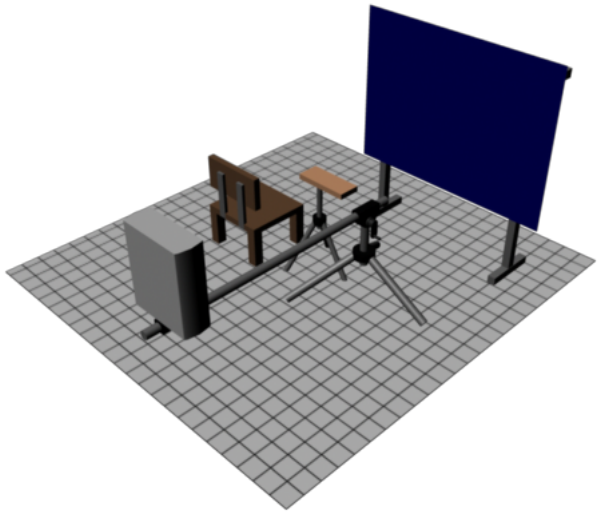


Figura 3.3: Rappresentazione del sistema d'acquisizione della mano.

I due dati di targa di interesse sono il campo di visibilità (WxH) dello scanner e la relativa precisione in base alla modalità operativa selezionata:

- 7.62x12.7cm in modalità *macro* per un oggetto posizionato a 16.51cm dallo scanner, con una precisione di 0.127mm;
- 25.4x33.02cm in modalità *wide* per un oggetto posizionato a 43.18cm dallo scanner, con una precisione di 0.381mm;
- 40.64x55.88cm in modalità *extended* per un oggetto posizionato fino a 76.2cm dallo scanner, con una precisione maggiore di 0.381mm.

NextEngine non è, però, un full body scanner; pertanto, anche se corredato di una base rotante per la scansione automatica della superficie piccoli oggetti, per ricostruire una superficie completa il dispositivo deve eseguire alcune scansioni parziali dell'oggetto ad angoli diversi e *fonderle* tramite un opportuno algoritmo. Nel caso non fosse possibile usare la base rotante, l'utente deve occuparsi del riposizionamento dell'oggetto a ogni singola scansione e fondere le viste parziali manualmente tramite un tool offerto dal software di acquisizione. La modellazione della mano rientra nel secondo caso, dato che non è possibile isolarla dal corpo del volontario per porla sulla base e quest'ultima non è in grado di sostenere il peso di un uomo.

Tuttavia, grazie alle ridotte dimensioni dello scanner (22.352x27.686x9.144cm WxHxD), l'idea adottata non è ruotare la mano o l'individuo mantenendo il dispositivo in una posizione fissa, bensì immobilizzare la mano e far ruotare lo scanner attorno a essa. Allo scopo, è stato progettato e costruito un rudimentale ma economico supporto rotante per lo scanner, costituito da un treppiede fotografico al quale è stata agganciata una testa orbitale su cui è stato fissato un leggero braccio in alluminio terminato da una base per macchine fotografiche. Quest'ultima serve per agganciare e sganciare rapidamente lo scanner per il trasporto o per riporlo dopo l'uso. Un secondo treppiede funge da supporto per bracciolo ottenuto da materiali di recupero. Il sistema è completato da un telo scuro appoggiato a una struttura in alluminio per evitare che la luce laser colpisca eventuali altre persone in laboratorio. Si tratta di una misura precauzionale aggiuntiva ma non indispensabile, data la ridotta intensità della luce laser.
Un'alternativa e più solida base rotante, realizzata da Giulio Marin[21], offre un appoggio in plexiglass per la stabilizzazione della mano, ma che al contempo appiattisce il palmo a causa del peso della mano stessa. Se la piattezza è accettabile e il soggetto in esame non è in grado di mantenere la posa, conviene usare il nuovo supporto.

3.1.2 Preparazione all'acquisizione

Prima dell'acquisizione, sulla mano del volontario vengono disegnati con un pennarello diversi *marker*; in alcuni casi, delle croci posizionate opportunamente evitano possibili ambiguità nel riconoscimento a occhio nudo dei marker. Non è stato definito un protocollo di posizionamento di questi ultimi in quanto non necessario, ma l'euristica adottata è quella di posizionare un marker in corrispondenza di ogni giunto delle dita, uno al centro del palmo, uno al centro del dorso, e gli altri concentrati in posizioni "strategiche" per il corretto allineamento, come la punta delle dita e il profilo superiore del pollice. I marker sono indispensabili per impostare manualmente alcune corrispondenze tra due viste da allineare, come richiesto dall'algoritmo ICP[6] nella sua concezione iniziale. In Figura 3.4 è riportato un tipico posizionamento dei marker; notare le croci in corrispondenza delle unghie.

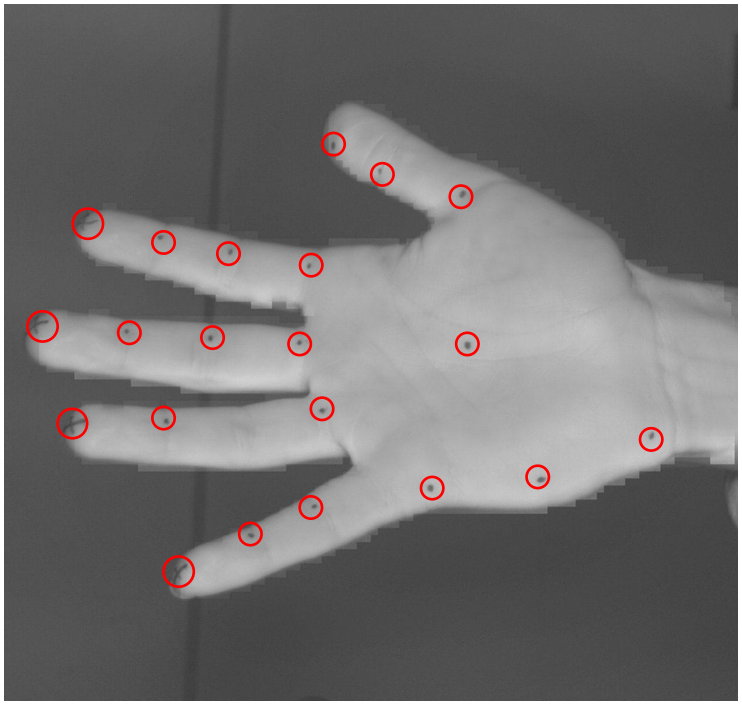


Figura 3.4: Esempio di posizionamento dei marker (lato palmo) con croci e marker evidenziati.

Dopo aver collegato lo scanner al PC e configurato il software d'acquisizione ScanStudioHD Pro, l'operatore regola l'altezza del bracciolo e la posizione dell'asta del treppiede dello scanner in modo che la mano del volontario rientri completamente nel volume visuale del dispositivo per ogni possibile rotazione della testa orbitale. Indicativamente, il baricentro della mano deve trovarsi in corrispondenza dl centro di rotazione della testa orbitale, e per evitare interferenze del bracciolo durante la scansione è necessario che il polso non sia appoggiato a quest'ultimo. La posizione lungo il braccio in alluminio è stata calibrata in fase di assemblaggio del sistema in modo che lo scanner si trovi a circa 45cm dal centro di rotazione, compatibilmente alla distanza ottimale a cui deve essere posizionato l'oggetto da acquisire in modalità *wide*. La rotazione avviene agendo su un'apposita manovella di cui è dotata la testa orbitale.
Durante la regolazione la mano può essere rilassata, ma né il braccio né il bracciolo possono essere spostati. Questo vale anche per i passi successivi.

3.1.3 Acquisizione

L'operatore esegue una sequenza di scansioni della mano ad angoli diversi, orientando manualmente il braccio prima di ogni scansione e assicurandosi che le vibrazioni dello scanner siano cessate. Si ribadisce il fatto che, durante il processo, le posizioni dei due treppiedi non vanno modificate, e il soggetto deve mantenere la mano **assolutamente** immobile. In Figura 3.5 è riportato un esempio di foto B/W scattata dallo scanner prima della scansione, e serve da *texture* da applicare alla futura mesh parziale ricostruita dal software tramite *triangolazione*. Come si nota dall'immagine, il volontario è munito di occhiali protettivi specifici per la lunghezza d'onda del laser, benché la sua intensità non sia tale da arrecare danni alla retina.
Il difetto principale di questo sistema è l'immobilizzazione della mano: l'algoritmo di fusione delle varie viste implementato nel software è una variante di ICP che, nella sua accezione originale, assume l'ipotesi di corpo rigido. Affinché l'ipotesi sia ancora valida nel caso della mano, è indispensabile che essa sia perfettamente immobile, altrimenti le viste si riferiscono a un diverso oggetto e la minimizzazione dell'errore di allineamento diventa impossibile. Benché il volontario possa trovare una posizione comoda su una sedia che non affatichi il braccio, è inevitabile che la mano dopo poco tempo compia micro movimenti involontari dovuti all'accumulo di tensione nel mantenimento della posa. Di per sé lo scostamento della mano di pochi millimetri non causa seri problemi, a patto che le dita rimangano immobili, ma questa situazione si verifica di rado in un contesto reale. Un

lieve scostamento di un dito da una vista all'altra può causare gravi errori nella fase di fusione delle varie viste.

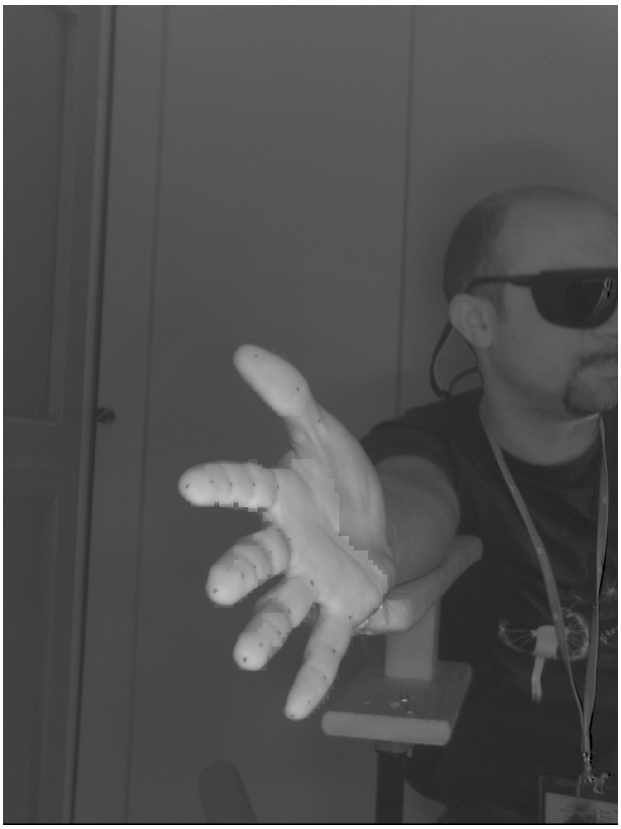


Figura 3.5: Esempio di foto B/W scattata da NextEngine™prima di una scansione.

Per ridurre l'affaticamento della mano, si è cercato di minimizzare il tempo richiesto da un'acquisizione completa agendo sulle impostazioni del software e riducendo il numero di viste, ma al contempo massimizzando il livello di dettaglio. L'ambizione non è quella di ricostruire fedelmente la mano del volontario per realizzare animazioni realistiche, ma ridurre al minimo gli errori nella successiva determinazione dei parametri del modello. Del resto, è sempre possibile ridurre il livello di dettaglio di una mesh successivamente tramite alcuni algoritmi di decimazione, mentre non è possibile aumentarlo tramite algoritmi di interpolazione senza introdurre errori.
In seguito a svariati test di posizionamento della mano mirati alla "copertura" della stessa con il minor numero di viste, è emerso che cinque o al massimo sei viste con il polso ruotato di circa 45°e con le dita aperte sono sufficienti a ricostruire una mesh di discreta qualità. Entrando nel dettaglio, due viste coprono il palmo e il dorso della mano, una copre la parte superiore del pollice e parte della pelle delle dita non coperta dalle due viste precedenti, mentre le ultime due sono le più

delicate in quanto coprono la punta delle dita e sono determinanti nel raccordo della vista del palmo con quella del dorso. L'angolo delle viste "di raccordo" non è fissato, ma è compreso tra i 60° e i 90° per la prima e tra i 90° e i 120° per la seconda. Eventualmente, un'ulteriore sesta vista, può servire a coprire la parte inferiore del mignolo e del palmo, ma in genere essa non è necessaria. La Figura 3.6 riassume graficamente quanto appena detto.

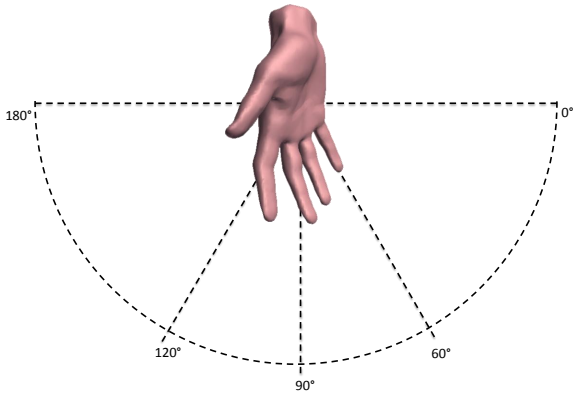


Figura 3.6: Posizioni del laser scanner scelte per l'acquisizione delle viste parziali.

Una singola scansione di una vista parziale a livelli di grigio in modalità *wide* con la minima risoluzione richiede meno di un minuto per essere portata a termine, mentre l'acquisizione completa richiede meno di dieci minuti. Una parte considerevole del tempo è quella richiesta dall'operatore per impostare manualmente l'angolo della vista agendo sulla testa rotante. Il tempo può essere ulteriormente ridotto operando in coppia: un operatore si occupa delle acquisizioni, mentre un altro riposiziona prontamente lo scanner al termine di ognuna di esse. Non si esclude che, in futuro, la testa venga motorizzata per accelerare il posizionamento.

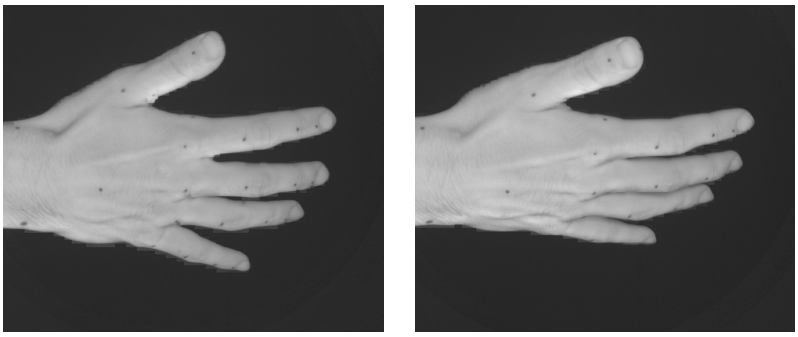
A termine della fase di acquisizione, i dati a disposizione nel software sono una sequenza di mesh incomplete con texture a livelli di grigio, ognuna riferita a una vista parziale della mano. Affinché i marker siano visibili, è necessario che il loro colore sia scuro e possibilmente non rosso. Il blu è adeguato allo scopo. Prima di passare alla fase successiva, vale a dire il pre-allineamento delle varie viste, l'operatore deve esaminarle assicurandosi che, almeno visibilmente, la mano non si sia mossa sensibilmente durante l'acquisizione. Un segno di eccessivo movimento è l'eventuale alone che si nota attorno alle dita.

In Figura 3.7 è mostrato uno screenshot di parte dell'interfaccia di ScanStudio HD al termine della raccolta di tutte le viste necessarie.

Figura 3.7: Sequenza di viste parziali raccolte in Scan Studio.

Per completezza, si conclude la sezione corrente riportando in Figura 3.8 un esempio di foto scattate dallo scanner durante l'acquisizione della mano di un volontario. Da esse saranno estratte le texture applicate automaticamente a ogni vista parziale, come accennato in precedenza. La vista opzionale in questo esempio non si è resa necessaria.

a) vista a 0° b) vista a circa 60° c) vista a circa 120°



d) vista a 180° e) vista a circa 30° di elevazione.

Figura 3.8: Esempio di foto scattate da NextEngine™prima dell'acquisizione di ogni vista parziale.

3.1.4 Allineamento

Le viste ottenute al passo precedente non sono ancora pronte a essere fuse in un'unica mesh per due motivi:

1. le viste parziali contengono con alta probabilità anche porzioni della superficie del bracciolo o di altri oggetti della scena
2. le informazioni sugli angoli delle viste sono assenti

Il primo problema è facilmente risolvibile grazie a un semplice tool per il *crop* offerto da ScanStudio, mentre il secondo problema deriva dal fatto di aver ruotato manualmente lo scanner attorno alla mano, al posto di averla fatta ruotare dalla base rotante: dal punto di vista del software d'acquisizione, infatti, né lo scanner né l'oggetto in esame si sono mossi. Se, idealmente si posizionasse ogni mesh parziale nello stesso world space, le viste parziali sarebbero sovrapposte e non ricostruirebbero, così, il profilo della mano voluto. Tale affermazione è verificabile importando le viste parziali non allineate in MeshLab, come mostrato in Figura 3.9.

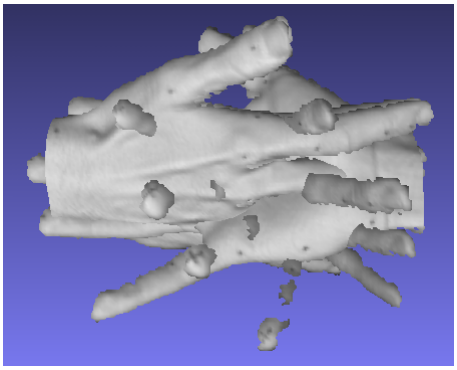
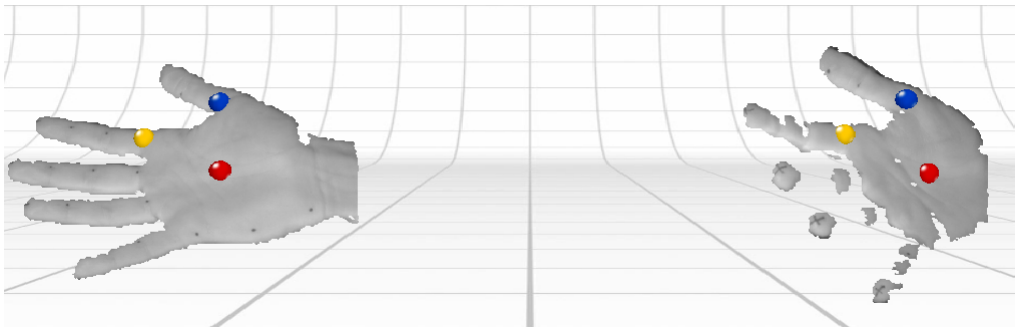


Figura 3.9: Sequenza di viste parziali raccolte in ScanStudio HD.

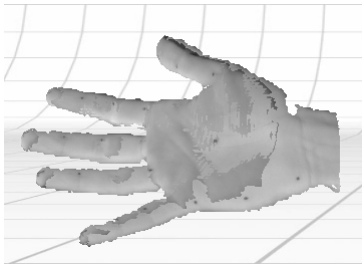
Per ovviare al secondo problema, è necessario reintrodurre le informazioni sugli angoli, e a dire il vero anche sulle posizioni[3], tramite una procedura di *allineamento* da effettuare tramite un tool interno a ScanStudio, benché non dia buoni risultati. Il motivo è che il tool analogo offerto da MeshLab, *align*, pur restituendo risultati migliori, non conserva le texture; di conseguenza, i marker non sono visibili nel tool di MeshLab in quanto appartenenti a texture, e non possono essere quindi usati come supporto all'allineamento. L'allineamento senza

[3]Difficilmente, infatti, la mano è posizionata in corrispondenza dell'esatto centro di rotazione della testa fotografica, ergo le viste potrebbero avere range diversi di profondità.

punti di riferimento è un'operazione ardua e soggetta a errori, pertanto è stata evitata.
L'idea adottata è effettuare un pre-allineamento manuale tra coppie di viste in ScanStudio, non necessariamente accurato, seguito da un raffinamento ad alta precisione in Meshlab. Il preallineamento è condotto scegliendo di volta in volta un numero limitato di marker per segnalare alcune corrispondenze, e al termine di questa fase il risultato è la registrazione globale delle varie viste, cioè, per ogni vista l'algoritmo estrae la matrice di rototraslazione che minimizzi l'*errore di allineamento*. Tale errore varia sensibilmente da circa 2.54mm a circa 0.03mm a seconda della qualità dell'acquisizione. La procedura richiede molto tempo ed è inevitabile anche a causa dell'indisponibilità di tool efficaci in grado di registrare globalmente le varie viste, nonostante letteratura esistano alcuni metodi come [11]. In Figura 3.10 è mostrato un esempio di allineamento di due viste parziali in ScanStudio.



a) prima dell'allineamento



b) dopo l'allineamento

Figura 3.10: Esempio di pre-allineamento di due viste parziali in ScanStudio HD.

Importando, ora, le viste pre-allineate in MeshLab per le successive elaborazioni, ciò che appare sullo schermo è una mesh incompleta ma con le fattezze della mano reale. A dire il vero, se la mano si è mossa eccessivamente in fase d'acquisizione, è probabile che siano presenti in alcune viste diversi artefatti, per esempio un pezzo di dito piegato che fuoriesce dal profilo corretto del dito o un

dito non allineato. In Figura 3.11 è mostrato un tipico artefatto reso evidente al termine del pre-allineamento.

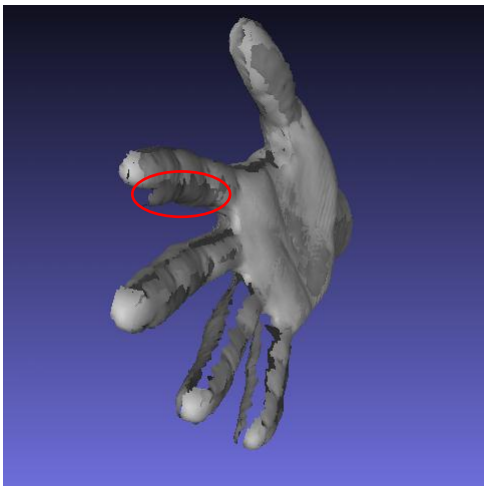


Figura 3.11: Viste pre-allineate e importate in Meshlab. L'artefatto è evidenziato in rosso.

In questi casi, spesso è sufficiente individuare la vista che contiene l'artefatto e, se possibile (in genere quando più viste coprono la stessa area), rimuoverlo tramite i tool di MeshLab. Se il numero di artefatti è eccessivo e non è possibile rimuoverli senza corrompere eccessivamente il profilo della mano, è necessario ripetere l'acquisizione.
Dopo questa prima "ripulitura", il pre-allineamento eseguito in ScanStudio viene rifinito tramite un tool di allineamento automatico di Meshlab. La registrazione globale stavolta è possibile in quanto il tool non richiede la specifica manuale di corrispondenze, a patto che le varie viste siano quasi del tutto allineate. Il risultato dell'allineamento è riportato in Figura 3.12.

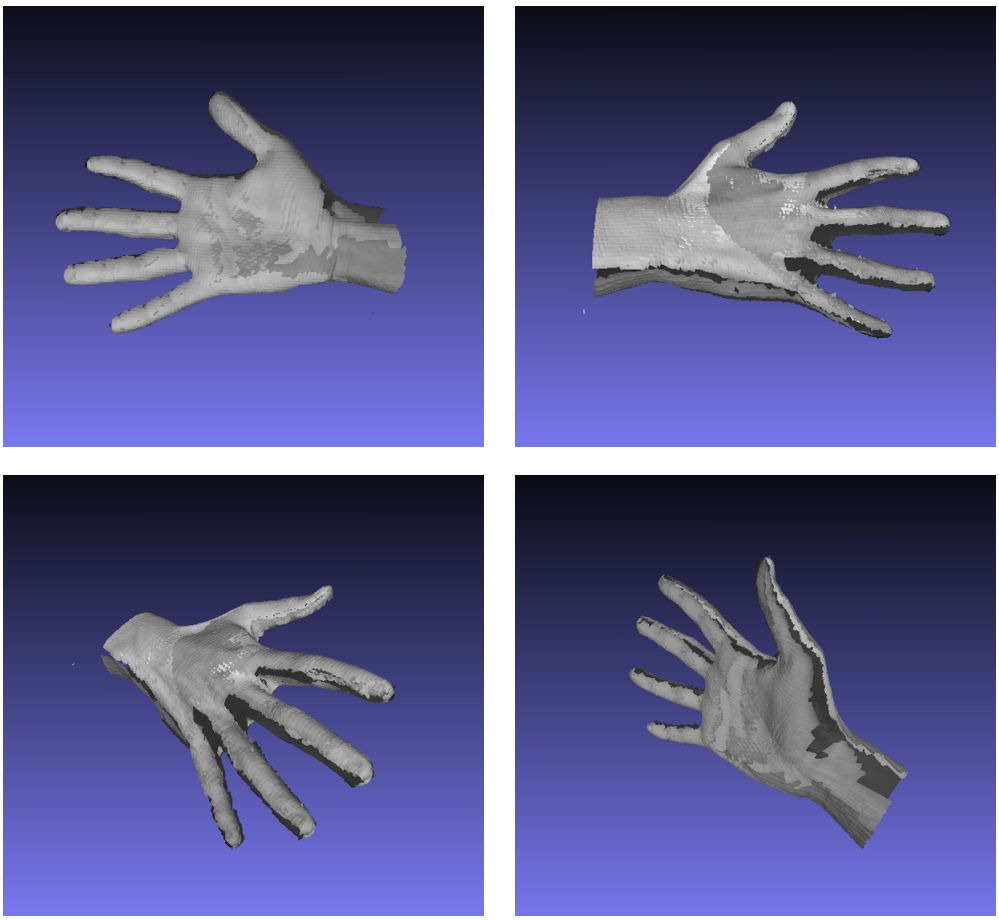


Figura 3.12: Esempio di viste parziali registrate globalmente tramite Meshlab.

3.1.5 Fusione e remeshing

La mesh risultante nelle fasi precedenti e illustrata in Figura 3.12 è ancora composta di varie viste, benché allineate. Il passo preliminare della fase corrente consiste nel far *collassare* in Meshlab tutte le viste, ognuna associata a un livello, in un unico livello. Dopo la fusione, eseguita in meno di un secondo, ha senso parlare di *mesh della mano*, la quale presenta visivamente innumerevoli "buchi" di grande dimensione dovuti principalmente a quattro concause:

1. regioni non colpite dalla luce laser
2. regioni che hanno assorbito completamente la luce laser
3. regioni che hanno rifratto la luce laser
4. errori dell'algoritmo di triangolazione

Esistono in letteratura due categorie di algoritmi di *hole filling*: gli algoritmi della prima categoria affrontano il problema ricercando una *patch* con curvatura compatibile a quella della superficie che circonda il buco da colmare; si cita il metodo di Liepa[18]. Gli algoritmi della seconda categoria, invece, adottano un approccio volumetrico campionando il volume in cui è immersa la mesh in *voxel* e determinando quali campioni appartengano alla superficie e quali allo spazio vuoto. Il maggiore esponente della seconda categoria è il metodo di [22] che, tra l'altro, impiega l'algoritmo Marching Cubes[19] per estrarre dai voxel la mesh completata. Gli autori dell'approccio hanno fornito un software che lo implementa, ma ai fini del progetto esso è stato abbandonato a causa degli innumerevoli artefatti introdotti nella ricostruzione. Entrambi gli approcci hanno in comune il fatto di poter chiudere solo buchi di dimensioni esigue, mentre non sono in grado di colmare le lacune dovute a serie mancanze di dati, come per la mano in esame. Normalmente, l'unica soluzione in questi casi è introdurre *ridondanza* nei dati aumentando il numero di viste parziali, allungando, però, il tempo di acquisizione. Inoltre, data la complessità della mano, un maggior numero di viste non assicura sempre una maggiore copertura della stessa.
In seguito a svariati tentativi fallimentari di hole filling in Meshlab, il problema è stato risolto brillantemente all'interno dello stesso programma ricostruendo[4] la mesh tramite l'algoritmo di ricostruzione secondo Poisson[15]. Il metodo, però, sembra funzionare correttamente solo in Windows 7 e richiede diversi secondi anche impiegando un computer con prestazioni elevate. In Figura 3.13 è mostrato

[4]Il termine tecnico è *remeshing*.

un esempio di mesh ricostruita a partire dalle viste riportate in Figura 3.12; i valori delle componenti cromatiche scelte per lo shading sono state prelevate da una foto.

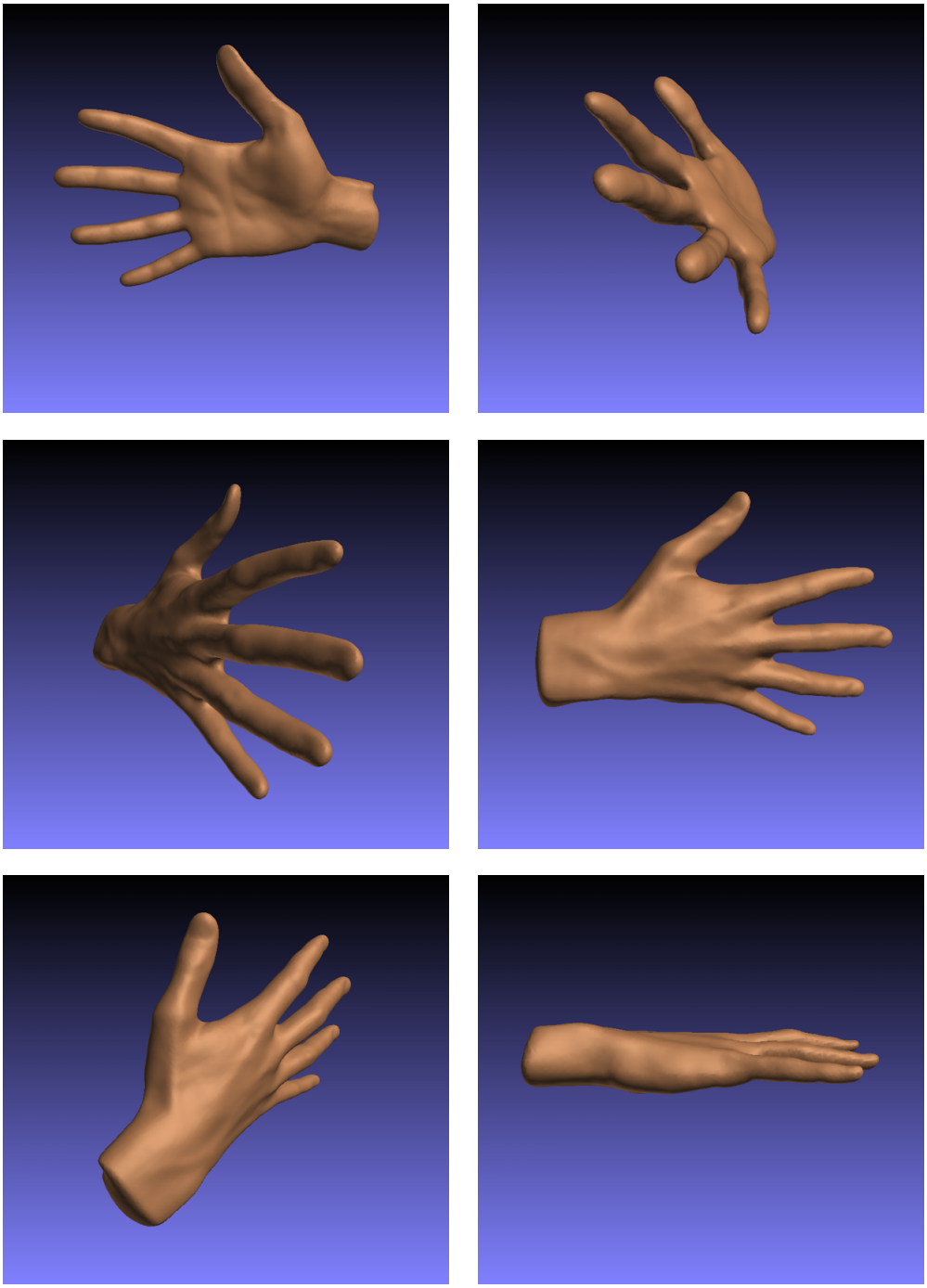


Figura 3.13: Esempio di mano ricostruita tramite l'algoritmo[15].

3.1.6 Semplificazione della mesh

La mesh restituita dal passo precedente è pressoché completa. L'elevata accuratezza di NextEngine[5] comporta un alto livello di dettaglio nella superficie ricostruita, ma al contempo aumenta vertiginosamente la *taglia* della mesh, vale a dire la dimensioni delle strutture dati contenenti le coordinate dei vertici e i loro raggruppamenti in facce che la descrivono. Una taglia elevata peggiora drammaticamente le prestazioni degli algoritmi che elaborano i dati sopracitati ma, a volte, è più che accettabile una riduzione del livello di dettaglio a fronte di un sensibile aumento delle prestazioni. La mano rappresentata in Figura 3.13, per esempio, è descritta da circa 10^5 facce e $5 \cdot 10^4$ vertici per una dimensione su disco poco superiore a 3MB.
La riduzione della dimensionalità, o equivalentemente la *semplificazione* della mesh, è condotta in Meshlab attraverso un filtro *decimatore* che implementa l'algoritmo *Quadric-based Surface Decimation*[13]. La decimazione è *ottimale* secondo una metrica definita dagli inventori del metodo. In Figura 3.14 è riportata la mesh della Figura 3.13 in seguito a una riduzione del 90% sia del numero di facce sia del numero dei vertici, e si nota che la forma della mano e la curvatura della superficie sono state preservate.

La scelta di ridurre la dimensionalità del 90% non è casuale, ma il valore è stato stimato tenendo conto dei valori dei dati di targa del sensore, in particolar modo la lunghezza focale e la risoluzione, e della specifica del modello pinhole in Figura 1.7. Alcuni semplici calcoli restituiscono l'ordine di grandezza del minor numero di poligoni che la mesh del modello dovrebbe avere affinché il livello di dettaglio non sia eccessivo rispetto alla bassa risoluzione del sensore. Secondo i dati di targa, il lato di un pixel del sensore è lungo $40 \cdot 10^{-6} m$[6], e dalle formule $u = d\frac{X}{Z}$ e $v = d\frac{Y}{Z}$ si ricava che un segmento di lunghezza l_{uv} giacente sul piano immagine e proiettato all'indietro in $\mathbb{R}^3$ ha lunghezza pari a $L_{xyz} = Z\frac{l_{uv}}{d}$. Dall'ultima formula si evince che la risoluzione diminuisce in maniera proporzionale all'aumentare della distanza dall'obiettivo, o equivalentemente la distanza minima tra due punti affinché questi non siano proiettati sullo stesso pixel aumenta all'aumentare della profondità. Estendendo il concetto all'area di un pixel, l'area minima di una faccia affinché quest'ultima non venga proiettata assieme alle facce adiacenti su uno stesso pixel è proporzionale del quadrato della distanza: $A_{xyz} = \frac{(Z\frac{l_{uv}}{d})^2}{2} = Z^2\frac{l_{uv}^2}{2d^2}$[7]. Di conseguenza, assumendo che la mano da tracciare

[5] Si ricorda essere inferiore agli 0.4mm in modalità *wide*.
[6] Nel caso ideale, infatti, i pixel sono quadrati, cosa non sempre vera nella realtà.
[7] Il 2 al denominatore deriva dalla scindibilità di una faccia quadrata in due facce triangolari.

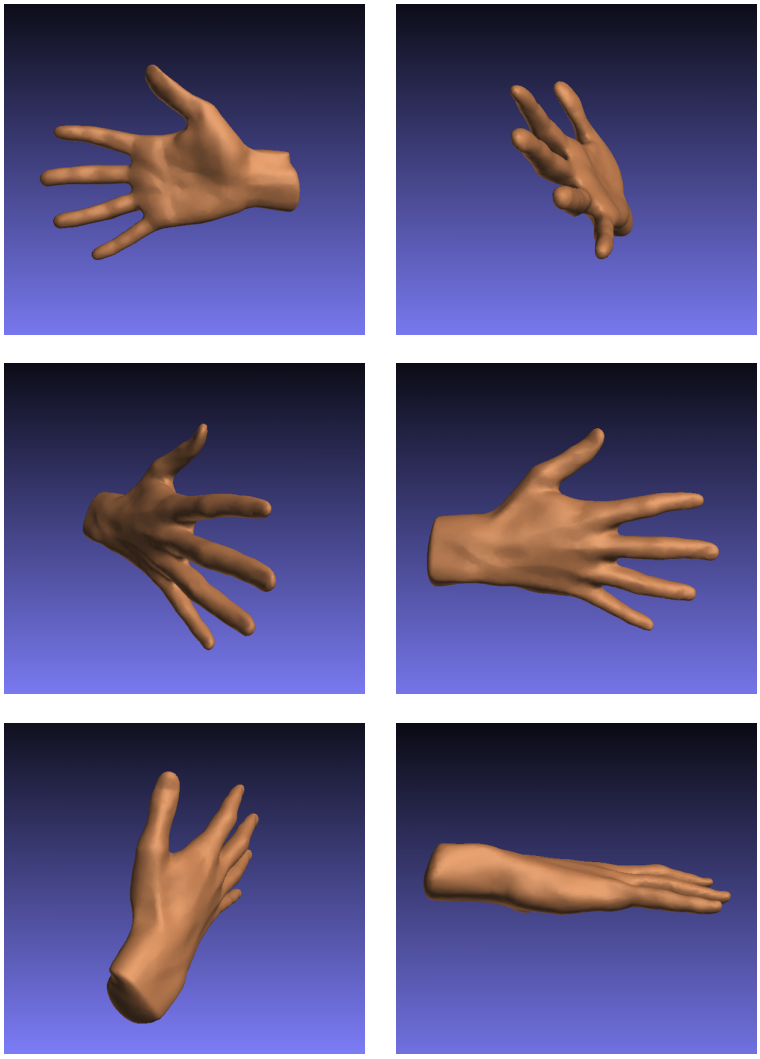
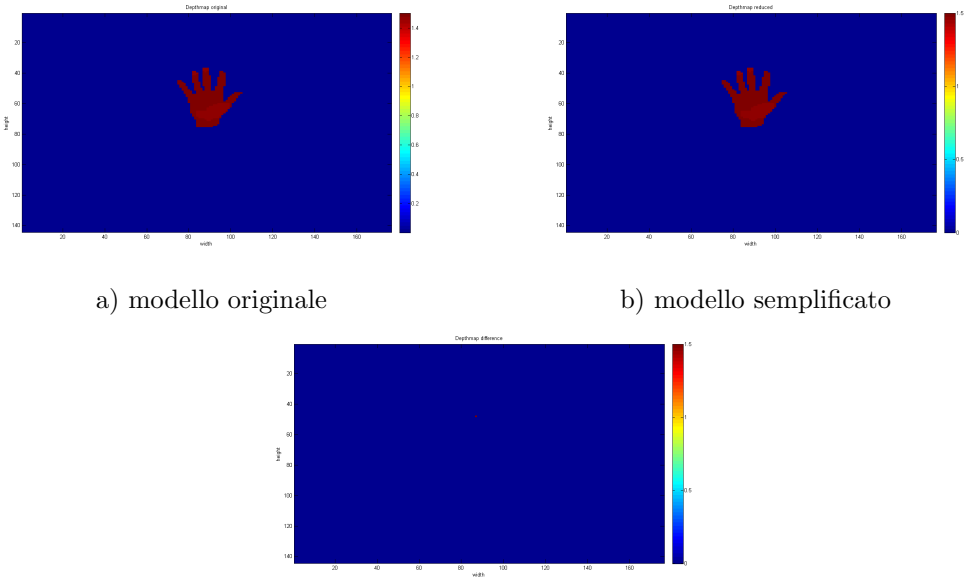


Figura 3.14: Esempio di decimazione della mesh in Figura 3.13 tramite l'algoritmo[13].

non disti meno di 1m dall'obiettivo, come negli esperimenti condotti nel progetto, si ha banalmente che $A_{xyz_{min}} = \frac{l_u v^2}{2d^2} \approx 4.5mm^2$ sostituendo i valori dei dati di targa. In Maya l'area della mesh in Figura 3.13 calcolata ha un valore di circa $49000mm^2$ che, diviso per l'area minima di una faccia stimata, restituisce una stima di poco meno di 11000 facce. Dato che la mesh di partenza ha più di 10^5 facce, è provato che la riduzione può spingersi fino al 90% senza una perdita visibile di dettaglio.
Come ulteriore conferma, la Figura 3.15 contiene un'immagine data dalla differenza della depthmap sintetica generata dal modello con la mesh non semplificata con la depthmap generata dallo stesso modello con mesh semplificata. La scelta della scala di colore e di impostare la profondità dei pixel dello sfondo a 0 al posto di 5m è stata operata solo per evidenziare le profondità dei pixel della mano.



a) modello originale

b) modello semplificato

c) differenza tra i modelli

Figura 3.15: Valutazione della semplificazione

Applicando opportuni filtri in Meshlab è possibile eliminare alcuni eventuali artefatti introdotti in fase di ricostruzione o semplificazione, o che comunque non sono stati rimossi durante le fasi precedenti. Tra i difetti principali si annoverano:

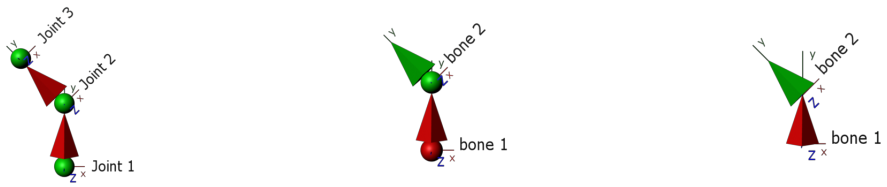
- vertici isolati dalla superficie (non manifold)
- vertici *vicini*[8]
- vertici ripetuti
- facce orientate in direzione opposta rispetto a quella delle facce adiacenti
- facce ripetute
- facce ad area nulla

Infine, è possibile orientare e posizionare la mesh nella maniera che si ritiene opportuna. In questo progetto, si è scelto roto-traslare la mesh affinché il baricentro (calcolato automaticamente dal programma) coincida con il sistema di riferimento globale e il palmo sia orientato in direzione del semiasse Z positivo. L'ultimo passo consiste nell'esportazione della mesh completa e orientata in un file per le successive elaborazioni.

[8]Secondo la norma euclidea. Il filtro fonde i vertici in esame in un unico vertice.

3.2 Definizione di uno scheletro

Dopo la ricostruzione della mesh, descritta nella Sezione 3.1, è stata affrontata la specifica di uno scheletro che conferisca l'articolarità al modello e una procedura di *inserimento* che permetta di calibrare la posizione dei giunti o, equivalentemente, la posizione e la lunghezza delle "ossa". A ogni giunto è associato un sistema di assi coordinati *locale* che definisce la posizione e l'orientamento relativi del giunto rispetto al giunto *padre*. La distanza euclidea relativa tra i due giunti definisce la *lunghezza dell'osso* e la rotazione relativa il suo orientamento. La rotazione di un giunto equivale alla rotazione dell'osso (o delle ossa) che originano da esso. In Figura 3.16 è esemplificato il legame tra giunti e ossa appena illustrato.



a) giunti evidenziati b) ossa evidenziate c) solo ossa

Figura 3.16: Possibili rappresentazioni di uno scheletro.

Secondo questa modellazione, usata anche in altri ambiti come la robotica, per definire la posizione e l'orientamento *globali* di ogni osso è sufficiente conoscere quelle del giunto padre, la distanza e l'orientamento locali. Assumendo plausibilmente l'invariabilità nel tempo della distanza relativa tra i due giunti, o equivalentemente l'invariabilità della lunghezza delle ossa, ciò che cambia sono solo le rotazioni locali. Applicando le note trasformazioni tra sistemi di riferimento si ha, secondo la definizione di *catena cinematica* riportata nell'Equazione 3.1

$$T_i = \begin{cases} T_{p(i)\to g} T_{i\to p(i)} & i \neq root \\ T_{i\to g} & i = root \end{cases}$$

con $T_{p(i)\to g}$ matrice di trasformazione dal sistema di riferimento *locale* del *predecessore* del giunto i-esimo al sistema di riferimento *globale* e $T_{i\to p(i)}$ matrice di trasformazione *locale* del giunto i-esimo rispetto al sistema di riferimento del suo *predecessore*

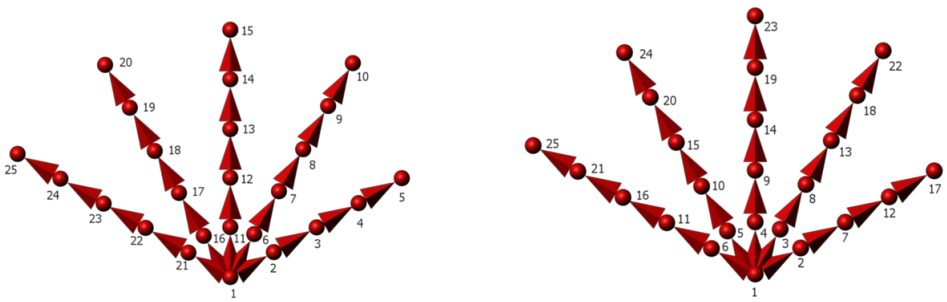
(3.1)

Nel progetto vengono impiegate le coordinate omogenee al posto delle coordinate in $\mathbb{R}^3$ sia per il dovuto passaggio agli spazi proiettivi, motivato nel Capitolo 4, sia per la maggiore compattezza notazionale e nel codice che le relative trasformazioni comportano. Secondo quanto appena detto, è possibile riscrivere l'Equazione 3.1 come:

$$T_i = \begin{cases} T_{p(i)\to g} \begin{bmatrix} R(\theta_i, \phi_i, \psi_i) & D_i \\ 0 & 1 \end{bmatrix} & i \neq root \\ \begin{bmatrix} R(\theta_1, \phi_1, \psi_1) & D_1 \\ 0 & 1 \end{bmatrix} & i = root \end{cases}$$

con $D_i = (d_{x_i}, d_{y_i}, d_{z_i})$ posizione relativa e $R(\theta_i, \phi_i, \psi_i)$ matrice di rotazione del giunto i-esimo rispetto al sistema di riferimento *locale* del giunto *predecessore*. Nel caso di root, $R(\theta_i, \phi_i, \psi_i)$ coincide con la matrice di rotazione del giunto i-esimo rispetto al sistema di riferimento *globale*. (3.2)

Dall'Equazione 3.2 si evince che, per definire la posa dello scheletro, è sufficiente fornire la posizione del giunto *radice* nello spazio tridimensionale e le *rotazioni relative* di ogni giunto rispetto al giunto padre o, secondo un'altra ottica, le rotazioni relative delle ossa imperniate nei giunti. In particolare, i giunti aventi come padre la *radice* conferiscono la mobilità al polso.
Per la mano è stato progettato uno scheletro costituito da 25 giunti e sono stati definiti due possibili ordinamenti topologici degli stessi, riportati in Figura 3.17.



a) b)

Figura 3.17: Possibili ordinamenti topologici valutati.

Le Tabelle 3.1 e 3.2 riportano, in ordine di indice crescente, le etichette associate per chiarezza ai singoli giunti. I nomi rimandano volutamente ai loro corrispettivi anatomici.

Joint index	Joint name	Joint index	Joint name
1	root	14	mean-phalanxs 3
2	thumb-wrist	15	mean-tip
3	thumb-phalanxs 1	16	ring-wrist
4	thumb-phalanxs 2	17	ring-phalanxs 1
5	thumb-tip	18	ring-phalanxs 2
6	index-wrist	19	ring-phalanxs 3
7	index-phalanxs 1	20	ring-tip
8	index-phalanxs 2	21	pinky-wrist
9	index-phalanxs 3	22	pinky-phalanxs 1
10	index-tip	23	pinky-phalanxs 2
11	mean-wrist	24	pinky-phalanxs 3
12	mean-phalanxs 1	25	pinky-tip
13	mean-phalanxs 2		

Tabella 3.1: Etichette per l'ordinamento a) illustrato in Figura 3.16.

Joint index	Joint name	Joint index	Joint name
1	root	14	mean-phalanxs 2
2	thumb-wrist	15	ring-phalanxs 2
3	index-wrist	16	pinky-phalanx 2
4	mean-wrist	17	thumb-tip
5	ring-wrist	18	index-phalanxs 3
6	pinky-wrist	19	mean-phalanxs 3
7	thumb-phalanxs 1	20	ring-phalanxs 3
8	index-phalanxs 1	21	pinky-phalanxs 3
9	mean-phalanxs 1	22	index-tip
10	ring-phalanxs 1	23	mean-tip
11	pinky-phalanxs 1	24	ring-tip
12	thumb-phalanxs 2	25	pinky-tip
13	index-phalanxs 2		

Tabella 3.2: Etichette per l'ordinamento b) illustrato in Figura 3.16.

I due ordinamenti hanno in comune il fatto di avere l'indice di un giunto sempre superiore a quello dei giunti che lo precedono nella catena cinematica. Tale regola è formalizzata nell'Equazione 3.3

$$i > j \ \forall \text{ i, j} = 1, 2, ..., \text{n t.c. } j \prec i \quad \text{con } \prec \text{ relazione di precedenza} \tag{3.3}$$

Il motivo di tale scelta è dovuto a una questione d'efficienza: il risultato di un'elaborazione riferita a un giunto può dipendere da risultati di elaborazioni riferite a giunti precedenti nella catena cinematica (nella fattispecie, si ha una catena separata per ogni dito) e mai a giunti che si trovano in posizioni successive. Segue che, eseguendo le elaborazioni secondo uno degli ordinamenti topologici precedentemente descritti, e adottando opportune strutture dati, il risultato di un'elaborazione di un giunto è immediatamente disponibile alle successive elaborazioni dei giunti *successori* nella catena. Durante l'elaborazione di un giunto, quindi, non è necessario riprocessare di volta in volta i giunti che lo precedono, ma solo richiamarne i risultati tramite una *look-up-table*. Tale principio è adottato anche nella definizione di *albero cinematico*: partendo dalla radice, la matrice di trasformazione dal sistema locale al giunto al sistema globale è data dal prodotto della matrice di trasformazione locale per la matrice di trasformazione dal sistema locale al giunto padre al sistema globale, calcolata al passo precedente. Calcolando le matrici di trasformazione sistema locale - sistema globale in ordine topologico, ogni elaborazione consiste solamente nel prodotto di due matrici 4x4. L'orientamento di ogni giunto è esprimibile specificando tre rotazioni elementari attorno agli assi coordinati e una convezione angolare, o da un versore a tre componenti e un angolo che definiscano una rotazione attorno all'asse specificato dal versore. Al momento nel codice le rotazioni sono specificate nel primo modo, perché è più intuitivo e facilita l'impostazione di vincoli nel modello, definiti nella Tabella 3.3

Joint name	Min. x[deg]	Max. x[deg]	Min. y[deg]	Max. y[deg]	Min. z[deg]	Max. z[deg]
root	0	359	0	359	0	359
thumb-wrist	-*	-*	-80	80	-*	-*
index-wrist	-*	-*	-80	80	-*	-*
mean-wrist	-*	-*	-80	80	-*	-*
ring-wrist	-*	-*	-80	80	-*	-*
pinky-wrist	-*	-*	-80	80	-*	-*
thumb-phalanxs 1	-	-	-45	15	0	30
index-phalanxs 1	-	-	-90	15	-15	30
mean-phalanxs 1	-	-	-90	0	-15	30
ring-phalanxs 1	-	-	-90	0	-15	15
pinky-phalanxs 1	-	-	-90	0	-15	30
thumb-phalanxs 2	-	-	-90	0	-	-
index-phalanxs 2	-	-	-90	0	-	-
mean-phalanxs 2	-	-	-90	0	-	-
ring-phalanxs 2	-	-	-90	0	-	-
pinky-phalanx 2	-	-	-90	0	-	-
index-phalanxs3	-	-	-90	0	-	-
mean-phalanxs 3	-	-	-90	0	-	-
ring-phalanxs 3	-	-	-90	0	-	-
pinky-phalanxs 3	-	-	-90	0	-	-
thumb-tip	-	-	-	-	-	-
index-tip	-	-	-	-	-	-
mean-tip	-	-	-	-	-	-
ring-tip	-	-	-	-	-	-
pinky-tip	-	-	-	-	-	-

Tabella 3.3: Vincoli per ogni giunto o osso.

N.B.: Il simbolo "-" indica il divieto alla rotazione del giunto attorno a un dato asse di rotazione, mentre il simbolo "-*" indica l'obbligo per il giunto di mantenere la rotazione impostata nella posa iniziale.

Nel codice sono, tuttavia, implementate anche le rotazioni secondo la formula di Rodrigues, maggiormente usata nel campo della Computer Vision. Per completezza, si riporta la formula nell'Equazione 3.4

$$\mathbf{v_{rot}} = \mathbf{v}\cos(\theta) + (\mathbf{k} \times \mathbf{v})\sin(\theta) + \mathbf{k}(\mathbf{k} \cdot \mathbf{v})(1 - \cos(\theta)) \quad \text{con } \mathbf{k}=[k_1, k_2, k_3]^T \text{ versore di rotazione} \tag{3.4}$$

La stessa formula è esprimibile in maniera più chiara in termini di matrici di rotazione:

$$\mathbf{v_{rot}} = R \quad \text{con } R = I + [k]_{\mathbf{v}}\sin(\theta) + (1 - \cos(\theta))(kk^T - I) \text{ e } [k]_{\mathbf{v}} \triangleq \mathbf{k} \times \mathbf{v} = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \tag{3.5}$$

Allo stesso modo, data una matrice di rotazione, è possibile esprimere quest'ultima secondo Rodrigues attraverso la coppia di formule indicate nell'Equazione 3.6

$$\theta = \arccos(\tfrac{tr(R)-1}{2}) \quad \text{con } tr(R) \triangleq diag(RR^T) \text{ traccia di R e } k = \tfrac{1}{2\sin(\theta)} \begin{bmatrix} R(3,2) - R(2,3) \\ R(1,3) - R(3,1) \\ R(2,1) - R(1,2) \end{bmatrix} \tag{3.6}$$

Apparentemente, essendoci 25 giunti, in tutto i gradi di libertà del modello sono 78: 75 per le rotazioni e 3 per la posizione del giunto radice (le altre posizioni dipendono solo dalle rotazioni). Grazie ai vincoli impostati, tuttavia, molte rotazioni attorno agli assi sono precluse alla maggior parte dei giunti. Per esempio, come indicato nella tabella 3.3, solo il giunto radice può ruotare di qualsiasi angolo, mentre quasi tutte le dita possono ruotare solo attorno all'asse Y locale. Le punte delle dita, inoltre, non possono ruotare. Si evince che i gradi di libertà effettivi sono inferiori a un terzo di quelli teorici. Il loro numero, benché ridotto rende ancora elevata la complessità del modello.
Una volta definito lo scheletro, la fase successiva consiste nel suo inserimento[9] nella mesh da deformare che, come già detto, ha lo scopo di regolare le posizioni e le distanze relative tra i giunti. Una procedura per l'embedding *automatizzato*

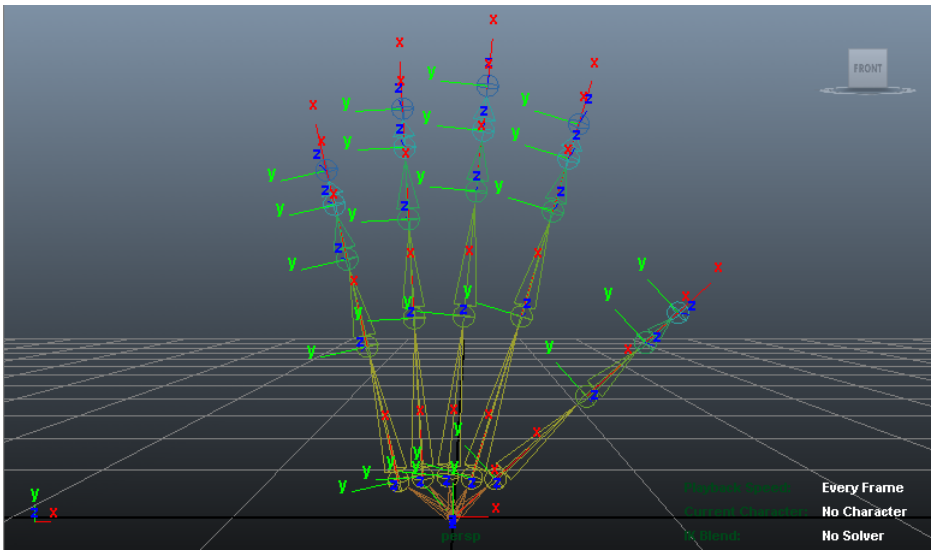
[9]Il termine tecnico è *embedding*.

di uno scheletro predefinito in una mesh è descritta in [5], ma si è optato per l'embedding *manuale* in quanto offre un maggiore controllo nella calibrazione. Il software di modellazione e animazione utilizzato allo scopo è Maya 2012, ma al suo posto è possibile utilizzare il più complicato e gratuito Blender3D. La mesh della mano ricostruita viene esportata in formato .OBJ da Meshlab per essere importata in Maya; per evitare incompatibilità tra i due programmi, vanno esportate solamente la lista dei vertici e la lista delle facce, gli unici dati di fatto richiesti per le successive elaborazioni. Prima di costruire lo scheletro, la mesh importata in Maya va orientata, se non lo è già stata in Meshlab, con il palmo diretto verso la direzione dell'asse Z del sistema globale e pressoché parallelo al piano X-Y.

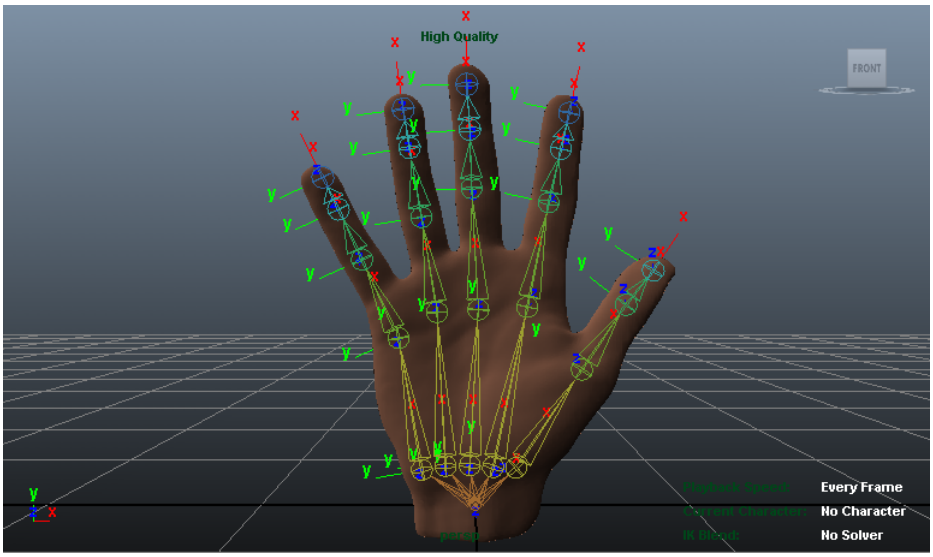
La costruzione dello scheletro è lunga e laboriosa: vengono costruite cinque catene cinematiche, una per dito, raccordate nel giunto radice da posizionare indicativamente nel polso. I giunti vanno etichettati in maniera da poter essere rintracciati rapidamente e, per esportare successivamente la posa senza errori, vanno azzerati tutti i loro angoli di rotazione locale e di orientamento. Dopo aver abbozzato lo scheletro, che al momento avrà la forma di un ventaglio, come in Figura 3.17, il passo successivo è il suo posizionamento all'interno della mesh in luogo del "vero" scheletro della mano. Si comincia posizionando il giunto radice come indicato e, in maniera ordinata, per ogni dito si posizionano i suoi giunti seguendo l'ordinamento sopracitato. Per evitare errori, conviene impostare la trasparenza della mesh e spostare i giunti verticalmente solo traslandoli lungo l'asse verticale del sistema locale (quello che attraversa in lunghezza l'osso), e spostarli orizzontalmente ruotandoli attorno agli assi del sistema locale al giunto padre. Se l'operazione è stata compiuta correttamente, il risultato è del tipo mostrato in Figura 3.18. Da notare il fatto che, in ogni dito, gli assi Z dei giunti sono pressoché allineati verticalmente e gli assi Y coincidono con gli assi attorno ai quali avvengono normalmente le rotazioni delle dita.

La sezione corrente termina con una considerazione: è stato deciso che il sistema di riferimento globale in Maya coincida con il sistema di riferimento locale al giunto radice, in modo da semplificare i calcoli. La mesh e lo scheletro vanno spostati solidalmente e ruotati in modo che il giunto root sia centrato nell'origine e il suo sistema di riferimento allineato con quello globale. Da ora in poi, avendo fatto coincidere il sistema locale del giunto radice con quello globale, per *posizione della mano nello spazio* si intenderà implicitamente la posizione del giunto radice nello spazio tridimensionale o scena in cui la mesh è immersa. In altre parole, il sistema di riferimento globale in Maya diverrà a sua volta un sistema locale

all'oggetto nel mondo virtuale in cui si sposta e deforma il modello.



a) solo scheletro



b) scheletro e mesh

Figura 3.18: Embedding dello scheletro in Autodesk Maya 2012.

3.3 Algoritmo di skinning

Dopo la costruzione e l'embedding dello scheletro nella mesh, il modello è completato dalla definizione di un legame tra le due componenti; la deformazione della mesh dipenderà dalla posa dello scheletro e sarà modulata dai valori di alcuni *pesi* secondo il particolare algoritmo di skinning scelto. Tra i vari algoritmi descritti in letteratura[14], si è scelto di implementare il Linear Blend Skinning[10], un algoritmo datato ma ancora usato spesso in animazione per la sua semplicità ed efficienza. Le sue caratteristiche lo rendono adatto a essere implementato direttamente anche nelle moderne GPU sollevando, così, la CPU, oltre dall'onere del rendering, dai calcoli per le animazioni.
LBS è descritto dalla seguente equazione:

$$v_k^t = LBS_k(M^0, S^0, S^t, v_k^0) = \sum_{i=1}^{b} \omega_{i,k} T_i^t (T_i^0)^{-1} v_k^0$$
con v_k^t k-esimo vertice della mesh deformata all'istante t, $v_k^0 \in M^0$
k-esimo vertice della mesh nella "posa base" (non deformata) con $S^0, S^t \in S$
pose appartenenti allo spazio delle pose assumibili dallo scheletro e T_i^t
definita nell'equazione 3.1. (3.7)

Sviluppando ulteriormente la formula, nell'Equazione 3.8 si evidenzia la dipendenza della deformazione dagli orientamenti relativi dei giunti e dalla posizione del solo giunto radice.

$$v_k^t = \begin{cases} \sum_{i=1}^{b} \omega_{i,k} T_{p(i)\to g} \begin{bmatrix} R(\theta_i, \phi_i, \psi_i) & D_i \\ 0 & 1 \end{bmatrix} (T_i^0)^{-1} v_k^0 & \text{per } i \neq root \\ \sum_{i=1}^{b} \omega_{i,k} \begin{bmatrix} R(\theta_1, \phi_1, \psi_1) & D_1 \\ 0 & 1 \end{bmatrix} & \text{per } i = root \end{cases} \tag{3.8}$$

Nonostante la posa dello scheletro caratterizzi la deformazione della mesh, i pesi $\omega_{i,k}$ presenti nelle Equazioni 3.7 e 3.8 sono la componente più *delicata* del modello in quanto, modulandoli opportunamente, è possibile ottenere deformazioni più o meno realistiche a parità di algoritmo di skinning. I valori dei pesi, infatti, quantificano l'influenza che ha un singolo osso nei confronti di un dato vertice della mesh, e devono rispettare la *condizione di normalizzazione* indicata nella Formula 3.9.

[10] Noto anche sotto il nome di "*skeleton subspace deformation*".

$$\sum_{i=1}^{b} \omega_{i,k} = 1 \ \forall k = 1, 2, ..., n \tag{3.9}$$

La procedura di assegnazione manuale è lunga e ardua, anche per gli esperti di animazione, per cui spesso i pesi vengono generati automaticamente secondo vari approcci, di cui i principali sono:

a) dato un vertice, assegnare una maggiore influenza alle ossa la cui distanza euclidea (nella fattispecie quella di un punto da una retta) dal vertice è minore, secondo l'Equazione 3.10[4]. Da notare che i pesi nell'equazione vanno *normalizzati* per rispettare il vincolo imposto nell'Equazione 3.9.

$$\omega_{i,k} = e^{-\frac{d(i,k)^2}{2\sigma^2}}$$
dove $d(i,k) \in \mathbb{R}$ è la distanza (norma euclidea) del k-esimo vertice dall'osso di indice i e $\sigma \in \mathbb{R}$ è proporzionale alla lunghezza dell'osso. (3.10)

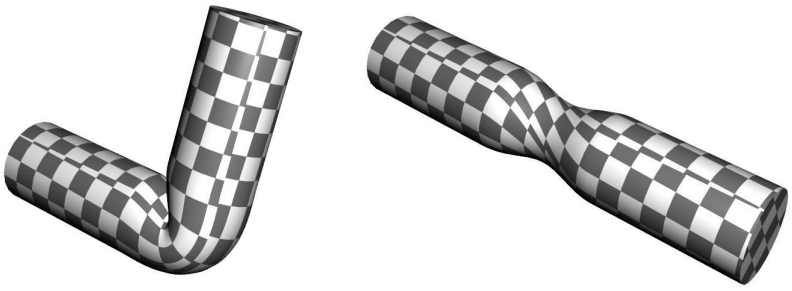
b) Usare un modello di diffusione del calore[5]

Il primo approccio è semplice ma, nel caso della mano, fortemente soggetto al rischio di assegnare un'elevata influenza a un osso di un dito differente da quello a cui appartiene il vertice. Il secondo approccio, invece, è molto più complesso, ma risolve il problema appena citato.
Normalmente, in seguito a una prima distribuzione automatica dei pesi, gli esperti in animazione passano a una lunga fase di raffinamento manuale per correggere, se possibile, gli artefatti generati da un'assegnazione errata. Nel progetto il calcolo dei pesi è affidato per semplicità a un tool di Maya. Eventualmente, in futuro il calcolo sarà implementato direttamente nel codice.

Linear Blend Skinning è noto per l'introduzione di due tipi di artefatti i cui effetti deleteri sono mitigabili da una sapiente correzione dei pesi nella zona compromessa:

- collasso del volume in corrispondenza dei giunti;
- effetto "candy-wrapping".

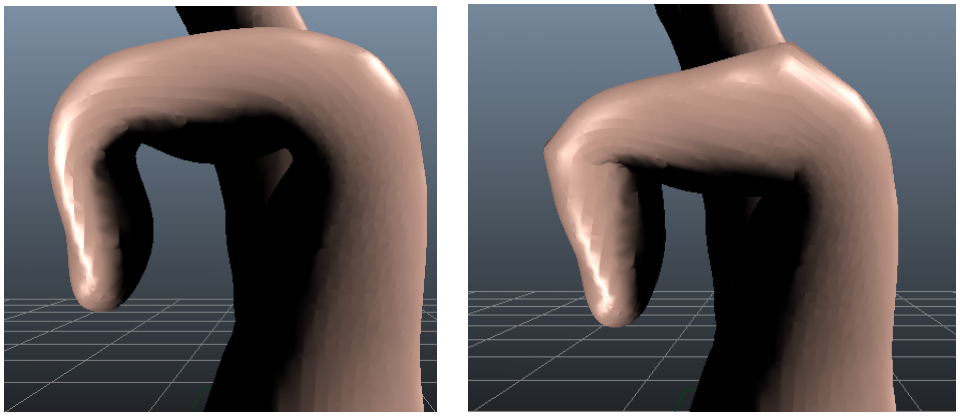
Il primo difetto si verifica quando la rotazione del giunto è eccessiva, mentre il secondo si verifica in seguito a una torsione. In Figura 3.19 è riportato un esempio per entrambi i difetti.



a) collasso del volume b) effetto candy-wrapping

Figura 3.19: Esempio di artefatti introdotti da LBS

Un altro controllo operabile sulla distribuzione dei valori dei pesi e che si fonda sul vincolo 3.9 è l'impostazione del massimo numero di giunti da cui può essere influenzata la deformazione di un vertice. Un numero ridotto porta ad avere deformazioni *rigide* e marcate e annulla molti pesi, dato il decremento di un peso deve essere bilanciato dall'incremento di un altro peso, mentre un numero elevato porta ad avere deformazioni *smussate* ma poco realistiche. Nella Figura 3.20 sono confrontati gli effetti di due diversi valori di influenza.



a) n = 5 b) n = 2

Figura 3.20: Esempio di deformazione di un dito a seconda del numero di giunti di influenza.

Per la mano, è stato deciso di impostare a 2 il numero di giunti che possono influenzare un vertice, per evitare un eccessivo "smussamento" delle nocche come esemplificato in Figura 3.20. Al momento è impostata la stessa influenza anche per le ossa del palmo, benché tale limite generi diversi artefatti dovuti alla mancanza della modellazione dell'elasticità della pelle da parte di LBS. Un leggero movimento di un osso nel palmo provoca, infatti, un allungamento spropositato dei lati delle facce ai bordi della ridotta zona di influenza dell'osso sui vertici

adiacenti. L'effetto, evidenziato in Figura 3.21, è la "lacerazione" della pelle del palmo.

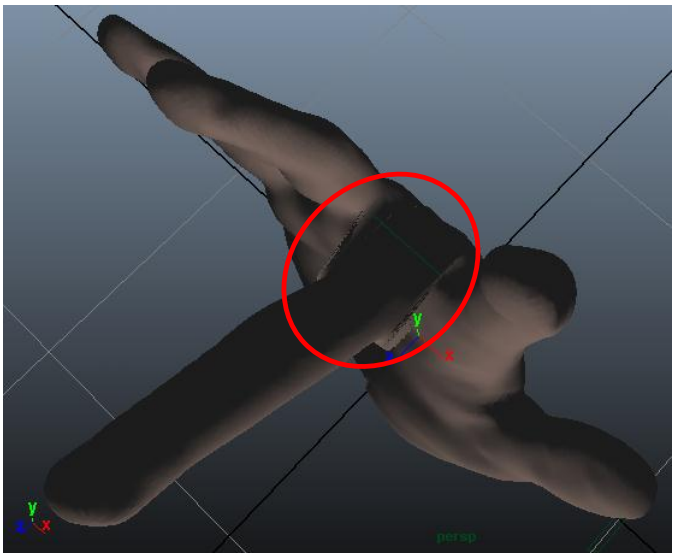


Figura 3.21: Esempio di mancata modellazione dell'elasticità della pelle.

Al termine del posizionamento e del calcolo dei pesi, il legame tra mesh e scheletro è stabilito. In Maya, la rotazione di un singolo osso deforma immediatamente la mesh, ed è opportuno sfruttare questa caratteristica per verificare che le rotazioni delle dita avvengano solo attorno all'asse di rotazione secondo cui avvengono nella realtà. In caso contrario, il legame va interrotto, l'orientamento del sistema locale affetto da errore va corretto e va creato un nuovo legame secondo la procedura descritta in precedenza.
I vertici, le facce, le normali ai vertici, lo scheletro e i pesi sono esportati da Maya in file separati tramite alcuni *script* specifici sviluppati nell'ambito del progetto. I file appena generati sono importabili nel software di tracking in via di realizzazione per le successive elaborazioni e la visualizzazione dei risultati. Le normali sono esportate solo per completezza: la loro validità è limitata, giacché andranno ricalcolate manualmente o automaticamente dal renderer di volta in volta in seguito a ogni deformazione.
Il modello, a questo punto, è completo. Le sezioni successive riguardano il "cuore" del progetto, cioè il tentativo di individuare la posa di una mano reale in movimento partendo dai dati ottenuti da un sensore ToF.

Capitolo 4

Stima della posa della mano

Nel Capitolo 1 e nel Capitolo 3 sono stati descritti rispettivamente un sistema sperimentale per l'acquisizione di una mano in movimento e un modello di quest'ultima in grado di riprodurre in maniera non del tutto realistica le deformazioni della pelle dovute alla particolare posa dello scheletro associato.
I dati a disposizione per la stima sono un flusso di frame contenenti ognuno una depthmap opportunamente antidistorta e segmentata. Le misure di profondità sono, quindi, considerate attendibili entro un certo errore e riferite unicamente ai punti della mano. Il modello è articolato e la posa dello scheletro è impostabile secondo necessità specificando la posizione nello spazio tridimensionale del giunto radice e gli orientamenti relativi tra i giunti. Inoltre, avendo specificato un modello del sensore, è possibile generare mappe di profondità sintetiche compatibili a quelle acquisite a patto che il sensore virtuale abbia la stessa caratterizzazione dello SR4000.
Nelle sezioni seguenti è spiegato come sfruttare l'informazione estratta dai dati acquisiti ai fini del tracking della mano. Va, però, tenuto conto di un particolare: il modello della mano non è generico, cioè è aderente alla conformazione fisica della mano del soggetto in esame; il tracking della mano di un individuo diverso impiegando lo stesso modello, quindi, al momento potrebbe fallire. Non si esclude che, in futuro, il modello venga reso adattabile alla mano di un qualsiasi individuo, possibilmente in maniera automatica.

4.1 Scelta della funzione obiettivo

Prima di passare al tracking, vale a dire la stima della traiettoria della mano, è necessario risolvere il sotto problema della stima della posa dato un singolo frame. Riprendendo un concetto accennato all'inizio del Capitolo3, l'idea di base è

spostare e deformare il modello agendo sui parametri della posa affinché la depthmap contenuta nel frame e quella sintetica generata dal sensore virtuale siano teoricamente *identiche*. Essendo, i sensori SR4000 e la sua virtualizzazione, *equivalenti*, in caso di collimazione delle due mappe si può assumere che il modello sia posizionato e orientato allo stesso modo della mano nella scena. Nella pratica è impossibile che le due depthmap coincidano, anche nel caso della migliore stima, in particolar modo quando il modello non riproduce alla perfezione la mano a cui si riferisce. Per questo motivo, è necessario specificare una *metrica* con cui *quantificare* la "bontà" della stima al variare dei valori dei parametri della posa. Nel progetto, l'*ottimizzazione* si pone come obiettivo la minimizzazione del funzionale indicato nell'Equazione 4.1 assegnando un valore opportuno a ogni parametro governabile. La *stima ottima* sarà, pertanto, quella in cui il valore del funzionale assume il *valore minimo*. Nella Sezione 4.2 è, tuttavia, discusso il motivo per cui tale minimo spesso non sia raggiungibile.

$$\begin{gathered}\operatorname{argmin} \sum_{l=1}^{176\cdot144}(r_l(t) - s_l(t))^2 \\ \text{con } r_l(t) \text{ e } s_l(t) \text{ profondità associate ai pixel delle depthmap acquisita} \\ \text{(real) e sintetica (synthetic) rispettivamente e} \\ l = 176 \cdot v + u \text{ linearizzazione dell'indice bidimensionale.}\end{gathered} \tag{4.1}$$

Sviluppando il termine $s_l(t)$, nella Formula 4.2 si evidenzia la sua dipendenza dal vettore dei parametri $\beta(t) = (d_x(t), d_y(t), \theta_1(t), \phi_1(t), \psi_1(t), \theta_2(t), \phi_2(t), \psi_2(t), ..., \theta_b(t), \phi_b(t), \psi_b(t), ..., \theta_N(t), \phi_N(t), \psi_N(t))$

$$\begin{gathered}\operatorname{argmin}_{\beta(t)} \sum_{l=1}^{176\cdot144}(r_l(t) - R_l(LBS(M^0, \beta(0), \beta(t))))^2 \\ \text{con } \beta(0) \text{ posa iniziale e } R_l \text{ profondità del pixel l-esimo dipendente} \\ \text{dal } \textit{rendering} \text{ della scena virtuale in cui è immerso il modello.}\end{gathered} \tag{4.2}$$

Il problema assume, così, la forma della ricerca dei parametri di una *regressione non lineare*. Ogni quadrato è anche chiamato *residuo* ed è dimostrabile che, per motivi statistici, in corrispondenza della soluzione ottima la funzione obiettivo assume il valore minimo.

Conviene soffermarsi momentaneamente sulla singola funzione $R_l(\cdot)$, definita come il valore di profondità associato al pixel l-esimo della depthmap sintetica calcolato tramite il *rendering* della *scena* in cui è immerso il modello. Per **rendering** si intende un processo, o meglio una serie di processi, che portano alla costruzione dell'immagine bidimensionale descrivente la parte del mondo tridimensionale vista dall'occhio di un *osservatore* o di una telecamera che riprende la scena. Semplificando, il *problema del rendering* è decidere il *colore* di ogni pixel che compone l'immagine. La decisione dipende dalle caratteristiche geometriche, di illuminazione e dei materiali di cui sono composti gli oggetti della scena, nonché dall'ottica della telecamera. Per una descrizione completa del processo e dei vari algoritmi coinvolti, si rimanda all'opera di Fusiello[10].
Secondo la funzione obiettivo, il rendering non viene impiegato per decidere il colore di ogni pixel, ma per il calcolo delle *profondità*. Entrando nel dettaglio, dato che lo SR4000 restituisce, per ogni pixel del sensore, la profondità misurata, il rendering è necessario per individuare le coordinate in pixel di ogni punto del modello proiettato sul piano del sensore virtuale. Se nessun punto del modello è proiettato in un dato pixel, a quest'ultimo viene assegnata la massima profondità misurabile[1]. Al posto del colore, quindi, a ogni pixel è associata la profondità di un punto del modello o la profondità massima in assenza di proiezione. Un grande vantaggio dell'impiego della profondità al posto del colore è il fatto che l'illuminazione e i materiali non influiscono in alcun modo sulla profondità; di conseguenza, ai fini del rendering, è possibile impostare il modello di illuminazione meno oneroso computazionalmente, o alternativamente non specificare alcuna fonte luminosa per evitare le numerose risoluzioni dell'*equazione della radianza*.
Da notare che per *punto del modello* non si intende necessariamente un *vertice* della mesh, ma un generico punto che può giacere sulla regione individuata da una faccia. Questa particolare è la ragione che ha indotto all'abbandono delle librerie openCV, nonostante siano apprezzate in Computer Vision: esse, infatti, non si occupano del rendering, ma solo della proiezione di una sequenza di punti tridimensionali in uno spazio bidimensionale secondo una matrice di proiezione specificata dall'utente. OpenCV non tiene conto, quindi, di altri due problemi affrontati dal rendering:

- Clipping
- Auto occlusioni

[1] Si ricorda essere pari a 5m

Il primo riguarda la determinazione degli oggetti o delle parti di un oggetto che non devono essere disegnate in quanto esterne al *volume visuale* della telecamera o *frustum*, mentre il secondo riguarda la determinazione degli oggetti o parti di un oggetto che non devono essere disegnate in quanto coperte da altri oggetti o da parti dell'oggetto stesso. Riassumendo, i metodi implementati in openCV proiettano punti tridimensionali anche se esterni al volume di vista o se sono occlusi dalle facce dello stesso oggetto, per cui non sono utilizzabili per la simulazione del funzionamento dello SR4000. A rafforzare tale affermazione è sufficiente il seguente esempio: assumendo che il sensore virtuale stia riprendendo il modello di una mano orientata con il palmo diretto verso l'obiettivo della telecamera, il metodo di openCV proietta indiscriminatamente sia i vertici[2] del palmo sia quelli del dorso che non si trovano sullo stesso *raggio ottico* che attraversa un vertice del palmo. Un renderer, invece, proietterebbe anche alcuni punti giacenti all'interno delle facce del palmo, ma non proietterebbe alcun vertice del dorso perché occluso dalle facce del palmo.
Data la complessità del processo di rendering, non è possibile esprimere la generica funzione $R_l(\cdot)$ per via analitica, impedendo, così, di sfruttare l'eventuale peculiarità della sua *forma* in fase di calcolo. Nella Sezione 4.1.1 è illustrata brevemente la soluzione adottata per il calcolo automatizzato di $R_l(\cdot)$.

4.1.1 Esecuzione del rendering

Il rendering è un processo oneroso computazionalmente, in particolar modo se la scena contiene oggetti complessi geometricamente e modellati da mesh con un numero elevato di vertici e poligoni. Gli esperti in Computer Graphics sono soliti affidare la gestione della grafica a GPU sempre più performanti, di cui sono dotati i nuovi acceleratori grafici[3], per destinare la CPU a tutt'altro genere di computazioni.
Inizialmente è stata tentata l'implementazione di un renderer *semplificato* e *minimale* che riducesse al minimo il carico sulla CPU e permettesse un maggiore controllo sulle elaborazioni dei dati. In seguito a numerosi tentativi i cui esiti non sono stati favorevoli, è stata valutata la possibilità di affidare il rendering del modello della mano alle librerie VTK[16] (Visualization Toolkit), sviluppate da un team di esperti come supporto a software per la visualizzazione e l'elaborazione di dati 3D.
Le librerie VTK, in realtà, non implementano un renderer, ma fungono da in-

[2] Qui si intende un vertice della mesh, non un generico punto giacente su una faccia.
[3] Più comunemente chiamati *schede video*.

terfaccia con le librerie openGL[4] per la visualizzazione dei dati tridimensionali; queste ultime sfruttano appieno e nativamente la potenza di calcolo della GPU. Generare e visualizzare una scena manualmente con i comandi openGL è complicato e richiede diverse righe di codice a corredo, mentre in VTK sono sufficienti poche righe di codice e la chiarezza è maggiore a scapito di un leggero *overhead* nelle prestazioni. In più, VTK offre alcuni metodi per l'importazione e l'esportazione di mesh memorizzate in file con le estensioni più usate nella modellazione 3D, quali PLY e OBJ. Tale funzionalità non è stata, però, utilizzabile in quanto l'ordinamento dei vertici deve coincidere con quello dei pesi calcolati in Maya, obbligando la produzione degli script di esportazione citati nel capitolo precedente. Un'altra caratteristica che ha motivato l'uso delle librerie è la possibilità di eseguire il *rendering off-screen*, invece che *on-screen*, vale a dire, il programmatore è libero di decidere se l'immagine costruita debba essere disegnata sullo schermo o meno. Tale funzionalità evita di dover visualizzare la mesh a ogni passo di ottimizzazione intermedio, operazione preliminare all'estrazione di ogni depthmap. Tuttavia, essendo il codice sviluppato nel progetto ancora in fase di debug, si è deciso di non avvalersi di questa possibilità ma di osservare l'andamento dell'ottimizzazione passo-passo in una finestra dedicata.
Prima di terminare la Sezione corrente, rimane il delicato problema dell'impostazione della telecamera, nella fattispecie il sensore virtuale; infatti, come già detto in precedenza, l'immagine costruita dal renderer è fortemente influenzata dal *punto di vista* della telecamera virtuale e dai suoi parametri ottici definiti nel modello pin-hole. La configurazione della telecamera non è semplice, perché è necessario tenere conto della definizione dei vari sistemi di riferimento scelta dagli autori della libreria e che differiscono dalla specifica riportata in Figura 1.7. Essendo VTK costruita su openGL, le convenzioni adottate appartengono a quest'ultima.
Il modello della telecamera in openGL è analogo al modello pin-hole ideale, ma introduce alcuni limiti nella visualizzazione: definisce un volume visuale a forma di tronco di piramide a base rettangolare e un range di distanze che restringono il volume della scena rappresentabile dalla telecamera. Gli oggetti al di fuori del volume non sono considerati nel rendering, mentre lo sono quelli compresi completamente suo interno o la loro parte interna se intersecano i piani che delimitano il frustum. La proiezione non avviene direttamente su schermo (o sul piano immagine), ma in due passi sequenziali: nel primo passo il volume di vista piramidale

[4] Alternativamente, in fase di compilazione è possibile sostituirle con le librerie Directx di Microsoft.

viene mappato in un volume normalizzato a forma di cubo, detto *canonico*; il motivo è che tale trasformazione semplifica l'operazione di clipping. Il secondo passo, invece, consiste in una proiezione ortografica normalizzata seguita da una trasformazione che adatta quest'ultima alla finestra adibita alla visualizzazione. Ancora più importante è l'effetto della catena di trasformazioni sulla profondità: anch'essa viene mappata nel volume canonico, tramite una trasformazione fortemente *non lineare*. La non linearità della trasformazione è spesso fonte di *errori di precisione*, infatti, il frame buffer che ospita la profondità, detto *z-buffer*, ha una profondità limitata[5] e per costruzione riporta una maggiore precisione nella memorizzazione delle profondità degli oggetti più vicini rispetto a quelle degli oggetti lontani. La profondità memorizzata degli oggetti lontani rischia, perciò, di differire sensibilmente da quella reale, generando artefatti quali la compenetrazione di oggetti disposti nella scena a profondità diverse. Il difetto citato, tuttavia, è meno evidente e in genere trascurabile negli acceleratori grafici moderni con z-buffer di profondità pari a 24 o 32 bit. La non linearità della trasformazione della profondità, inoltre, impedisce l'utilizzo diretto del contenuto dello z-buffer come depthmap sintetica da confrontare con una acquisita, per il semplice fatto che lo z-buffer memorizza i valori di *pseudo-profondità* e non quelli di profondità. Fortunatamente, la depthmap è ricostruibile a partire dallo z-buffer risolvendo per ogni pixel di quest'ultimo l'Equazione 4.3, funzione delle due soglie z_{near} e z_{far}.

$$z_{lin}(144 - v, u) = \frac{z_{near} z_{far}}{z_{far} - z_{buff}(v, u)(z_{far} - z_{near})}; \tag{4.3}$$

La configurazione della telecamera è ulteriormente complicata dall'impostazione dei suoi parametri intrinseci, ottenuti dalla calibrazione dello SR4000. Riprendendo un concetto espresso nel Capitolo 1, si ricorda che il sensore Time-of-Flight e la sua virtualizzazione possono ritenersi equivalenti quando condividono gli stessi valori per i parametri intrinseci. Prima di regolare l'ottica della camera virtuale, è necessario impostare i sui parametri estrinseci e la posizione del centro di proiezione affinché i sui sistemi di riferimento siano orientati compatibilmente alla convenzione adottata in Figura 1.7. Una lista dei principali parametri modificabili in openGL/VTK è la seguente:

[5] Si intende il numero di bit per un generico valore di profondità memorizzabile

- posizione nello spazio del punto focale (intersecato dal piano immagine);
- posizione nello spazio della telecamera;
- direzione view-up della telecamera;
- distanza (positiva) dei piani di clipping z_{near} e z_{far} dal piano immagine;
- angolo visuale verticale o orizzontale;
- rapporto d'aspetto.

In particolare, nel codice, per evitare errori sono stati seguiti i passi sotto indicati.

1) impostazione $z_{near} = 10$mm e $z_{far} = 5$m;
2) rotazione della telecamera di 180° attorno al view-up vector che coincide con l'asse Y del sistema globale (metodo Azimuth());
3) impostazone di (0, 0, 0) come posizione della telecamera e 10mm come *distanza focale*.

I parametri intrinseci non sono, purtroppo, modificabili manualmente in openGL/VTK, in quanto nessun metodo della classe VTK che implementa la telecamera contempla le due *distanze focali* o le coordinate del *punto principale*. L'unica alternativa è agire direttamente sulla matrice di proiezione della camera virtuale per impostare i parametri mancanti, un'operazione permessa in openGL ma non in VTK. La ricerca di una soluzione ha richiesto una profonda analisi del codice VTK per individuare il punto in cui viene generata la matrice di proiezione della telecamera, individuato nella classe vtkOpenGLcamera. Quest'ultima offre al programmatore due punti di ingresso nella filiera del rendering in cui inserire eventualmente due matrici di trasformazione aggiuntive; il punto di ingresso di interesse è quello che permette l'inserimento di una matrice di trasformazione 4x4 da *premoltiplicare*[6] alla proiezione secondo openGL. Definendo P^* come la matrice di proiezione voluta e conoscendo la matrice di proiezione P_{VTK} generata automaticamente dal codice della libreria, si evince che inserendo nella filiera la matrice $P^* P_{VTK}^{-1}$ si ha come risultato finale la forzatura di P^* come matrice di proiezione della camera virtuale. P^*, tuttavia, non coincide con P_{openCV} sia

[6]Comporta l'esecuzione della trasformazione *dopo* la trasformazione corrente.

perché openGL richiede matrici 4x4 e non 3x3, sia perché la concezione di "proiezione" in openGL differisce sensibilmente da quella in openCV. Il problema è noto tra i programmatori che utilizzano entrambe le librerie, ed è facilmente risolvibile costruendo P^* come indicato nell'Equazione 4.4.

$$P^* = \begin{bmatrix} 2\frac{f_x}{176} & 0 & 2\frac{c_x}{176} - 1 & 0 \\ 0 & 2\frac{f_y}{144} & 2\frac{c_y}{144} - 1 & 0 \\ 0 & 0 & -\frac{z_{far}+z_{near}}{z_{far}-z_{near}} & -2\frac{z_{far}z_{near}}{z_{far}-z_{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{4.4}$$

In assenza di errori di programmazione, il sensore virtuale correttamente impostato è impiegabile in fase di ottimizzazione per la sintesi di depthmap il cui confronto con le mappe acquisite influisce sulla regolazione dei parametri della posa.

4.2 Scelta dell'algoritmo di ottimizzazione

Nella Sezione 4.1 il problema della stima della posa per un frame è stato definito come un problema di ottimizzazione avente come ingresso la depthmap riferita al frame in esame e come uscita un'assegnazione di valori per la posizione del giunto radice e per le rotazioni locali dei giunti attorno agli assi coordinati. L'assegnazione deve essere *consistente*[7], vale a dire non violare alcun vincolo imposto e indicato nella Tabella 3.3, e deve essere tale da *minimizzare* il valore della metrica definita nell'Equazione 4.1.
La stima della posa, secondo la funzione obiettivo sopracitata, si configura come un problema di ottimizzazione di una funzione non lineare. In letteratura gli algoritmi atti a risolvere tale tipologia di problemi sono classificabili in tre categorie:

- algoritmi a discesa di gradiente (gradient descent)
- algoritmi a *regione di confidenza* (trust region)
- algoritmi ibridi

Tutti gli algoritmi delle tre tipologie hanno in comune il fatto di cercare *iterativamente*, a ogni passo, l'incremento di ogni parametro rispetto al valore al passo

[7] In analogia ai problemi Constraint Satisfaction Problems (CSP).

corrente che comporti il massimo decremento del valore del *modello* della funzione obiettivo. Per *modello della funzione obiettivo* si intende un'approssimazione di quest'ultima tramite espansione secondo serie di Taylor troncata in genere al secondo termine. In questo caso il modello si dice *quadratico* e assume la forma indicata nell'Equazione 4.5

$$\begin{gathered} p, x_c \in \mathbb{R}^n \\ f : \mathbb{R}^n \to \mathbb{R} \\ m_c(x_c + p) \triangleq f(x_c) + \nabla f(x_c)^T p + \tfrac{1}{2} p^T \nabla^2 f(x_c) p \\ \text{con } p^T \nabla^2 f(x_c) p > 0 \text{ (semidefinita positiva)} \end{gathered} \tag{4.5}$$

Gli algoritmi a discesa di gradiente cercano il decremento del modello quadratico lungo una *direzione discendente* a partire dal valore attuale assunto da ogni parametro. Il nome "discesa di gradiente" deriva dalla dimostrabilità che la direzione lungo la quale si verifica il massimo decremento è la direzione opposta alla direzione del gradiente della funzione obiettivo. Più in generale, vale la definizione seguente:

$$p, x_c \in \mathbb{R}^n, \|p = 1\|, p \text{ is } \textit{descent direction} \text{ from } x_c \Rightarrow \nabla f(x_c)^T p < 0 \tag{4.6}$$

Gli algoritmi a regione di confidenza sono molto simili agli algoritmi a discesa di gradiente, in quanto differiscono solo per la *lunghezza* massima del passo di ricerca: i primi cercano il massimo decremento lungo una direzione discendente (non necessariamente la direzione opposta al gradiente) *entro una sfera in* $\mathbb{R}^n$ di raggio limitato, mentre i secondi non hanno un limite sulla lunghezza del passo. Gli algoritmi ibridi, invece, adottano l'una o l'altra strategia a seconda della *vicinanza* della stima attuale all'ottimo.
Avendo espresso la funzione obiettivo nell'Equazione 4.1 come somma di quadrati di residui, il problema si configura come un'ottimizzazione ai *minimi quadrati*. Data la particolare forma della funzione obiettivo, indicata nell'Equazione 4.7, l'algoritmo per l'ottimizzazione è il metodo di Levenberg-Marquardt. Si tratta di una variante dell'algoritmo di Gauss-Newton per l'ottimizzazione di una generica funzione non lineare con approccio trust-region, e il procedimento ricalca in parte quello impiegato nella soluzione di un sistema di equazioni non lineari.

$$S(\beta) = \sum_{i=1}^{N} (y_i - f(x_i, \beta))^2 = \|Y - F(X, \beta)\|^2$$
dove y_i è la y-esima misura, $f(x_i, \beta)$ il modello della curva funzione dei parametri da ottimizzare,
$X = [x_1, x_2, ..., x_N]^T$, $Y = [y_1, y_2, ..., y_N]^T$ e $F(X, \beta) = [f(x_1, \beta), f(x_2, \beta), ..., f(x_N, \beta)]^T$. (4.7)

Per una trattazione approfondita si rimanda a [26, 9], mentre ai fini della motivazione delle scelte effettuate e della comprensione dei risultati illustrati successivamente si riportano solo alcuni punti fondamentali dell'algoritmo. Il primo riguarda il modello della curva da adattare al vettore delle misure Y, specificato nell'Equazione 4.8.

$$m_c(x_c + \beta) \triangleq \sum_{i=1}^{N} y_i - f(x_i, \beta) - J_i\delta)^2 = \|Y - F(X, \beta) - J\delta\|^2$$
dove J_i è la i-esima della matrice jacobiana J (ovvero il gradiente)
di $f(x_i, \beta)$ e δ il vettore degli incrementi per il vettore dei parametri β. (4.8)

Ancora più importante è il calcolo di δ a ogni iterazione, indicato nell'Equazione 4.9, dove emerge la dipendenza del risultato da un parametro dimensionato nella Sezione 4.3.2. Il calcolo consiste nella risoluzione di un sistema di equazioni lineari, e la sua efficienza è migliorabile applicando alcune tecniche di scomposizione delle matrici. Il parametro μ assolve due funzioni: evita la possibile non-invertibilità della matrice J^TJ e regola indirettamente la lunghezza del passo della trust-region; un suo corretto dimensionamento è, quindi, indispensabile.

$$(J^TJ + \mu J^TJ)\delta = J^T[Y - F(X, \beta)]$$
dove μ è il parametro di *damping*. (4.9)

Il secondo punto fondamentale deriva dalle Equazioni 4.8, 4.9 e riguarda il calcolo della matrice jacobiana, o equivalentemente il calcolo del singolo gradiente della funzione di β. I gradienti calcolati analiticamente, ove possibile, comportano in genere una convergenza rapida al minimo locale; tuttavia, se il calcolo non è possibile o risulta eccessivamente oneroso, l'approssimazione alle differenze finite è spesso una valida alternativa ma con il rischio di una convergenza più *lenta*. Come accennato nella Sezione 4.1.1, a causa dell'impossibilità di specificare il modello (intendendo $f(x_i, \beta)$) analiticamente per via del rendering, non è stato possibile calcolare il gradiente ma solo approssimarlo alle *differenze finite*. Nell'Equazione 4.10 sono confrontate due possibili approssimazioni del gradiente:

- approssimazione alle differenze in avanti
- approssimazione alle differenze centrali

Il calcolo di $J \in \mathbb{R}^{n \times m}$ richiede per la prima m(n+1) valutazioni della funzione, mentre per la seconda ne richiede il doppio comportando, però, una migliore approssimazione.

$$\underset{a)\text{differenze in avanti}}{J_{i,j} = \frac{f_i(\beta + e_j\Delta) - f_i(\beta)}{\Delta}} \quad \underset{b)\text{differenze centrali}}{J_{i,j} = \frac{f_i(\beta + e_j\Delta) - f_i(\beta - e_j\Delta)}{2\Delta}}$$

dove Δè la lunghezza del passo di derivazione e $e_j$il j-esimo vettore canonico di $\mathbb{R}^m$. (4.10)

Ai fini del progetto si è ritenuto opportuno non implementare l'algoritmo nel codice, ma di avvalersi di una sua implementazione di libero utilizzo codificata nella libreria Levmar2.5[20]. Essa si appoggia alla libreria Lapack[2] per il calcolo algebrico efficiente, la quale è realizzata rispettando le specifiche BLAS[17]. Levmar2.5 raccoglie diverse varianti dell'algoritmo di Levenberg-Marquardt, classificabili in base al tipo di dati utilizzato, dal tipo di calcolo della matrice Jacobiana e dall'eventuale tipo di vincoli imponibili. Per chiarezza si riportano qui sotto solamente le caratteristiche principali della libreria.

Tipo di dati

- virgola mobile
- doppia precisione

Matrice Jacobiana

- analitica
- approssimata alle *differenze finite*
  - differenze *forward*
  - differenze *centrali*

Tipo di vincoli

- nessuno (ottimizzazione unconstrained)
- lineari (Ax = b)
- disuguaglianze (Ax $\geq$ b)

- limiti superiori e inferiori (box constraints)
- compresenza di due o più tipi di vincoli precedenti

Il processo di ottimizzazione è quasi del tutto *trasparente* al programmatore, infatti, oltre all'impostazione di alcuni parametri dell'algoritmo, è richiesta solamente la specifica di un metodo che calcoli per ogni residuo i-esimo il valore $f(x_i, \beta)$ ed eventualmente un metodo che calcoli il gradiente del modello per ogni residuo.
Un'altra peculiarità della libreria riguarda l'approssimazione della matrice jacobiana alle differenze finite: ove possibile, al posto di ricalcolare l'intera matrice a ogni iterazione, l'implementazione dell'algoritmo impiega l'*approssimazione di Broyden*, che richiede solo m rivalutazioni della funzione per aggiornare J ma può essere usata al massimo n volte consecutivamente. Per completezza, tale approssimazione è riportata nell'Equazione 4.11. Per una trattazione approfondita si rimanda a [8, 1].

$$J_{k+1} = J_k + \frac{(F(\beta_k + \delta_k) - F(\beta_k) - J_k * \delta_k)\delta_k^T}{\|\delta_k\|^2} \tag{4.11}$$

4.3 Calibrazione parametri di ottimizzazione

I risultati dei primi test di ottimizzazione, che consideravano il problema nella sua interezza e non comportavano la modifica delle impostazioni di default dell'algoritmo erano, come aspettato, diversi da quelli desiderati. La causa del fallimento era difficile da determinare, data l'elevata complessità del problema di ottimizzazione dovuta all'elevato numero di parametri da ottimizzare e di vincoli da rispettare. Si è deciso, pertanto, operare una prima semplificazione del problema:

- sostituzione delle depthmap acquisite con depthmap sintetiche;
- ottimizzazione di al più tre parametri contemporaneamente a scelta tra le coordinate della posizione del giunto radice nello spazio;
- perturbazioni limitate (pochi millimetri);
- rimozione dei vincoli;
- stima iniziale dei parametri da ottimizzare vicina all'ottimo;

Il motivo di tali scelte è che, assumendo l'implementazione dell'algoritmo corretta e operando le semplificazioni sopracitate, l'algoritmo teoricamente deve convergere al *minimo globale* con un errore di stima pressoché nullo. I risultati dei primi esperimenti di ottimizzazione in seguito alla semplificazione del problema erano ancora instabili e inaccettabili in quanto spesso l'algoritmo divergeva o terminava prematuramente senza modificare i valori iniziali dei parametri.
Una delle cause di fallimento più importanti e ipotizzate è la probabile convergenza a un qualche *minimo locale*, dato che la terminazione dell'algoritmo segnalata era spesso dovuta alla variazione pressoché nulla dei valori dei parametri da un'iterazione alla successiva. Un'altra causa ipotizzata è un problema di *scala* dei parametri: il difetto si manifestava con variazioni di pochi millesimi o decimi di grado degli angoli, con la conseguente variazione nulla del valore della funzione costo da un'iterazione alla successiva. Dopo un approfondito studio della letteratura, si è scoperta la possibilità di rimuovere tale difetto ottimizzando parametri preventivamente scalati, in modo che le variazioni degli angoli e delle distanze siano omogenee. Il difetto, però, non affligge il problema semplificato perché i parametri considerati hanno tutti lo stesso ordine di grandezza.
L'ipotesi di convergenza a un minimo locale è stata, quindi, ritenuta la più plausibile ed è stata a lungo studiata svolgendo diversi test illustrati nelle sezioni successive. Lo scopo degli esperimenti era scoprire l'eventuale esistenza di un'opportuna configurazione dei parametri μ e Δ che permetta all'algoritmo di convergere all'ottimo globale quando la stima iniziale è prossima a quest'ultimo, indipendentemente dalla posizione reale della mano nello spazio. Prima dei test di ottimizzazione, però, è stato condotto uno *studio di fattibilità della risoluzione del problema*. I risultati sono commentati nella Sezione 4.3.1.

4.3.1 Studio di fattibilità

La prima serie di test condotti ha avuto come scopo lo studio della funzione scelta come funzione obiettivo, per determinare se essa fosse realmente adatta come funzione di ottimizzazione. Più specificatamente, le prove consistevano nel generare una mappa sintetica a partire dal modello con arbitrariamente posizionato e orientato all'interno del frustum, in vece della depthmap acquisita, e calcolare il valore della funzione obiettivo al variare di uno o al più due parametri del modello contemporaneamente (non più di due perché altrimenti la funzione non è rappresentabile graficamente). Il risultato teorico atteso era la presenza di un unico minimo globale in corrispondenza di un'assegnazione di valori ai parametri pari a quelli della posa iniziale, e un aumento del valore della funzione proporzionale

alle perturbazione dei parametri dalla posa iniziale.
Dato l'elevato numero di parametri e l'ancor più elevato numero di loro possibili combinazioni da valutare, lo studio è stato condotto limitatamente alla posizione del giunto radice e alla sua rotazione.
I grafici riportati nella Sezione A.1 sono riferiti a un modello con circa 10^4 poligoni e circa $5 \cdot 10^3$ vertici ottenuto tramite la procedura di semplificazione descritta nella Sezione 3.1.6 e si è visto che coincidono con quelli riferiti al modello originale con circa 10^5 poligoni e circa $5 \cdot 10^4$ vertici. Per questo motivo, da ora in poi il modello considerato è quello con dimensionalità ridotta.

Test 1: studio della fattibilità delle traslazioni

La posa iniziale scelta per il modello nel test è descritta in seguito nella Sezione 4.4, mentre la posizione del giunto radice è stata impostata arbitrariamente a [-0.3m, 0.1m, 1m].
Dai grafici in Figura A.1 si nota che la funzione obiettivo, al variare di uno solo dei parametri, decresce avvicinandosi al minimo globale, che si vede correttamente coincidere con la posizione iniziale arbitrariamente impostata. L'assoluta "piattezza" della funzione in alcuni intervalli al variare di X o Y è dovuta all'assenza della mano all'interno del frustum. I due intervalli in cui la funzione assume il valore massimo sono problematici in quanto il *ripple* causa continui lievi incrementi e decrementi del valore della funzione. Tali variazioni portano a innumerevoli inversioni della direzione del gradiente che possono limitare la ricerca di un minimo in quella regione fino alla terminazione prematura dell'algoritmo per il superamento del massimo numero di iterazioni, o ancor peggio spostare il modello fuori dallo schermo. Il ripple è causato dall'elevata risoluzione del modello che cattura fedelmente le pieghe della pelle, variando sensibilmente anche per spostamenti lievi il set dei vertici proiettati. Alcuni vertici, infatti, sono occlusi da altri o meno a seconda dell'orientamento del modello rispetto alla telecamera virtuale, e il motivo è da ricercare nella valutazione della loro visibilità che coinvolge, tra l'altro, il prodotto interno tra le loro normali e la direzione della telecamera. Il valore massimo della funzione lo si ha quando la mano si trova all'interno del volume di vista ma in una posizione *distante* dall'ottimo. Le pendenze che raccordano le zone assolutamente piatte con quelle piatte affette da ripple sono problematiche perché possono "ingannare" l'algoritmo portandolo a spostare la mano fuori dallo schermo durante la ricerca del minimo. Anche i due picchi nell'intorno di -0.4 e -0.1 nel grafico A.1.a sono pericolosi: possono portare a una convergenza locale e non globale. Infine, nel grafico riferito al parametro

Z si notano due particolarità: il valore della funzione ha un picco in prossimità di z = 0 e diminuisce leggermente a partire da una certa profondità. La prima particolarità è dovuta al fatto che nel modello pinhole specificato in Figura 1.7 il piano immagine è posto davanti al punto focale, mentre la seconda è dovuta al calo del livello del dettaglio e alla diminuzione del numero di punti della mano proiettati all'aumentare della distanza dall'obiettivo.
Per i grafici nelle Figure A.2, A.3 e A.4 valgono le stesse considerazioni fatte per la Figura A.1, e la funzione obiettivo mostra nuovamente l'andamento atteso: in particolar modo dalle curve di livello, si vede che la funzione ha nuovamente un minimo globale in corrispondenza dell'ottimo. Si può affermare, quindi, che la funzione obiettivo scelta è adatta all'ottimizzazione di almeno due parametri congiuntamente.
Un'altra proprietà interessante che si nota dai grafici è che fissando ipoteticamente il valore di uno dei due parametri di una delle coppie precedenti, il grafico bidimensionale ottenuto ha la *stessa forma* del grafico generato al variare del rimanente parametro[8]. Come si vedrà in seguito, questa particolarità è alla base della soluzione adottata per dimensionare i parametri dell'algoritmo di ottimizzazione.

Test 2: studio della fattibilità delle rotazioni

La posa iniziale della mano in questo test è nuovamente quella indicata nella Sezione 4.4, mentre la posizione del giunto radice è stata impostata a [0, 0, 1m]; per chiarezza, si specifica anche l'orientamento del giunto radice: [0, 180°, 0]. Analogamente al test precedente, sono state studiate prima le rotazioni del giunto radice attorno a un unico asse coordinato del sistema globale e in seguito le rotazioni attorno a due assi coordinati.
Per i grafici in Figura A.5 valgono le stesse considerazioni riguardo l'ottimalità illustrate nella Sezione 4.3.1. I grafici riferiti alle rotazioni attorno X e attorno Y mostrano un certo grado di simmetria, e unitamente al grafico riferito alle rotazioni attorno Z, si può notare la presenza di molti minimi locali che portano l'algoritmo di ottimizzazione a divergere dall'ottimo se la stima dell'orientamento iniziale non è accurata.
Nuovamente, i grafici nelle Figure A.6, A.7 e A.8 mostrano alti gradi di simmetria e si vede permanere la validità della proprietà scoperta nel Test 1 anche per le rotazioni del giunto radice. I risultati di questo test sono un'ulteriore prova dell'adeguatezza della funzione descritta nell'Equazione 4.1 come funzione obiettivo.

[8] A patto di fissare il valore in un intorno dell'ottimo, ovviamente.

4.3.2 Dimensionamento dei parametri dell'algoritmo

I risultati dei test riportati nella Sezione 4.3.1 sono prova dell'adeguatezza della funzione descritta nell'Equazione 4.1 come funzione obiettivo, almeno nell'ambito del problema semplificato in esame. Potenzialmente, quindi, data una "buona" stima iniziale della posizione e dell'orientamento del giunto radice, l'algoritmo è in grado di restituire una stima accurata della posa effettiva.
Come citato nella Sezione 4.2, l'algoritmo di Levenberg-Marquardt, oltre al valore impostato per alcune soglie, dipende fortemente dai parametri μ e Δ. Data la complessità della teoria sottostante e la conseguente impossibilità di determinare a priori i migliori valori da impostare per i parametri dell'algoritmo, sono stati condotti alcuni test di simulazione di un possibile *tracking* per determinare il valore di Δ che minimizzi l'errore di stima. Il valore di μ, invece, è stato fissato a 10^{-20} perché alcuni test hanno mostrato che l'effetto della sua variazione è trascurabile rispetto all'effetto della variazione di Δ.
Analogamente allo studio di fattibilità della sezione precedente, è stata considerata la sola pura traslazione della mano. I primi test simulano il tracking lungo un'unica coordinata a scelta tra X, Y e Z, mentre i successivi simulano il tracking lungo una diagonale ottenuta incrementando una coppia o tutte e tre le coordinate dello stesso valore. Il motivo alla base di tale strutturazione dei test è lo studio del comportamento dell'algoritmo di ottimizzazione in vista di un futuro tracking reale, dove i più semplici ed elementari gesti eseguibili da un utente sono la traslazione della mano lungo una direzione o un movimento leggermente più complesso costituito dalla composizione di spostamenti elementari.
Ogni test è stato ripetuto per ogni diversa lunghezza del passo di tracking, con l'idea che passi di pochi millimetri simulino gli spostamenti di una mano in presenza di un framerate di acquisizione elevato. Al contrario, passi di diversi centimetri non servono a simulare un tracking reale, ma piuttosto a valutare la capacità dell'algoritmo di portare a termine con successo la calibrazione iniziale della mano. Il baricentro della mano adottato come stima della posa iniziale, infatti, può distare anche fino a 10 cm dall'ottimo, ed è indispensabile che l'algoritmo sia in grado di convergere sia in fase di tracking sia in fase di calibrazione.

Test 3: simulazione di un tracking *fittizio* in una direzione

Il termine "fittizio" sta a indicare che il tracking simulato nei seguenti test non è un vero tracking, ma una sua versione *semplificata*. Mentre nel tracking ordinario, infatti, la stima iniziale della posizione per l'ottimizzazione del frame corrente coincide con la stima finale dell'ottimizzazione del frame precedente, nel tracking semplificato in esame la stima iniziale viene fatta coincidere con la posizione della mano nel frame precedente. In altre parole, il tracking semplificato non è altro che il tracking ordinario in cui si assume che la stima della posizione nel frame precedente sia *esatta*. I motivi per valutare tale semplificazione sono due: studiare il comportamento dell'algoritmo in assenza di *divergenza* dall'ottimo causata da una sequenza di stime non accurate, e il fatto che se l'algoritmo non funziona in un caso semplificato è altamente improbabile che funzioni in un caso generico. L'idea è tentare di risolvere prima il problema più semplice e, in caso di successo, tentare di risolvere problemi via via più complessi fino a risolvere il problema nella sua interezza.
La Tabella4.1 raccoglie per ognuno dei parametri e ogni lunghezza del passo di tracking fittizio il valore di Δ che minimizza l'errore di stima, secondo i grafici riportati nella Sezione A.3.

Step [mm]	Δ for X	Δ for Y	Δ for Z
2	10^{-1}	10^{-2}	10^{-2}
5	10^{-1}	10^{-2}	10^{-2}
10	10^{-1}	10^{-2} o 10^{-1}	10^{-2} o 10^{-1}
20	10^{-1} o 10^{-2}	10^{-2}	10^{-1} o 10^{-2}
50	10^{-1}	10^{-1}	10^{-2}
80	10^{-1}	10^{-1}	10^{-3} o 10^{-2}

Tabella 4.1: Dimensionamento di Δ per un tracking fittizio lungo una sola direzione.

Dai grafici si nota che l'algoritmo di ottimizzazione è in grado di restituire stime molto accurate con un opportuno dimensionamento di Δ, e dalla Tabella 4.1 si evince che l'errore di stima di X è minimo per $\Delta = 1e^{-1}$ indipendentemente dalla lunghezza del passo di tracking, mentre gli errori di stima di Y e Z sono minimi con $\Delta = 1e^{-2}$ per passi di tracking ridotti e con $\Delta = 1e^{-1}$ per passi più ampi.

Test 4: simulazione di un tracking reale in una direzione

Il test è identico a quello descritto nella Sezione 4.3.2, tranne che nell'impostazione della stima iniziale della posa in ogni frame: come già detto, dovendo l'esperimento simulare un possibile tracking di una mano reale, la stima iniziale della posa coincide con la stima finale della posa per il frame precedente. Quest'ultima può non essere accurata portando, così, l'algoritmo a divergere rapidamente dall'ottimo.
La Tabella 4.2 raccoglie per ognuno dei parametri e ogni lunghezza del passo di tracking il valore di Δ che minimizza l'errore di stima, secondo i grafici riportati nella Sezione A.4. Dalla tabella si evince che per l'impostazione del valore di Δ valgono le stesse considerazioni effettuate nella Sezione 4.3.2, con la differenza che per minimizzare l'errore di stima di Z non è necessario tenere conto della lunghezza del passo di tracking. I grafici della Sezione A.4 mostrano anche che l'impostazione

Step [mm]	Δ for X	Δ for Y	Δ for Z
2	10^{-1}	10^{-2}	10^{-2}
5	10^{-1}	10^{-2}	10^{-2}
10	10^{-1}	10^{-2} o 10^{-1}	10^{-2} o 10^{-1}
20	10^{-1}	10^{-2}	10^{-1} o 10^{-2}
50	10^{-1}	10^{-1}	10^{-1} o 10^{-2}
80	10^{-1}	10^{-1}	10^{-1} o 10^{-2}

Tabella 4.2: Dimensionamento di Δ per un tracking reale lungo una sola direzione.

di un valore eccessivamente ridotto per Δ porta l'algoritmo alla divergenza, un risultato apparentemente insolito. Infatti, si potrebbe erroneamente pensare che la stima migliori riducendo Δ perché così l'approssimazione del gradiente riportata nell'Equazione 4.10 si avvicina maggiormente alla definizione di *derivata*, e di conseguenza la *direzione di ricerca* dell'ottimo "segue il profilo della funzione". In realtà, va tenuto conto che il vettore degli incrementi δ non è calcolato sommando Δ a ogni parametro, ma tramite l'Equazione 4.9, per cui, anche se Δ è relativamente elevato, può succedere che gli incrementi effettivi siano esigui. Inoltre, gli elevati errori di stima per Δ ridotto sono dovuti probabilmente alla convergenza a uno dei tanti minimi locali di cui è disseminata la funzione; un valore di Δ elevato, invece, almeno nei primi passi dell'algoritmo, permette di evitare gran parte dei minimi locali e di avvicinarsi rapidamente all'ottimo.

Test 5: simulazione di un tracking *fittizio* lungo una diagonale nel piano X-Y

Il test corrente ha lo scopo di valutare il comportamento dell'algoritmo di ottimizzazione nel caso di spostamenti della mano leggermente più complessi, nella fattispecie lungo una diagonale che giaccia in un piano parallelo al piano immagine del sensore. L'esperimento serve a saggiare la capacità dell'algoritmo di seguire gli spostamenti mano mentre compie un gesto che non richieda un movimento in profondità, per esempio lo spostamento di una finestra. In caso di successo, è possibile studiare movimenti ben più complessi che coinvolgano anche la profondità.
La Tabella 4.3 raccoglie per la coppia di parametri X-Y e ogni lunghezza del passo di tracking fittizio il valore di Δ che minimizza l'errore di stima di X e il valore che minimizza l'errore di stima di Y, secondo i grafici riportati nella Sezione A.5.

Step [mm]	Δ for X	Δ for Y
2	10^{-2}	10^{-2}
5	10^{-2}	10^{-2}
10	10^{-2}	10^{-2}
20	10^{-1}	10^{-1}
50	10^{-1}	10^{-1}
80	10^{-1}	10^{-1}

Tabella 4.3: Dimensionamento di Δ per un tracking fittizio sul piano X-Y.

Dalla Tabella 4.3 si estrae un risultato interessante: in caso di ottimizzazione congiunta, gli errori di stima di X e di Y sono minimizzati entrambi per lo stesso valore di Δ, e il suo dimensionamento segue gli stessi criteri illustrati nella Sezione 4.3.1. Anche la distribuzione degli errori di stima è simile a quella risultante dall'ottimizzazione dei singoli parametri.

Test 6: simulazione di un tracking reale lungo una diagonale nel piano X-Y

Il test ha le stesse finalità del test descritto nella Sezione 4.3.2 ma con la simulazione di un tracking reale al posto di uno fittizio. La Tabella 4.4 raccoglie per la coppia di parametri X-Y e ogni lunghezza del passo di tracking il valore di Δ che minimizza l'errore di stima di X e il valore che minimizza l'errore di stima di Y, secondo i grafici riportati nella Sezione A.6.

Step [mm]	Δ for X	Δ for Y
2	10^{-2}	10^{-2}
5	10^{-2}	10^{-2}
10	10^{-2}	10^{-2}
20	10^{-2}	10^{-1}
50	10^{-1}	10^{-1}
80	10^{-1}	10^{-1}

Tabella 4.4: Dimensionamento di Δ per un tracking reale sul piano X-Y.

Nuovamente, dato che la Tabella 4.4 è quasi coincidente alla Tabella 4.3, si può affermare che il criterio di dimensionamento di Δ coincide con quello adottato nella Sezione 4.3.2.

Test 7: simulazione di un tracking *fittizio* lungo una diagonale nello spazio (ottimizzazione globale)

Il test corrente ha lo scopo di sfruttare l'informazione estratta dai grafici della SezioneA.3 per restringere il numero di valori di Δ con cui valutare il comportamento dell'algoritmo di ottimizzazione nel caso di spostamenti della mano leggermente più complessi e compositi, nella fattispecie lungo una traiettoria rettilinea immersa nel frustum del sensore. L'esperimento serve a saggiare la capacità dell'algoritmo di seguire il movimento della mano mentre compie un gesto che richieda anche uno spostamento in profondità, per esempio l'avanzamento all'interno di un desktop 3D.
I parametri che definiscono la posizione sono ottimizzati *contemporaneamente* e nel calcolo della matrice jacobiana J la lunghezza di Δ è comune a tutti e tre i parametri.
La Tabella 4.5 raccoglie per i parametri X, Y, Z e ogni lunghezza del passo di

tracking fittizio il valore di Δ che minimizza l'errore di stima di ogni dimensione, secondo i grafici riportati nella Sezione A.7.

Step [mm]	Δ for X	Δ for Y	Δ for Z
2	10^{-2}	10^{-2}	10^{-2}
5	10^{-2}	10^{-2}	10^{-2}
10	10^{-2}	10^{-2}	10^{-2}
20	10^{-2}	10^{-2}	10^{-1}
50	10^{-1}	10^{-1}	10^{-1}
80	10^{-1}	10^{-1}	10^{-1}

Tabella 4.5: Dimensionamento di Δ per un tracking fittizio nello spazio con ottimizzazione globale.

La Tabella 4.5 conferma ancora una volta la condivisione con i test precedenti del criterio di impostazione del valore di Δ. I grafici della Sezione A.7 mostrano, tuttavia, che anche nel tracking fittizio gli errori di stima sono mediamente superiori rispetto a quelli incontrati nei test di ottimizzazione singola della Sezione 4.3.2.

Test 8: simulazione di un tracking reale lungo una diagonale nello spazio (ottimizzazione globale)

Il test è analogo al suo corrispettivo illustrato nella Sezione 4.3.2, ma per una simulazione di un tracking reale.
La Tabella 4.6 raccoglie per i parametri X, Y, Z e ogni lunghezza del passo di tracking il valore di Δ che minimizza l'errore di stima di ogni dimensione, secondo i grafici riportati nella Sezione A.8.

Valgono le stesse considerazioni effettuate per il test della Sezione 4.3.2, ma dalla Tabella 4.6 si nota che stavolta il cambio di valore di Δ avviene già per un passo di tracking di 20mm. I grafici della Sezione A.8 riportano, inoltre, errori di stima molto elevati anche per passi di tracking ridotti.

Test 9: simulazione di un tracking lungo una diagonale nello spazio (ottimizzazione adattata)

Il test è identico a quello descritto nelle Sezioni 4.3.2 e 4.3.2, tranne che per una modifica nel calcolo della matrice jacobiana definita nell'Equazione 4.10: mentre nei Test 7 e 8 Δ assumeva lo stesso valore di per tutti i parametri, nel test

Step [mm]	Δ for X	Δ for Y	Δ for Z
2	10^{-2}	10^{-2}	10^{-2}
5	10^{-2}	10^{-2}	10^{-2}
10	10^{-2}	10^{-2}	10^{-2}
20	10^{-1}	10^{-1}	10^{-1}
50	10^{-1}	10^{-1}	10^{-1}
80	10^{-1}	10^{-1}	10^{-1}

Tabella 4.6: Dimensionamento di Δ per un tracking reale nello spazio con ottimizzazione globale.

corrente esso differisce per ogni parametro ed è *scalato* dall'ordine di grandezza di quest'ultimo. In altre parole, $\Delta_X = \Delta X$, $\Delta_Y = \Delta Y$ e $\Delta_Z = \Delta Z$, dove Δ_X, Δ_Y e Δ_Z sono gli effettivi passi di derivazione e cambiano contestualmente al variare del valore del parametro a cui sono riferiti. L'euristica adottata è impostare come valore del passo di derivazione per un parametro il valore che nel test della Sezione 4.3.1 è risultato minimizzare l'errore di stima nell'ottimizzazione del singolo parametro. Le tre coordinate della posizione del giunto radice sono nuovamente ottimizzate contemporaneamente.
I grafici riferiti al test e riportati nella Sezione A.9 hanno una struttura diversa rispetto a quella adottata nei grafici degli altri test; infatti, i confronti non sono più tra gli errori di stima in funzione di Δ, ma tra gli errori di stima del tracking reale e fittizio a parità di passo di tracking.
I grafici della Sezione A.9 mostrano che, come ipotizzato, gli errori di stima nella simulazione del tracking reale sono di gran lunga superiori agli errori di stima nella simulazione del tracking fittizio, ma questi ultimi rimangono comunque molto elevati e superiori a quelli rilevati nei test delle Sezioni 4.3.2 e 4.3.2. Una possibile spiegazione è che Δ_X, Δ_Y e Δ_Z assumono valori inferiori a 10^{-2} a causa della scalatura dovuta al campo di variabilità di X, Y e Z che spesso non supera il metro in valore assoluto.

Test 10: simulazione di un tracking fittizio lungo una diagonale nello spazio (ottimizzazione sequenziale)

Dai grafici riportati nelle Sezioni A.7 e A.8 si evince che l'algoritmo restituisce stime pessime quando i tre parametri della posizione sono ottimizzati contemporaneamente, anche per la simulazione di un tracking fittizio. Neanche la modifica

nel calcolo della matrice jacobiana valutata nel test della Sezione 4.3.2sembra apportare miglioramenti. Pertanto, in base alla proprietà dei parametri citata nella Sezione 4.3.1 e all'analisi dei relativi grafici riportati nella Sezione A.1, il test corrente ha come scopo scoprire se l'algoritmo è in grado di restituire una stima accettabile della posizione ottimizzando *singolarmente* i parametri in sequenza. Essendoci tre parametri, le sequenze testate sono $3! = 6$, e per l'ottimizzazione di ogni parametro è stato impostato come valore di Δ quello che è risultato minimizzare l'errore di stima nei test di ottimizzazione di un singolo parametro.

Step [mm]	seq. for X	seq. for Y	seq. for Z
2	yzx/zxy	xyz	zxy/zyx
5	zxy	yzx	zxy/zyx
10	all	zyx	zyx
20	yzx	zyx	zxy/zyx
50	zxy/zyx	zxy/zyx	zxy/zyx
80	zxy	zxy	zxy/zyx

Tabella 4.7: Dimensionamento di Δ per un tracking fittizio nello spazio con ottimizzazione sequenziale.

La Tabella 4.7 indica per ogni lunghezza del passo di tracking fittizio la sequenza di parametri da ottimizzare singolarmente che minimizza l'errore di stima di X, Y e Z rispettivamente. I grafici della Sezione A.10 confrontano la distribuzione dell'errore di stima a seconda della sequenza di ottimizzazione e a parità di passo di tracking. Come si vede nella Tabella, la sequenza migliore varia da parametro a parametro e al variare del passo di tracking, ma in generale gli errori di stima minori corrispondono a sequenze che iniziano l'ottimizzazione con la Z. In quasi tutti i casi, inoltre, è difficile determinare la migliore sequenza, in quanto più sequenze corrispondono a distribuzioni d'errore confrontabili. Dai grafici si nota anche che gli errori di stima sono decisamente inferiori rispetto agli errori risultanti dall'ottimizzazione globale, il che fa supporre che l'ottimizzazione sequenziale sia la giusta strada da intraprendere per l'ottenimento di una stima della posa accettabile.

Test 11: simulazione di un tracking reale lungo una diagonale nello spazio (ottimizzazione sequenziale)

Per il tracking reale simulato in quest'ultimo test valgono le stesse considerazioni sulla riduzione dell'errore di stima effettuate per test nella Sezione 4.3.2, ma cambia sensibilmente la migliore sequenza di ottimizzazioni che lo riduce. Dalla Tabella4.8 si nota, infatti, che il primato è conteso tra le sequenze che iniziano l'ottimizzazione per Z o per X, contrariamente a quanto risulta nella simulazione del tracking fittizio. Dai grafici della Sezione A.11 si nota anche che l'errore di stima è esiguo per un'opportuna scelta della sequenza di ottimizzazione, dando un'ulteriore conferma della validità dell'ottimizzazione sequenziale come strada da intraprendere per una futura stima completa della posa.

Step [mm]	seq. for X	seq. for Y	seq. for Z
2	yzx/zxy	xyz/zyx	zxy/zyx
5	xyz/xzy	xyz/xzy	zxy/zyx
10	all	all	zxy/zyx
20	xyz/xzy	zyx	zxy/zyx
50	xyz/xzy	zxy/zyx	zxy/zyx
80	xyz/xzy	xyz	zyx

Tabella 4.8: Dimensionamento di Δ per un tracking reale nello spazio con ottimizzazione sequenziale.

4.4 Stima della posa iniziale

L'algoritmo descritto nella Sezione 4.2 è un algoritmo di ottimizzazione *locale*, e richiede pertanto una stima iniziale attendibile del punto di ottimo globale a cui mirare, vale a dire una stima della posa della mano al frame corrente. Tale affermazione sembra un controsenso, giacché lo scopo dell'ottimizzazione è la stima della posa stessa. In altre parole, l'algoritmo richiede una stima iniziale della soluzione da raffinare fino a trovare una stima migliore.
Dato che il modello descritto nel Capitolo 3 è costruito a partire da una mano con le dita pressoché tese e aperte a ventaglio, è impensabile che la sua posa sia una valida stima iniziale della posa della mano in un qualsiasi frame, per esempio, quando la mano è chiusa a pugno. L'algoritmo in questo non caso non può convergere.
La soluzione adottata nel progetto è "agganciare il modello alla mano", o più correttamente inizializzare il tracking, calcolando una stima attendibile della posa nel *primo frame*. Quando il frame-rate del sensore è elevato e la velocità di spostamento della mano non eccessiva, è lecito supporre che la posizione e l'orientamento della mano tra due frame consecutivi differiscano al più per qualche millimetro nel world-space, e i grafici in Appendice mostrano che sotto questa ipotesi l'algoritmo è in grado di convergere. Affinché la posa iniziale sia stimabile efficacemente, all'utente viene richiesto di aprire la mano a ventaglio analogamente al modello; i primi frame acquisiti servono a raffinare leggermente le rotazioni dei giunti del modello. Prima del raffinamento delle rotazioni, è richiesta una stima della posizione del giunto radice, ergo della mano intera. I test nelle sezioni precedenti suggeriscono come stima la *back-projection* delle coordinate del *baricentro* dei pixel la cui profondità è inferiore a 5m. I grafici relativi ai test, infatti, mostrano che l'algoritmo può ancora convergere, dato che la distanza del baricentro dal polso, ove è situata la radice dell'albero cinematico, non supera in genere i 10cm per dimensione. L'Equazione 4.12 formalizza tale stima.

$$\begin{aligned}
\overline{u} &= \frac{\sum_{v=1}^{144}\sum_{u=1}^{176} z(u,v)u}{\sum_{v=1}^{144}\sum_{u=1}^{176} z(u,v)} \text{con u,v t.c. } z(u,v) < 5m \\
\overline{v} &= \frac{\sum_{v=1}^{144}\sum_{u=1}^{176} z(u,v)v}{\sum_{v=1}^{144}\sum_{u=1}^{176} z(u,v)} \text{con u,v t.c. } z(u,v) < 5m \\
\overline{z} &= \frac{\sum_{v=1}^{144}\sum_{u=1}^{176} z(u,v)}{144 \cdot 176} \text{con u,v t.c. } z(u,v) < 5m
\end{aligned} \tag{4.12}$$

$$\begin{aligned}
X_w &= \frac{\overline{u} - Z_w c_x}{f_x} \\
Y_w &= \frac{\overline{v} - Z_w c_y}{f_y} \\
Z_w &= \overline{z}
\end{aligned}$$

Conclusioni

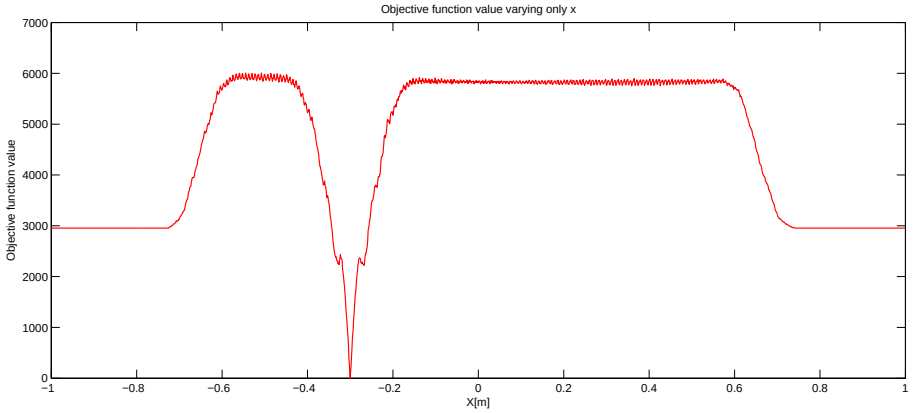
In questa tesi è stata descritta una possibile realizzazione del primo passo nella progettazione di un'interfaccia uomo-macchina tridimensionale che impieghi unicamente la mano nuda di un generico utente come controller. Partendo dalla caratterizzazione di un *gesto* della mano, inizialmente si è ripercorsa a ritroso la filiera specificando il tipo di dati richiesto all'ingresso di ogni blocco intermedio. In seguito la filiera è stata risalita dal passo iniziale, consistente nella progettazione del sistema d'acquisizione dei movimenti della mano illustrato nel Capitolo 1, passando poi alla descrizione del modello della mano e della lunga procedura per costruirlo arrivando, infine, al dimensionamento dei parametri dell'algoritmo d'ottimizzazione. L'ultimo passo è necessario per ottenere una stima accurata della posa della mano in ogni frame, e le stime della posa ottenute sono i dati d'ingresso per l'algoritmo di stima della traiettoria della mano, oggetto della tesi ma solo citata per mancanza di tempo. La sola ricostruzione della mesh ha richiesto diversi mesi di lavoro e innumerevoli esperimenti, anche con il coinvolgimento di volontari, ma alla fine la procedura di modellazione definita sembra portare a buoni risultati, come mostrato in Figura 3.13. Anche lo sviluppo del codice di supporto all'algoritmo di ottimizzazione ha richiesto molto tempo e continui cambiamenti nella struttura, in particolar modo l'implementazione efficiente dell'algoritmo Linear Blend Skinning e l'integrazione della libreria VTK per la visualizzazione dei dati e la definizione del sensore virtuale. Allo stesso modo, l'acquisizione di una sufficiente padronanza di Maya e lo sviluppo degli script di esportazione dei vertici e delle facce hanno richiesto un tempo considerevole. Tuttavia, la parte più complessa, importante, lunga e incompleta del progetto è stata lo studio dell'algoritmo di ottimizzazione e la calibrazione dei parametri che lo governano. Il problema da risolvere, infatti, ha una taglia elevata dovuta al consistente numero di gradi di libertà dello scheletro della mano, e solo continui e numerosi test hanno provato che i parametri che definiscono la posa non possono essere ottimizzati congiuntamente. Il solo problema semplificato ha rivelato di essere risolvibile limitatamente a un opportuno dimensionamento di Δ

in una situazione controllata, e al momento non si può generalizzare il risultato a un qualsiasi movimento o al dimensionamento di Δ per altri parametri senza svolgere ulteriori test. Da questi ultimi è emerso che il modo migliore per ridurre l'errore di stima della posizione del giunto radice della mano in movimento nello spazio è ottimizzare ogni dimensione in maniera isolata e secondo un particolare ordinamento. I grafici nelle Sezioni A.10 e A.11 riportano errori di stima esigui (al massimo pochi millimetri) anche nella simulazione di un tracking reale a patto di aver dimensionato correttamente Δ_X, Δ_Y e Δ_Z. Tali risultati favorevoli suggeriscono che, forse, la stima ottenuta tramite ottimizzazioni sequenziali è la strada giusta da intraprendere per il prosieguo del progetto, e che l'approccio possa estendersi anche all'ottimizzazione delle rotazioni. Infine, i grafici in Appendice nel complesso indicano che, almeno per spostamenti limitati della mano, la distribuzione dell'errore di stima risultante dalla simulazione di un tracking reale è simile a quella risultante dalla simulazione di un tracking fittizio. Per spostamenti consistenti, invece, l'algoritmo diverge durante la simulazione di un tracking reale, ma la divergenza è evitabile dimensionando opportunamente Δ. La similitudine delle distribuzioni degli errori in entrambi i tipi di simulazione è indice che l'algoritmo di stima correttamente calibrato ha le potenzialità per essere impiegato in un tracking reale. Una lecita critica sollevabile dopo un'attenta analisi dei grafici in Appendice è che le distribuzioni dell'errore di stima presentano spesso picchi elevati anche per un opportuno dimensionamento di Δ, e che in corrispondenza di questi ultimi la stima è poco accurata. In realtà, i grafici mostrano anche tali picchi non sono un vero problema, in quanto in presenza di un framerate elevato la mano al frame successivo è spostata quanto basta per allontanarsi dal picco ma al contempo non eccessivamente da far perdere l'allineamento. In conclusione, il problema in esame è molto complesso ma ha buone possibilità di essere risolto in futuro, e questa tesi apre la strada per il raggiungimento di tale obiettivo.

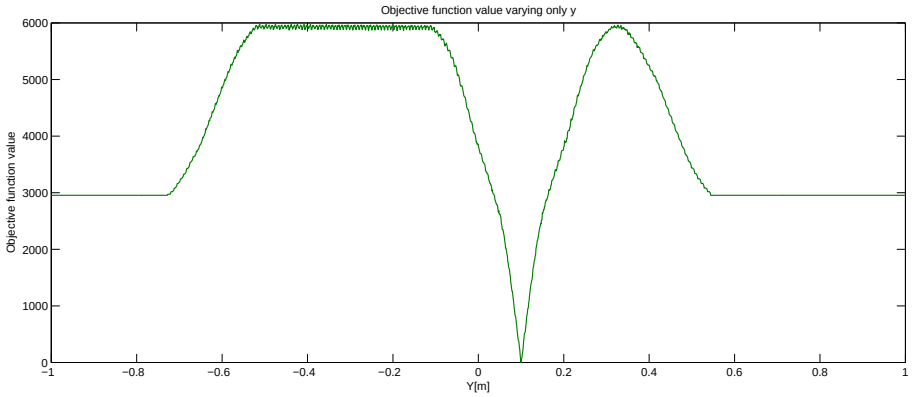
Appendice A

Grafici dei test

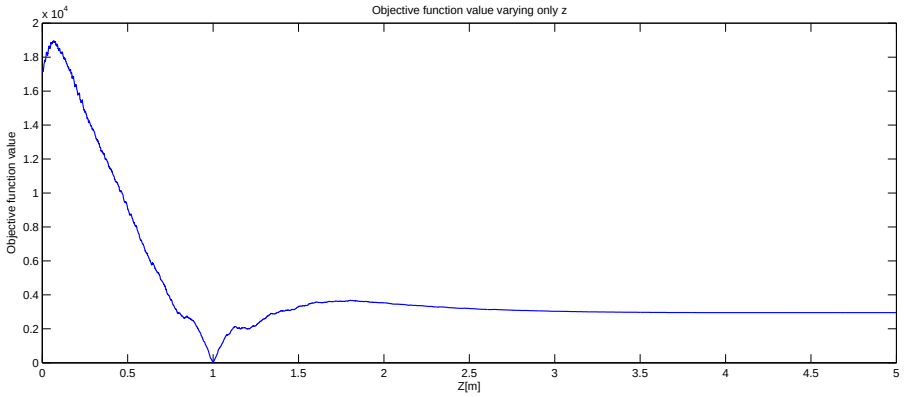
A.1 Grafici per il Test 1



a) Andamento della funzione obiettivo al variare di X

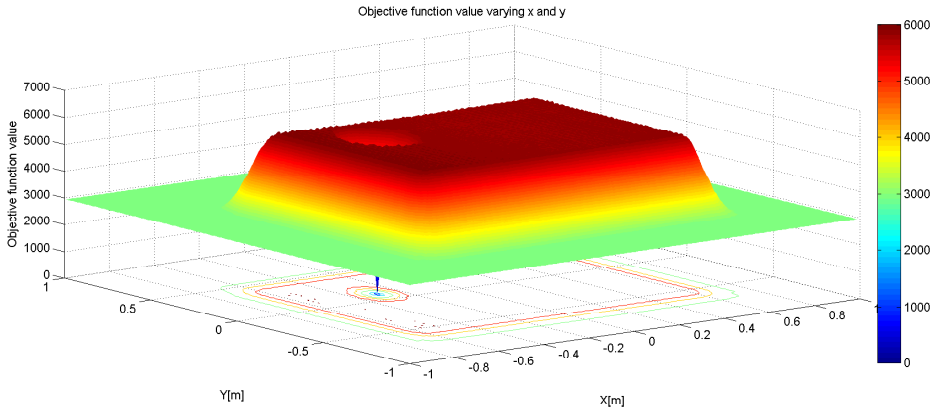


b) Andamento della funzione obiettivo al variare di Y

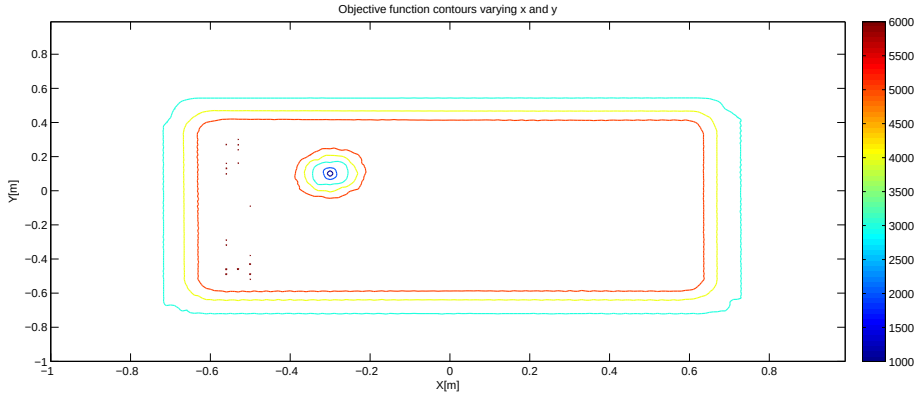


c) Andamento della funzione obiettivo al variare di Z

Figura A.1: Andamento della funzione obiettivo al variare di una sola coordinata della posizione del giunto radice.

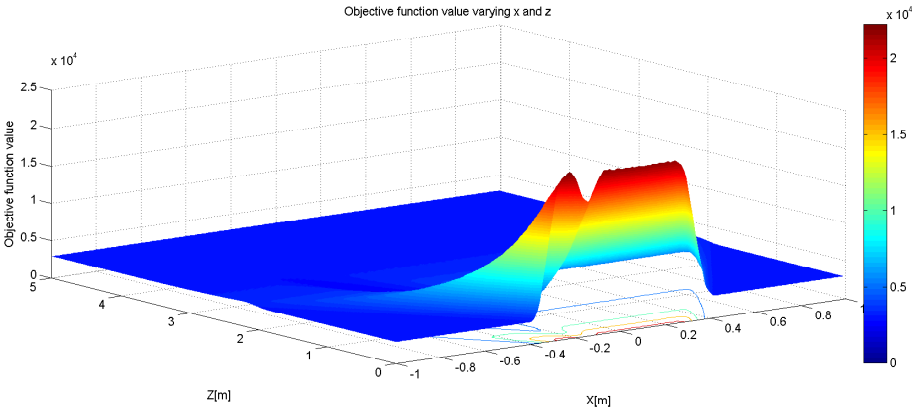


a) Andamento della funzione obiettivo al variare di X e Y

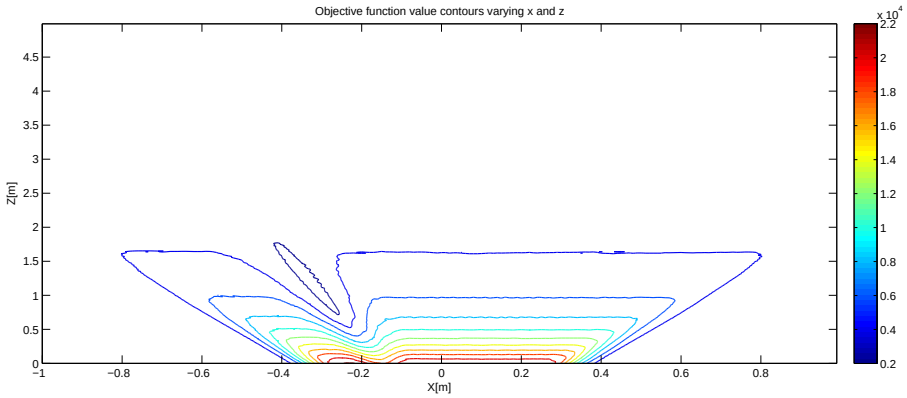


b) Relative curve di livello

Figura A.2: Traslazione nel piano X-Y.

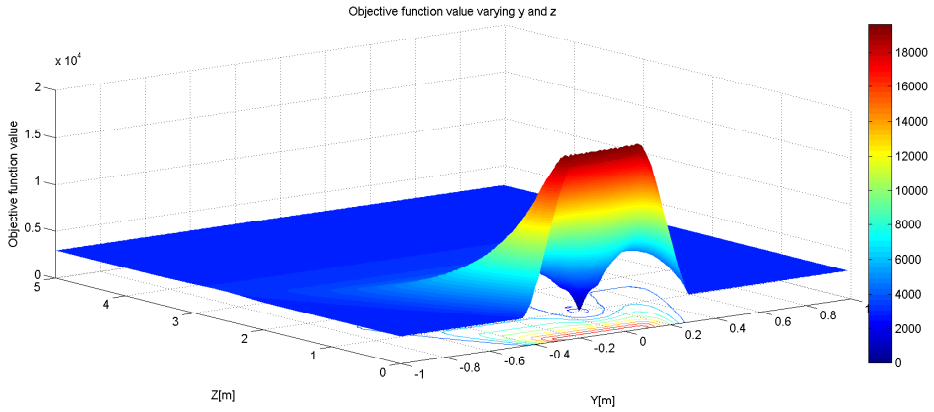


a) Andamento della funzione obiettivo al variare di X e Z

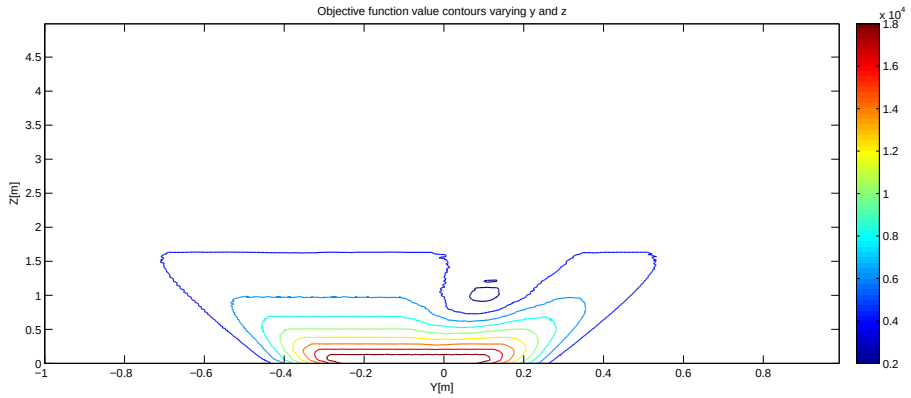


b) Relative curve di livello

Figura A.3: Traslazione nel piano X-Z.



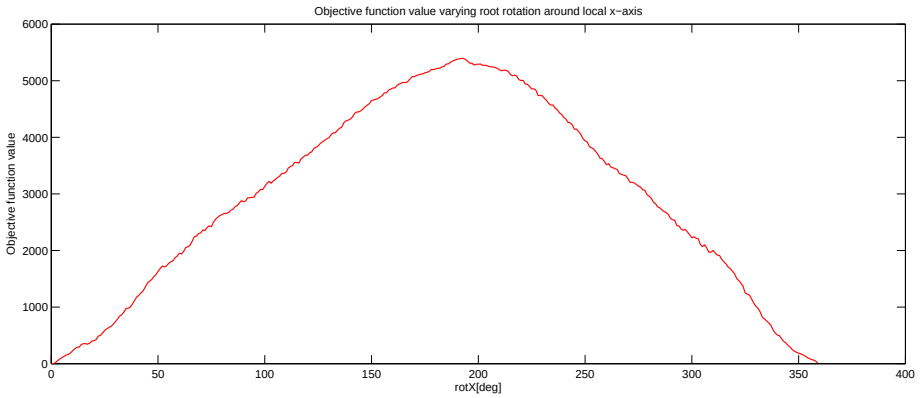
a) Andamento della funzione obiettivo al variare di Y e Z



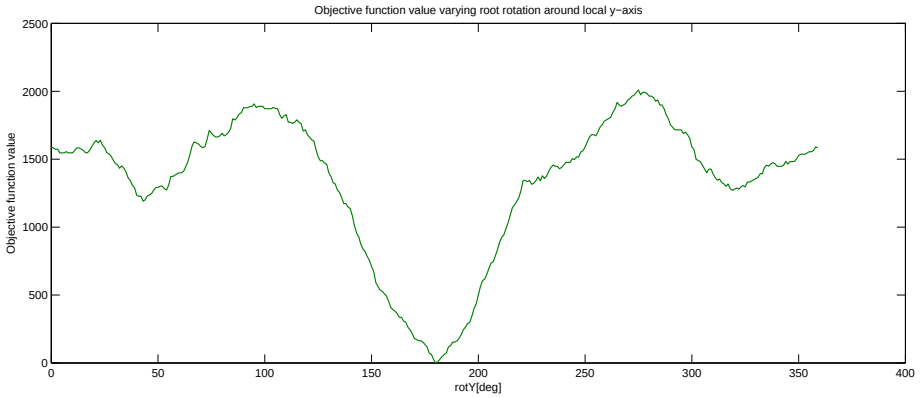
b) Relative curve di livello

Figura A.4: Traslazione nel piano Y-Z.

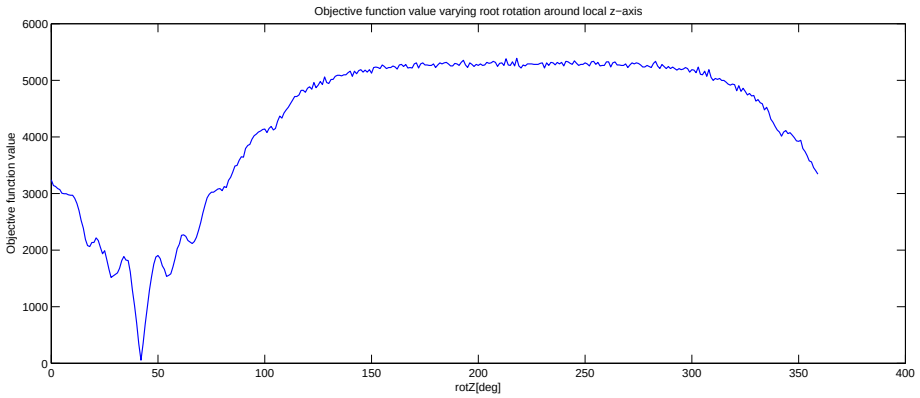
A.2 Grafici per il Test 2



a) Andamento della funzione obiettivo al variare di rot_x

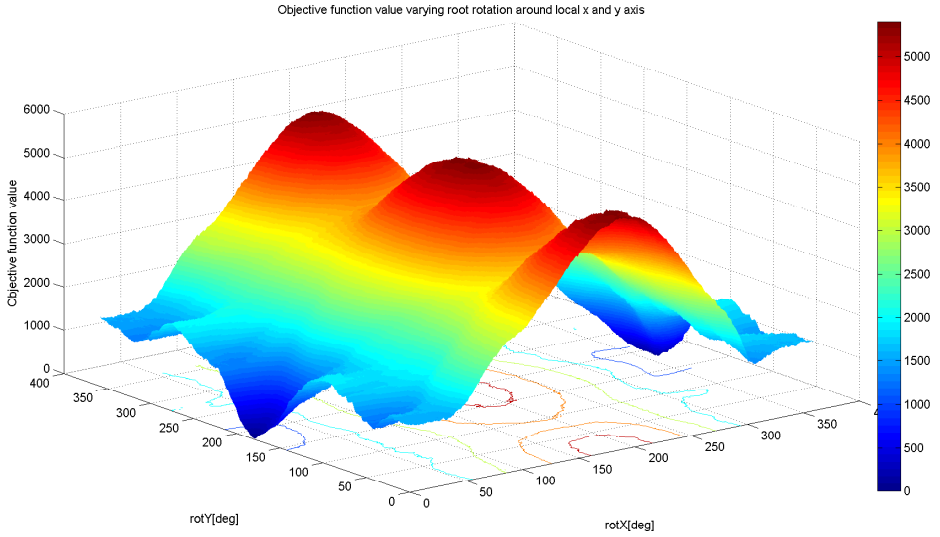


b) Andamento della funzione obiettivo al variare di rot_y

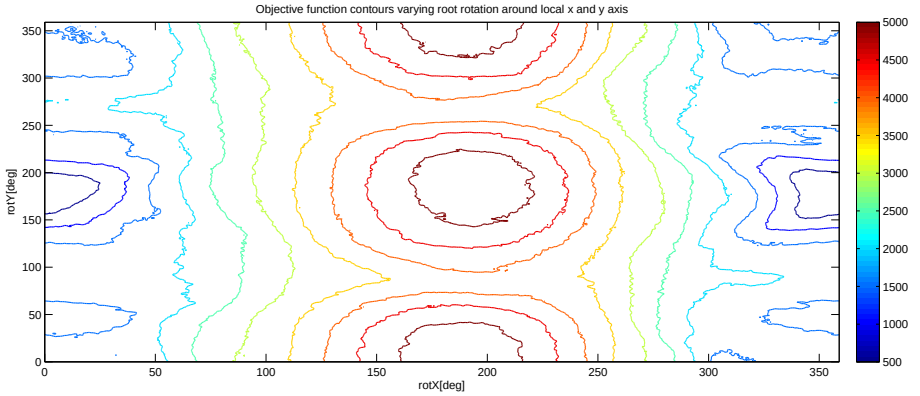


c) Andamento della funzione obiettivo al variare di rot_z

Figura A.5: Andamento della funzione obiettivo al variare della rotazione del giunto radice attorno a un solo asse coordinato.



a)



b)

Figura A.6: Andamento della funzione obiettivo al variare di rot_x e rot_y (a)) e relative curve di livello (b)).

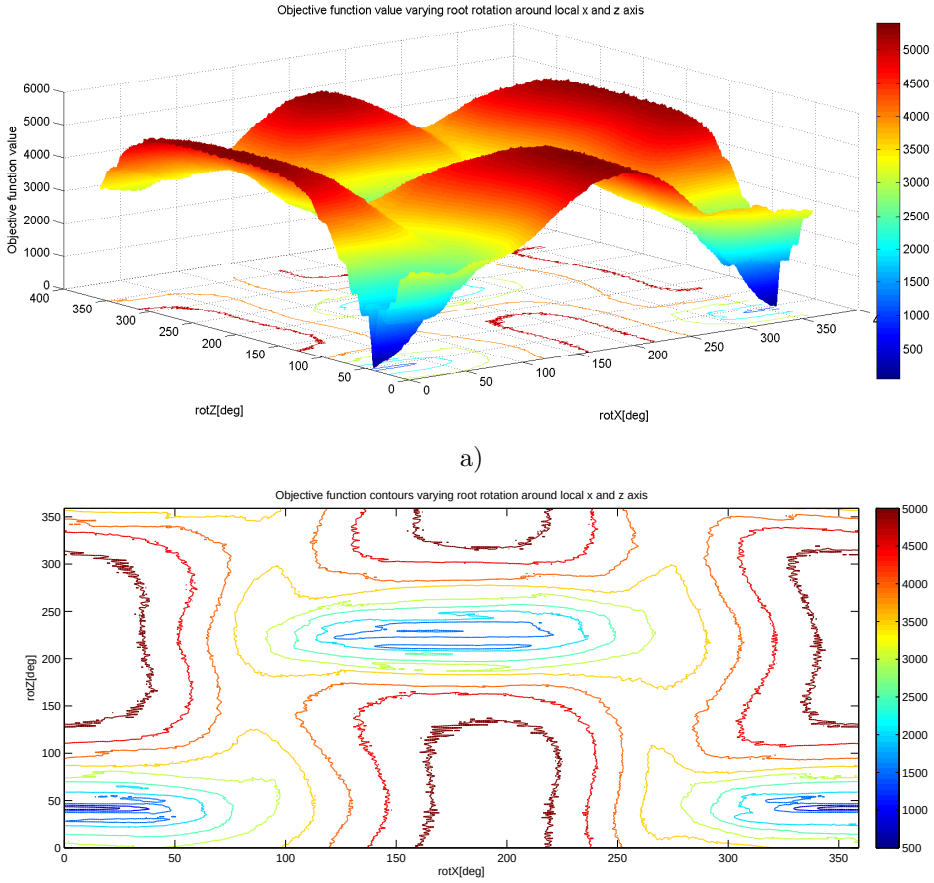


Figura A.7: Andamento della funzione obiettivo al variare di rot_x e rot_z (a)) e relative curve di livello (b)).

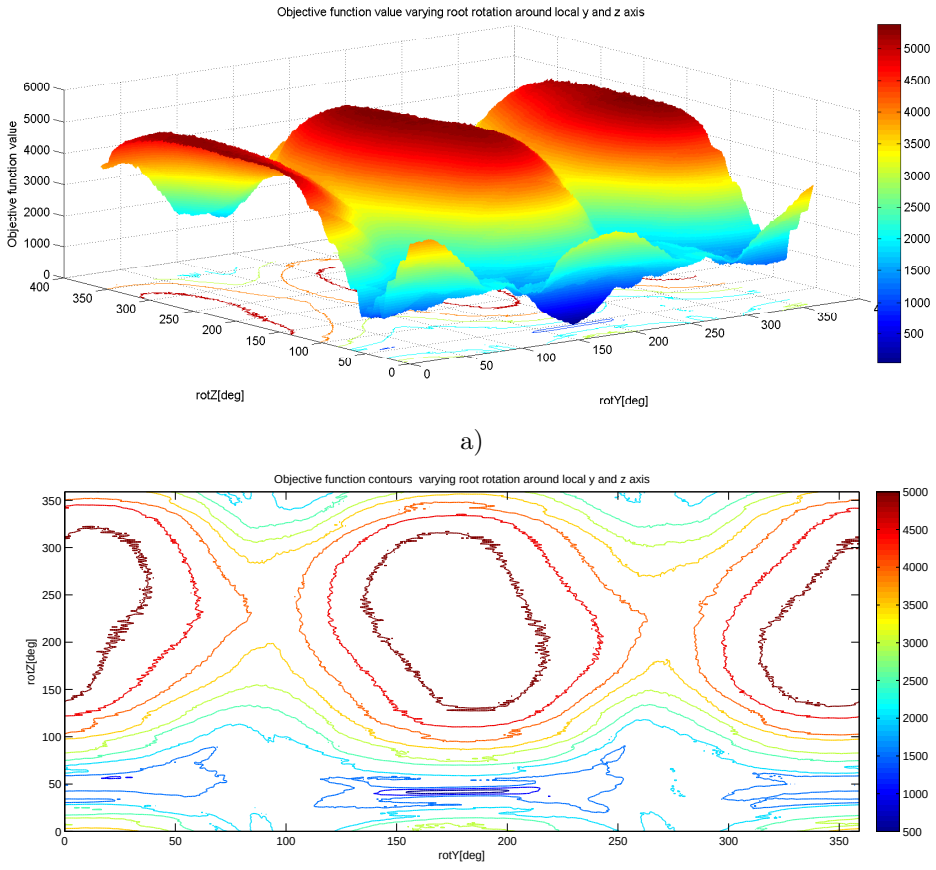
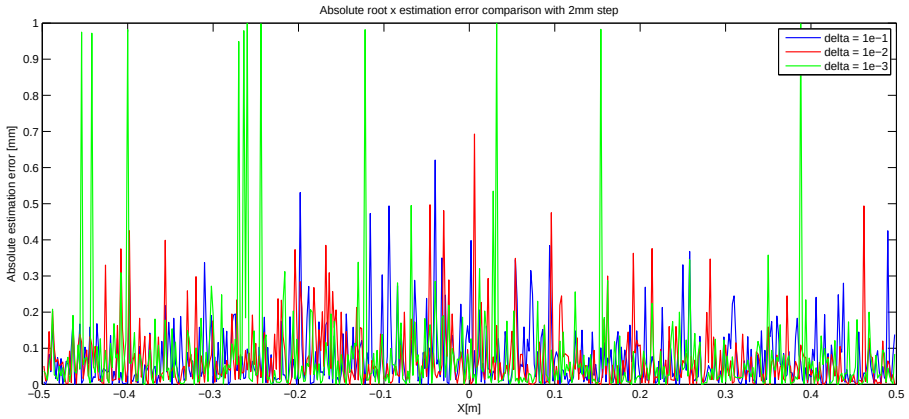
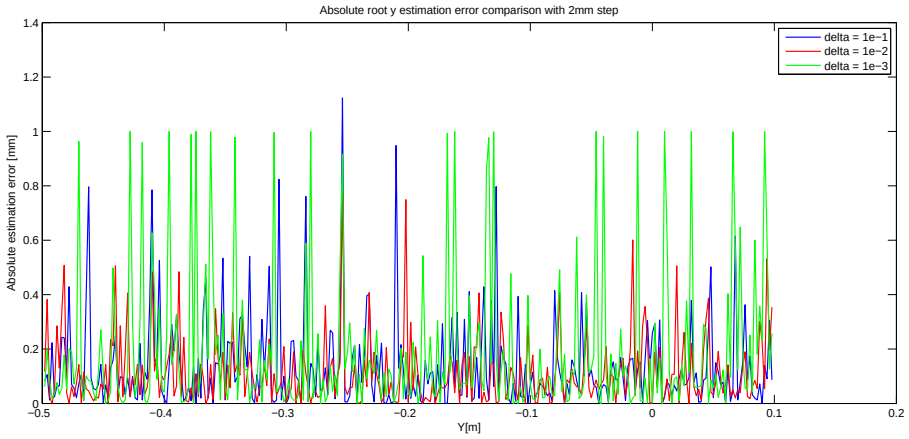


Figura A.8: Andamento della funzione obiettivo al variare di rot_y e rot_z (a)) e relative curve di livello (b)).

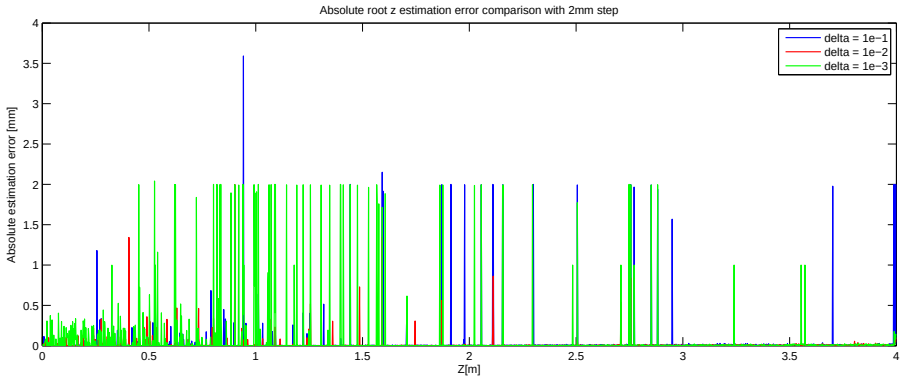
A.3 Grafici per il Test 3



a) Stima di X

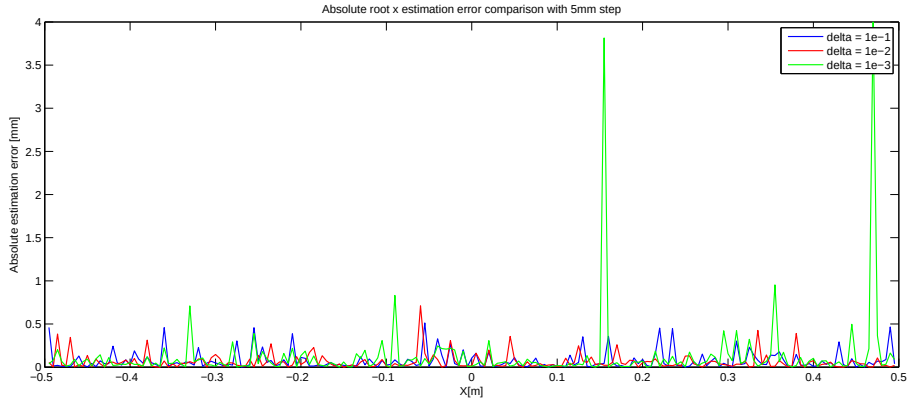


b) Stima di Y

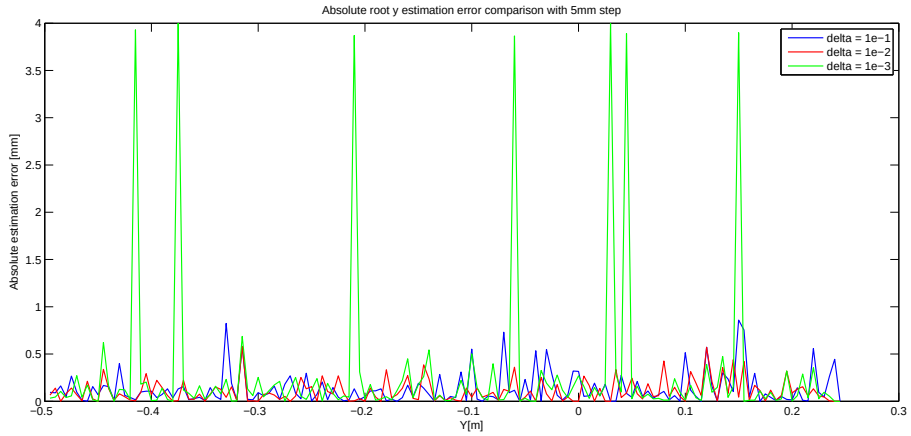


c) Stima di Z

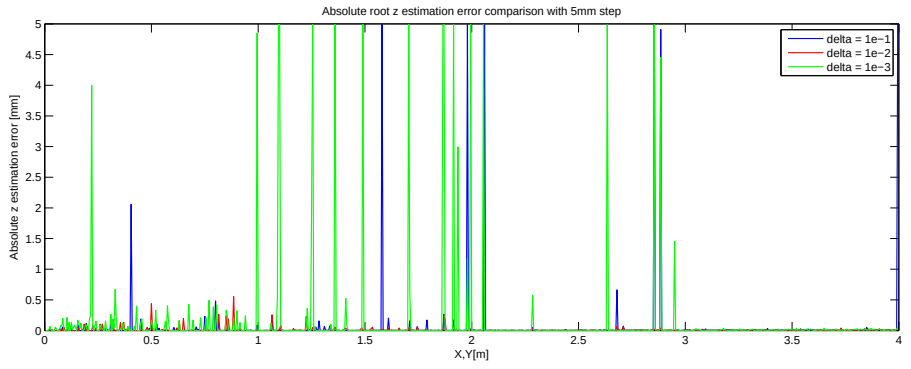
Figura A.9: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 2mm.



a) Stima di X

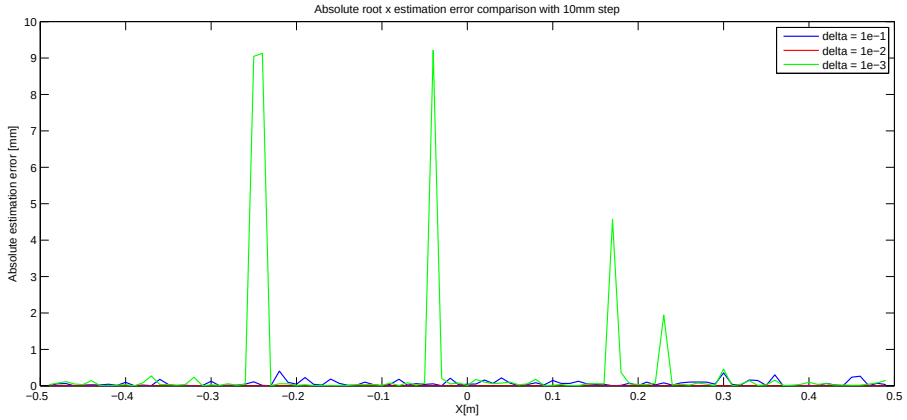


b) Stima di Y

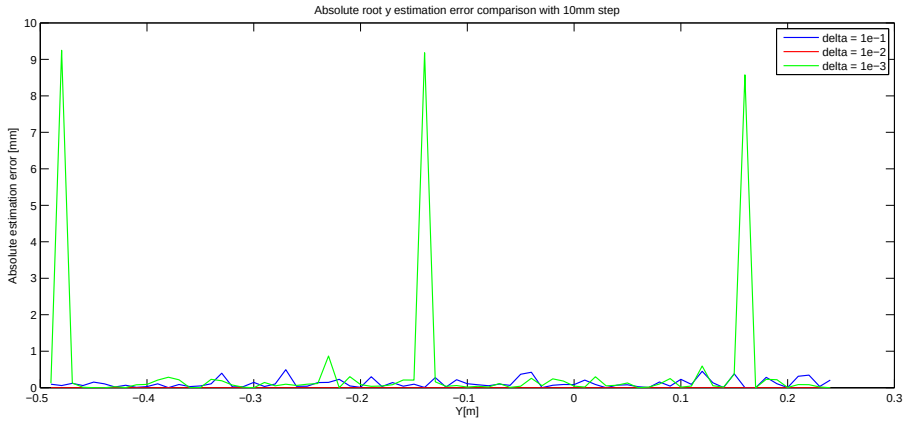


c) Stima di Z

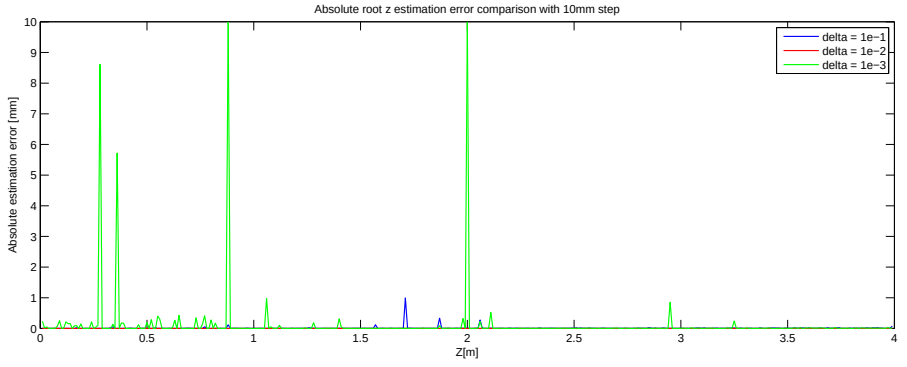
Figura A.10: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 2mm.



a) Stima di X

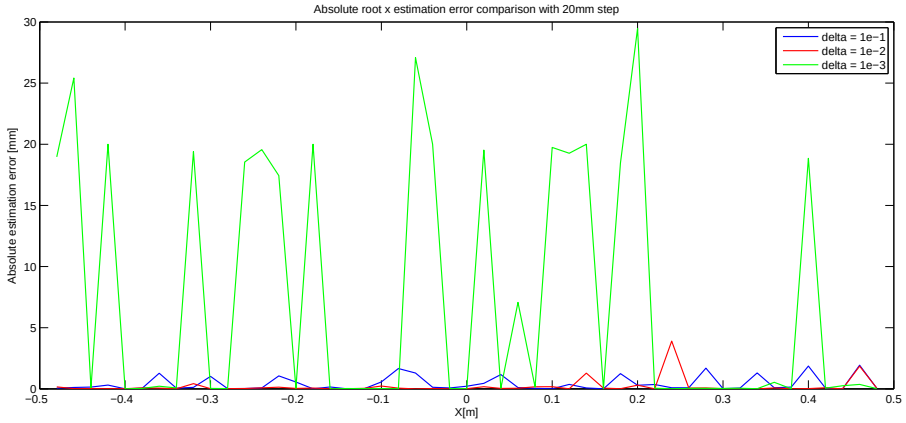


b) Stima di Y

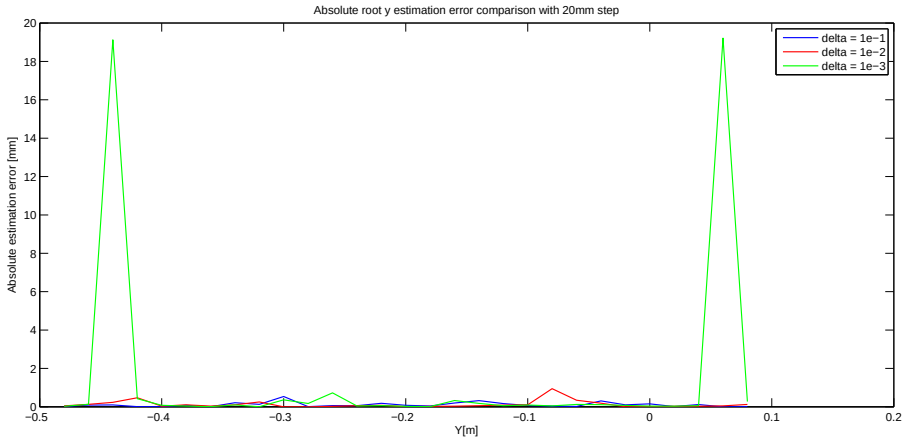


c) Stima di Z

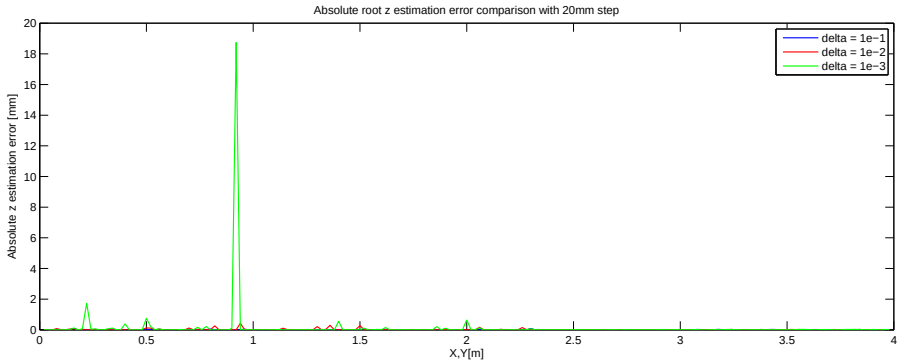
Figura A.11: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 10mm.



a) Stima di X

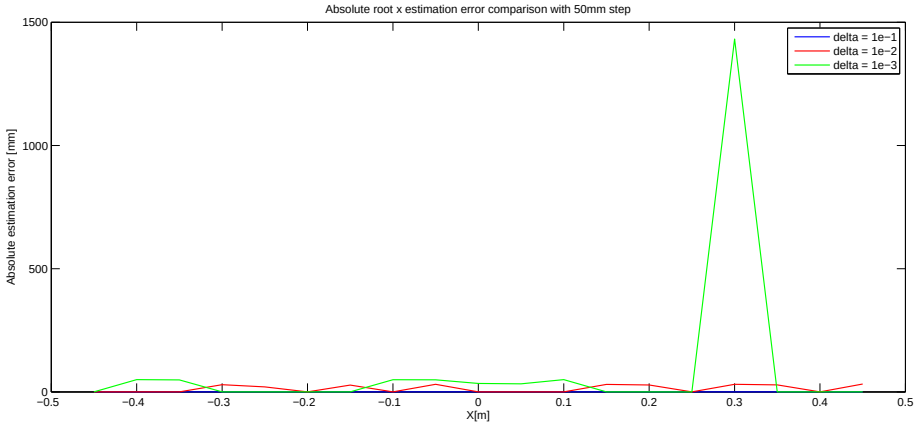


b) Stima di Y

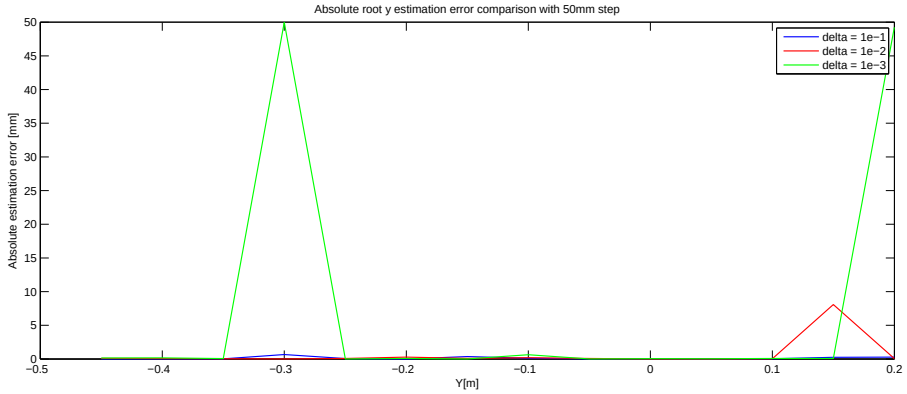


c) Stima di Z

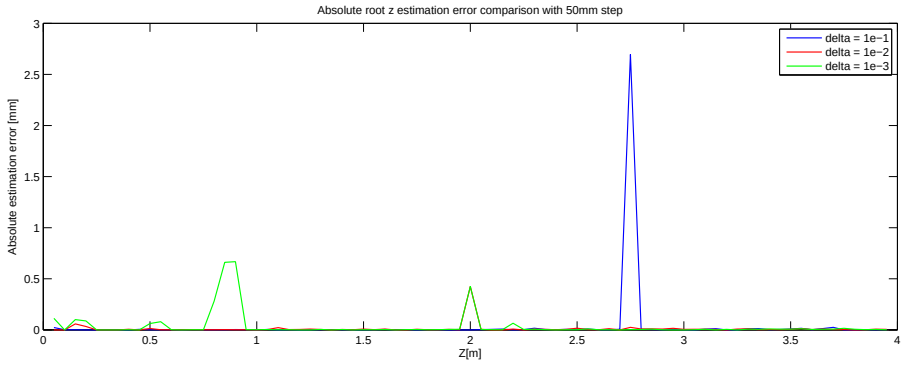
Figura A.12: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 20mm.



a) Stima di X

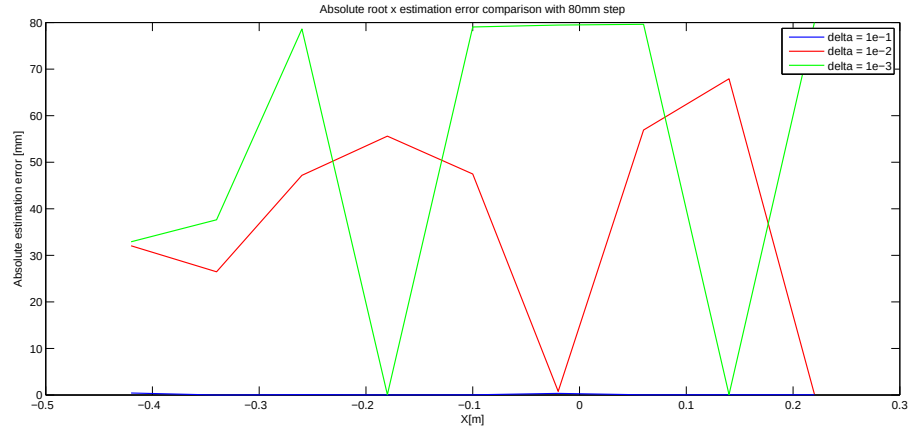


b) Stima di Y

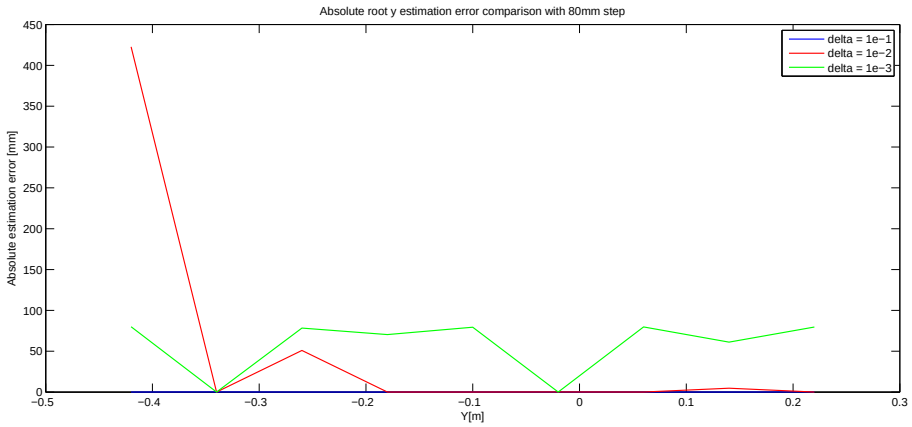


c) Stima di Z

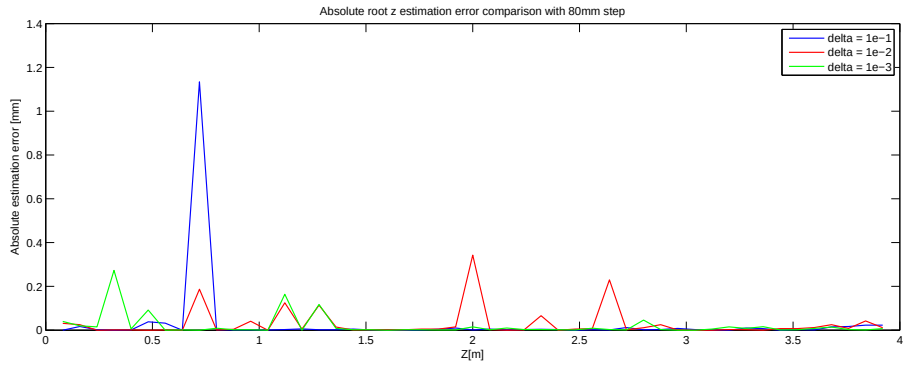
Figura A.13: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 50mm.



a) Stima di X

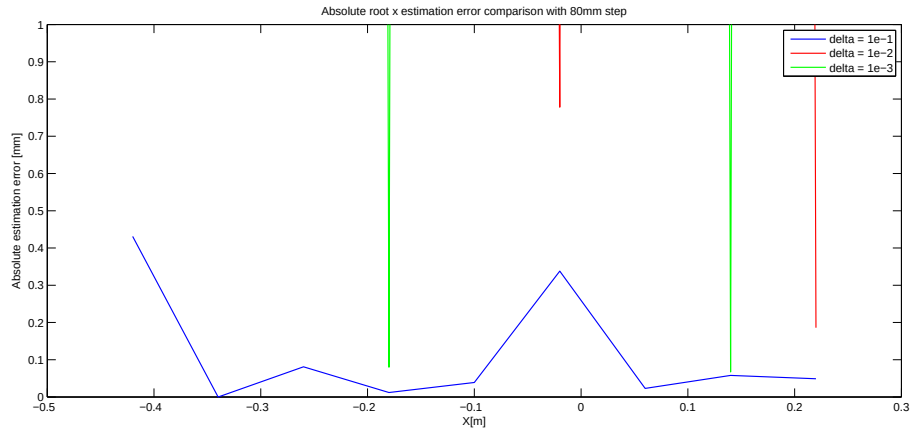


b) Stima di Y

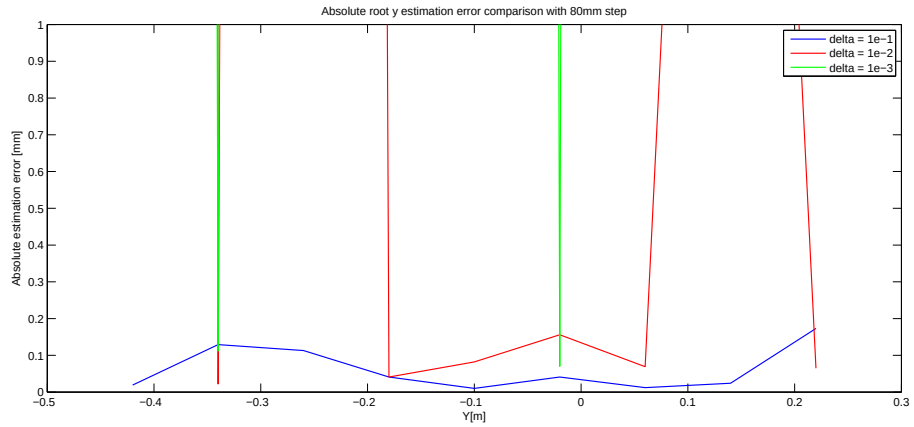


c) Stima di Z

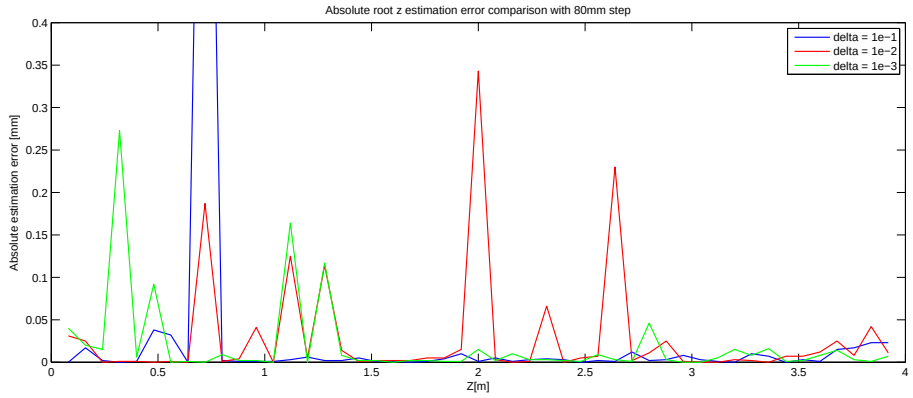
Figura A.14: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 80mm.



a) Stima di X



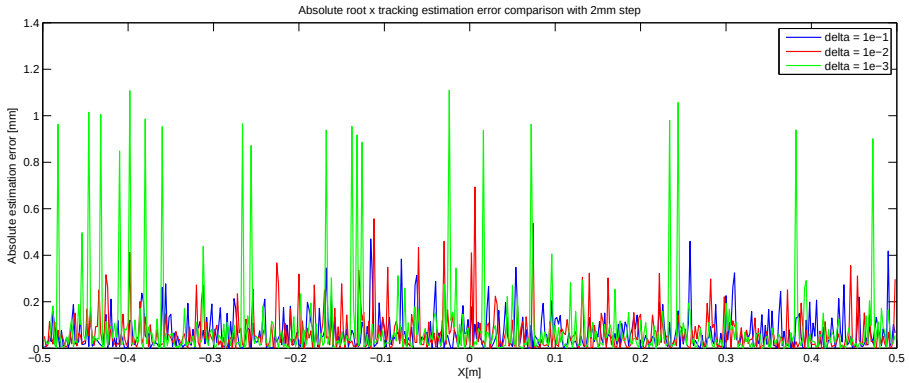
b) Stima di Y



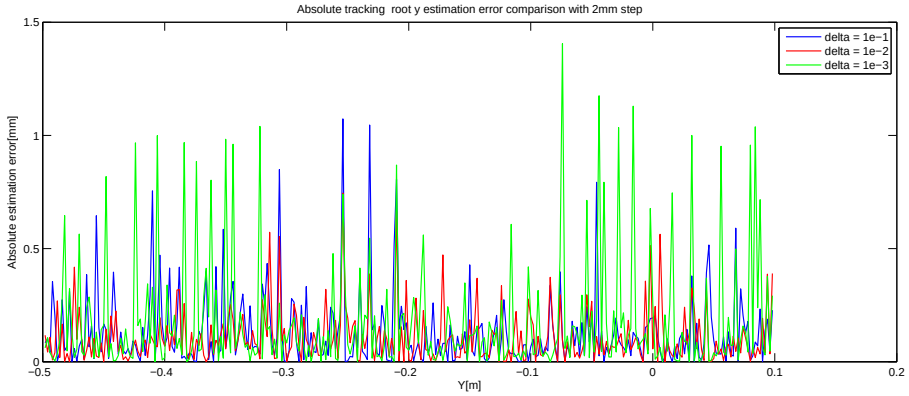
c) Stima di Z

Figura A.15: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 80mm (ingrandimento).

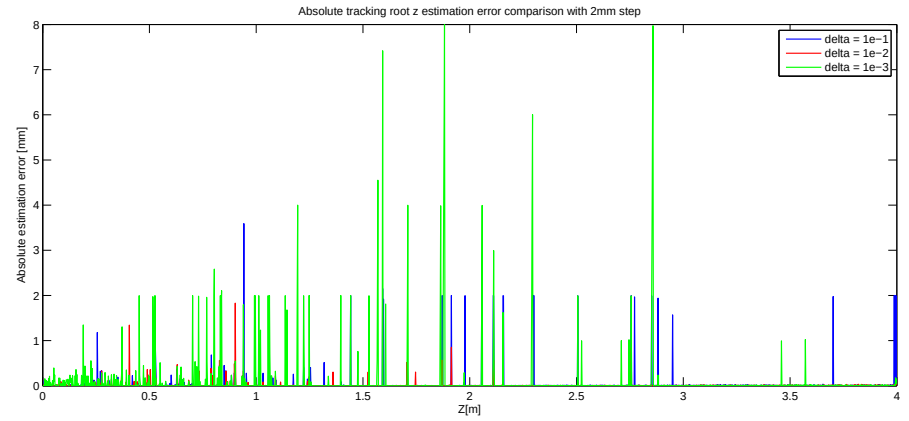
A.4 Grafici per il Test 4



a) Stima di X

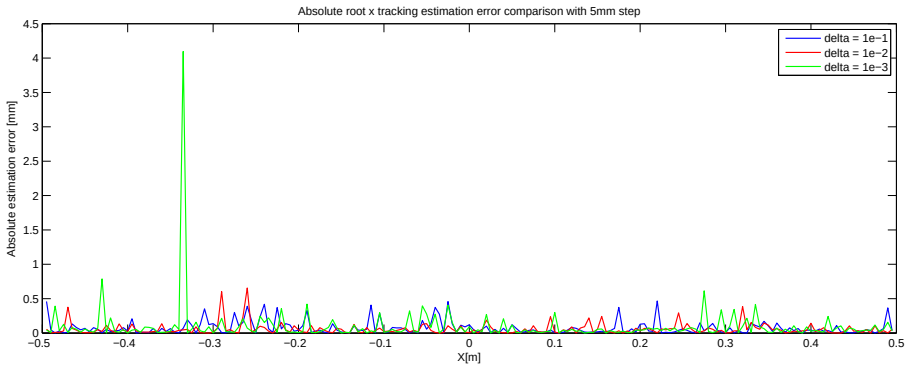


b) Stima di Y

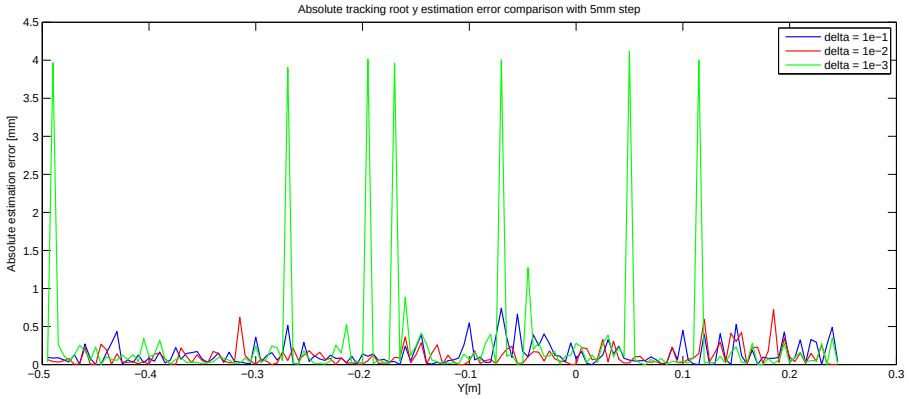


c) Stima di Z

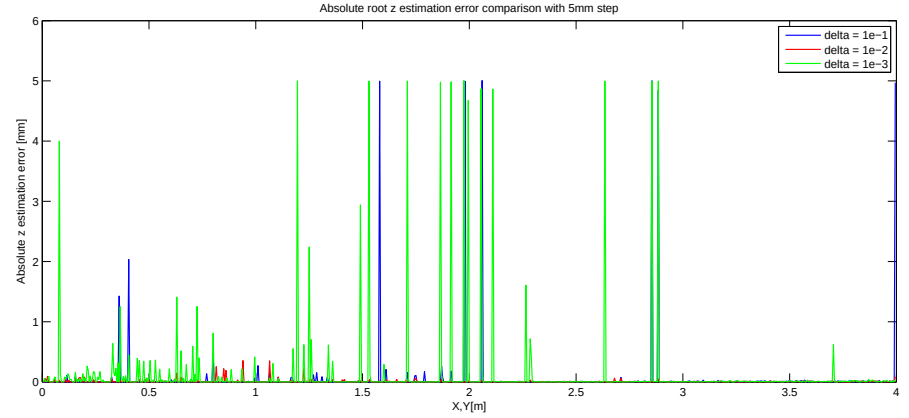
Figura A.16: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 2mm in modalità tracking.



a) Stima di X

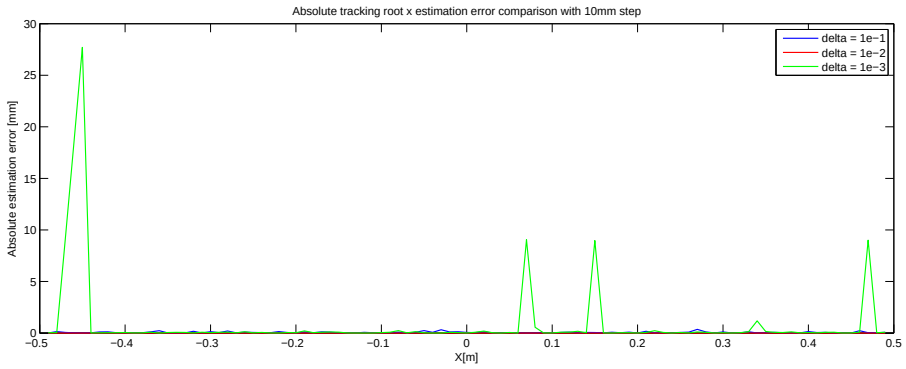


b) Stima di Y

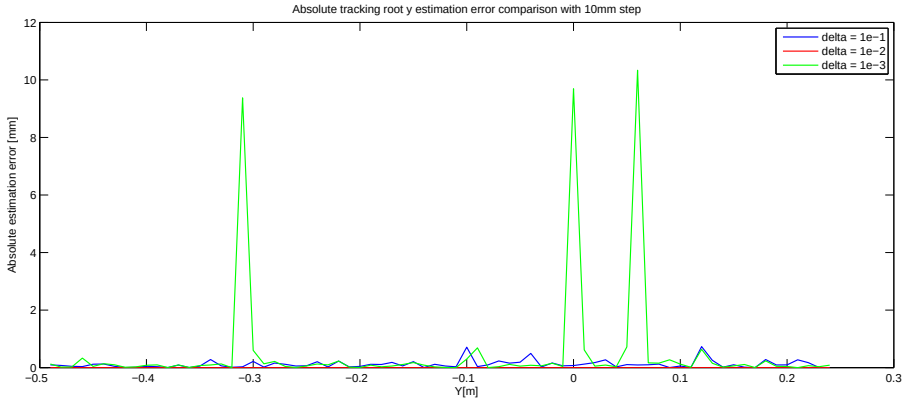


c) Stima di Z

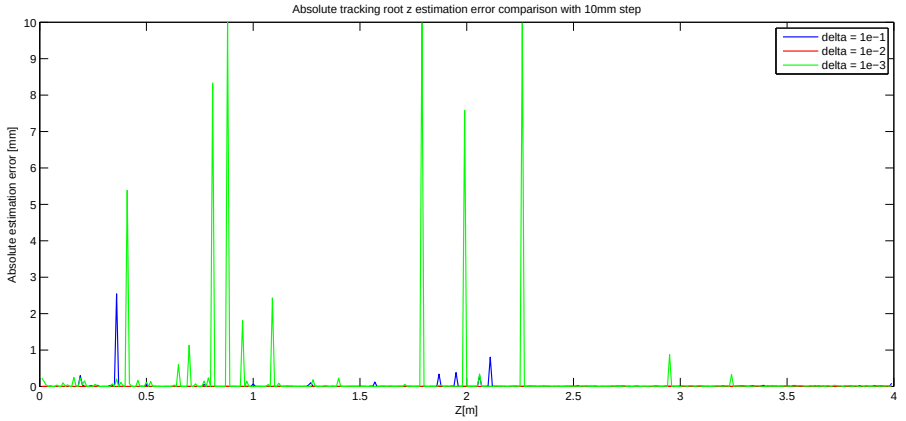
Figura A.17: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 5mm in modalità tracking.



a) Stima di X

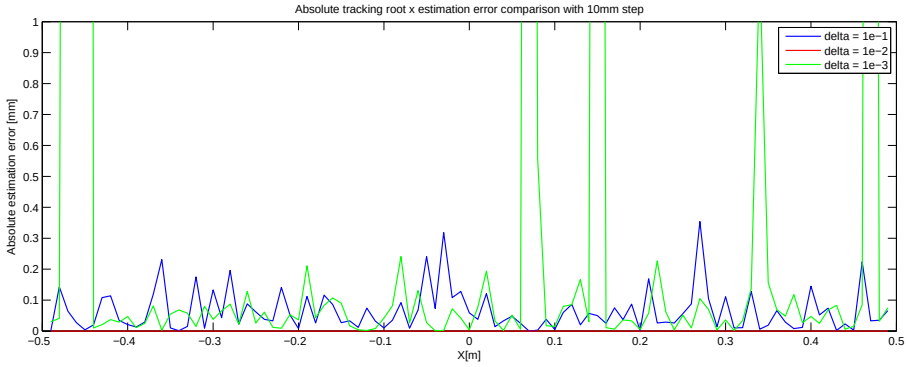


b) Stima di Y

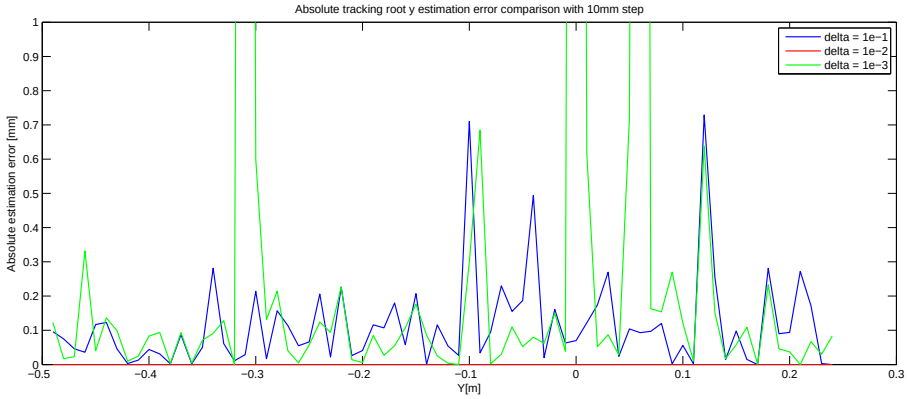


c) Stima di Z

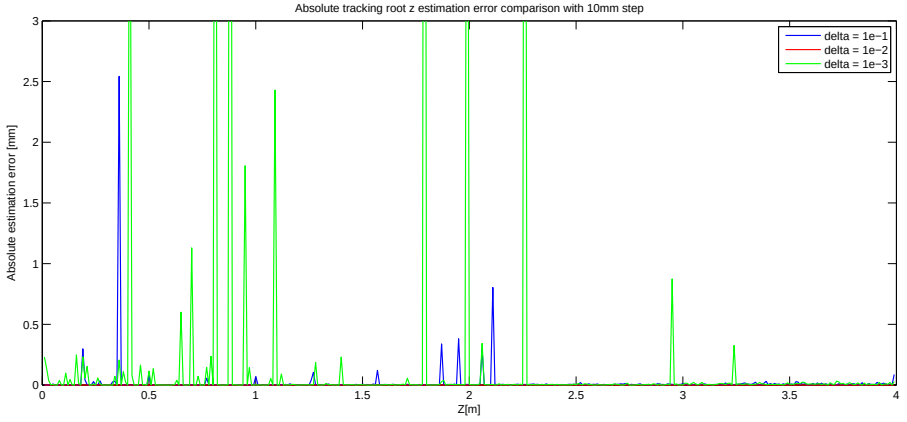
Figura A.18: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 10mm in modalità tracking.



a) Stima di X

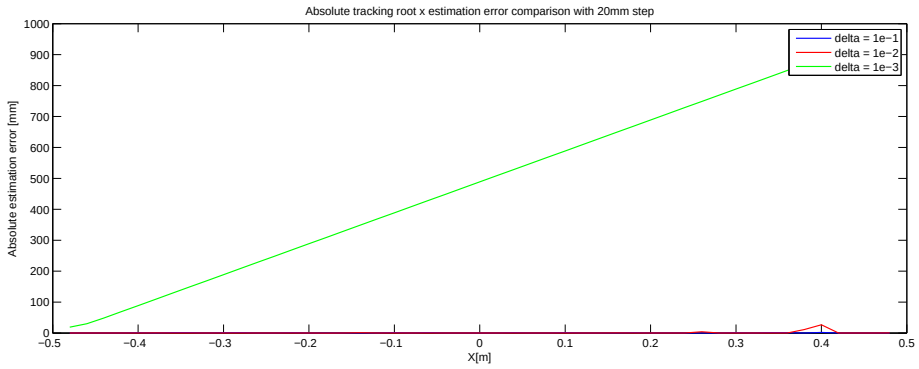


b) Stima di Y

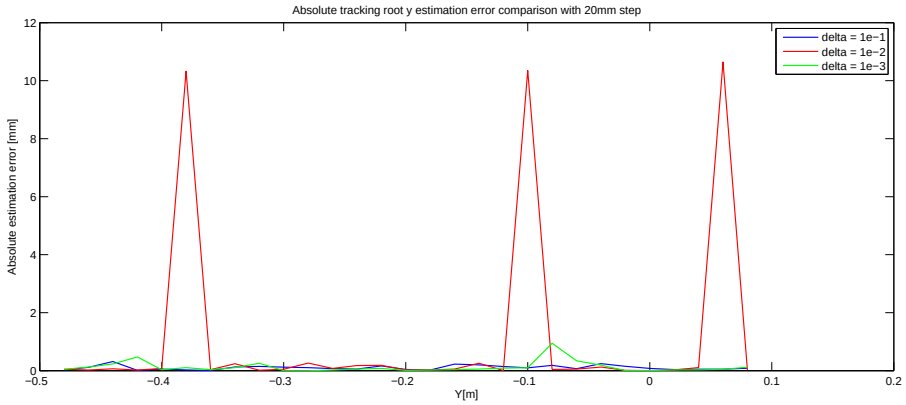


c) Stima di Z

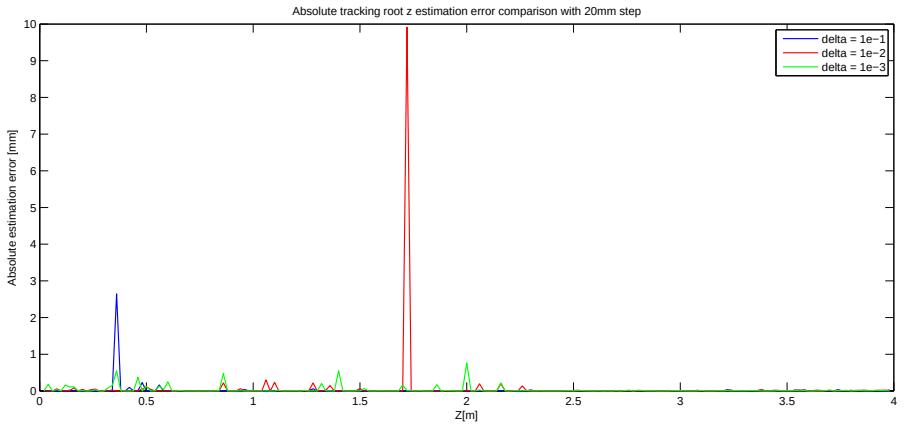
Figura A.19: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 10mm in modalità tracking (ingrandimento).



a) Stima di X

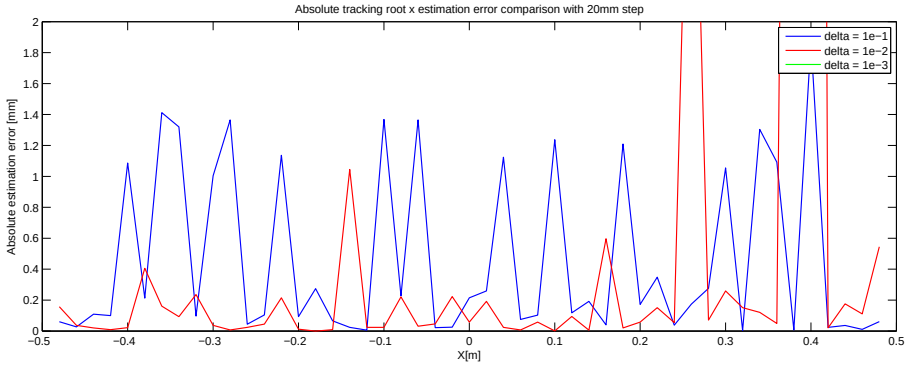


b) Stima di Y

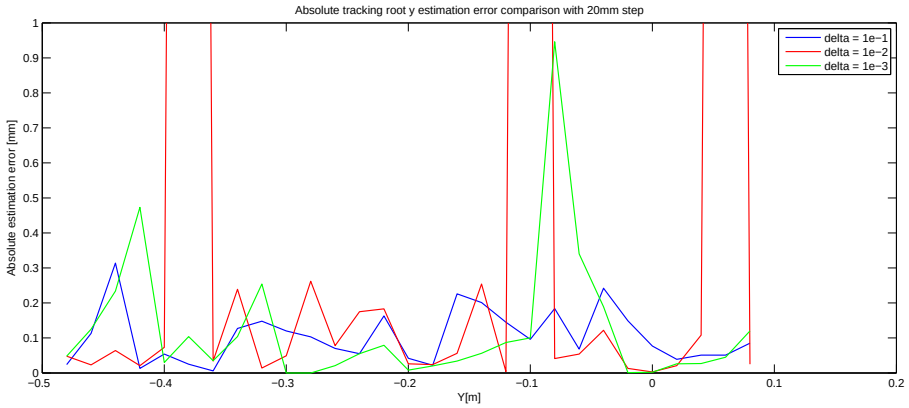


c) Stima di Z

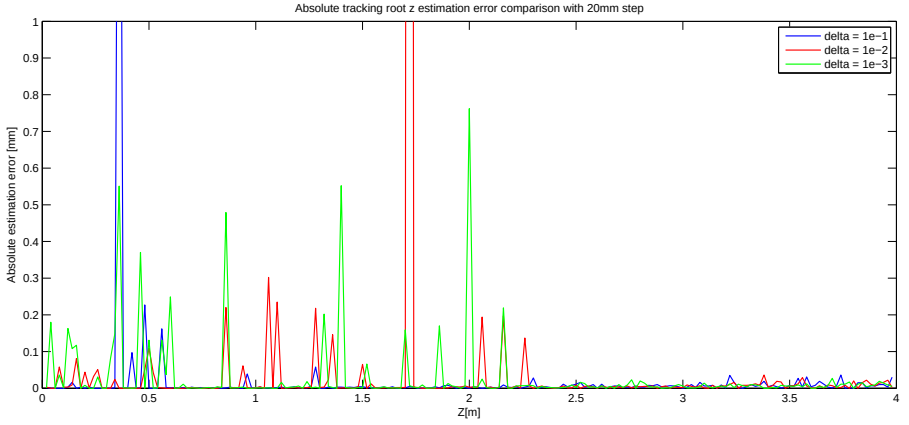
Figura A.20: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 20mm in modalità tracking.



a) Stima di X

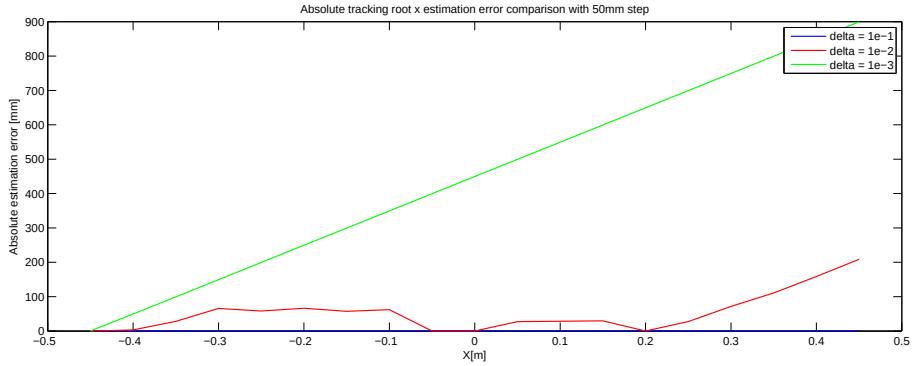


b) Stima di Y

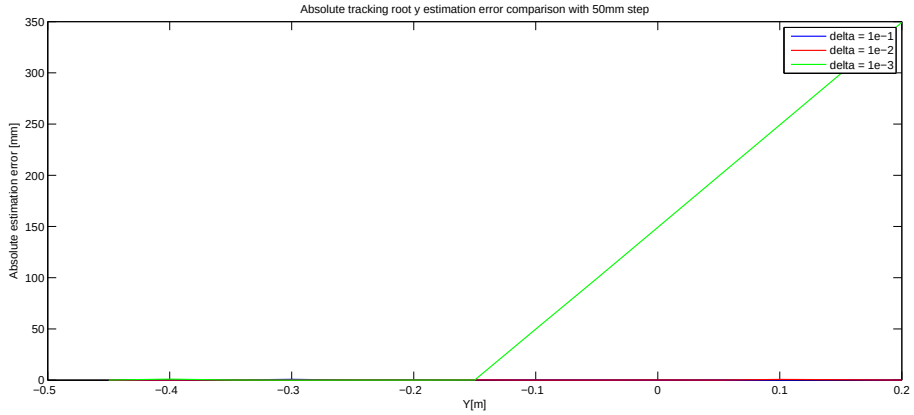


c) Stima di Z

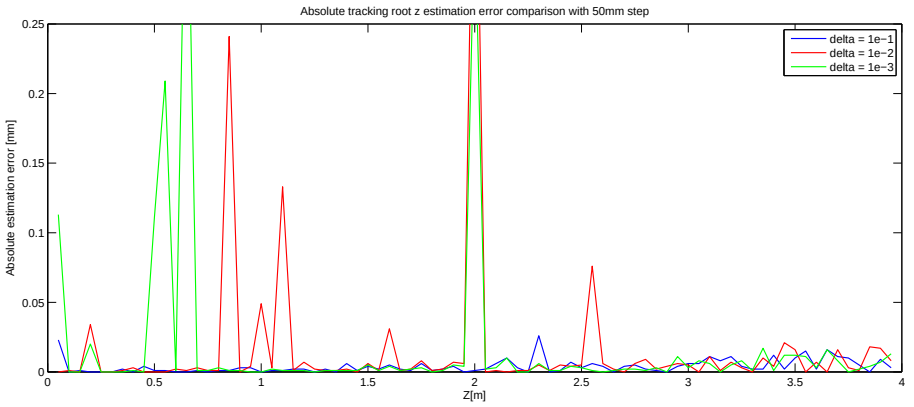
Figura A.21: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 20mm in modalità tracking (ingrandimento).



a) Stima di X

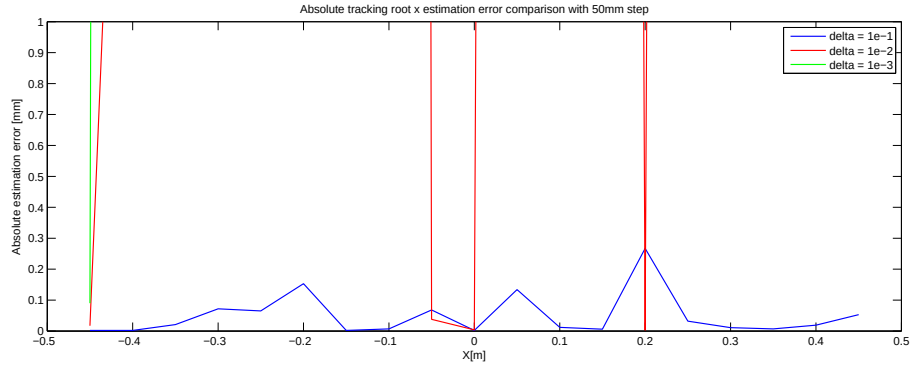


b) Stima di Y

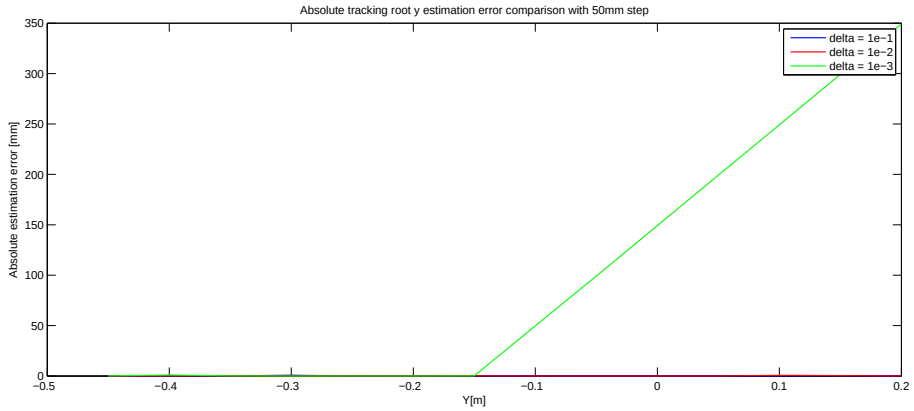


c) Stima di Z

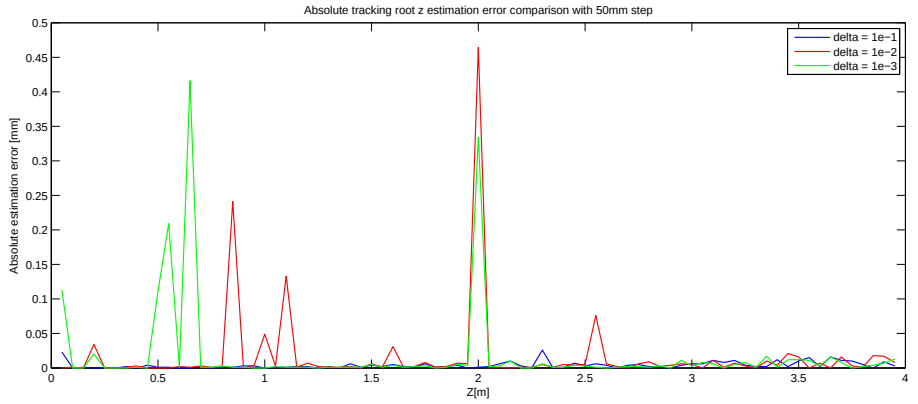
Figura A.22: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 50mm in modalità tracking.



a) Stima di X

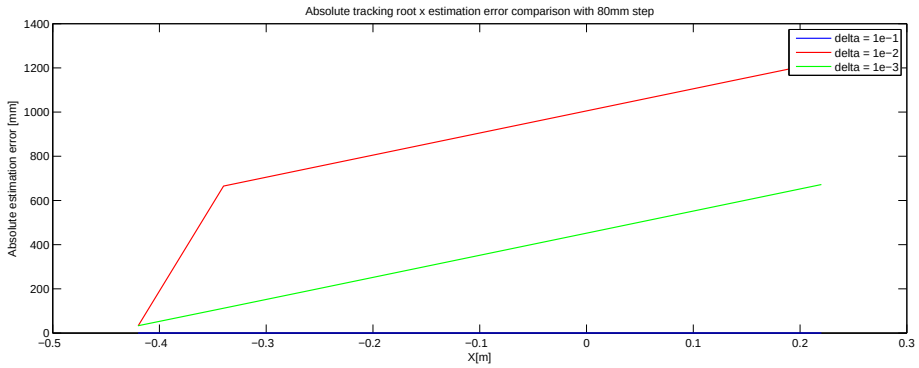


b) Stima di Y

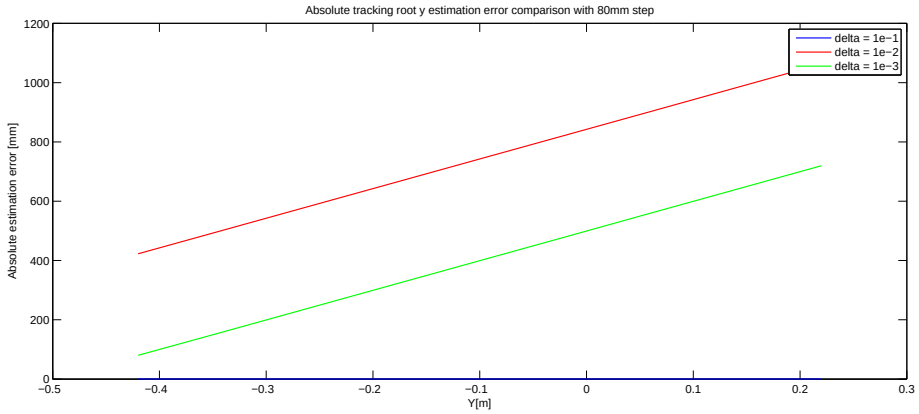


c) Stima di Z

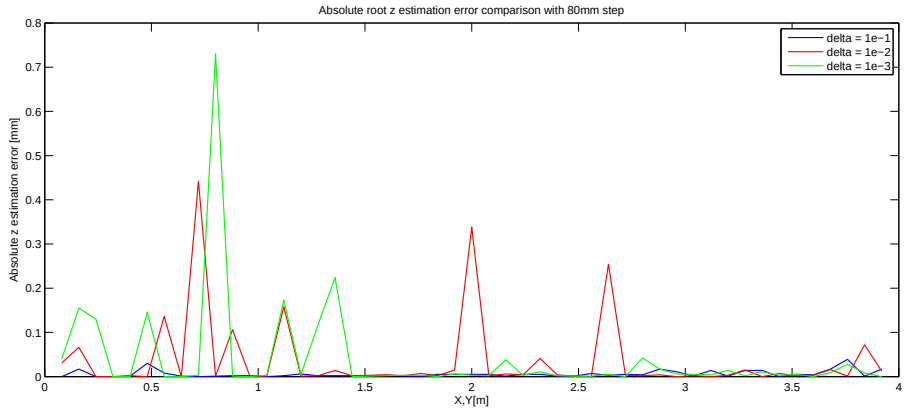
Figura A.23: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 50mm in modalità tracking (ingrandimento).



a) Stima di X

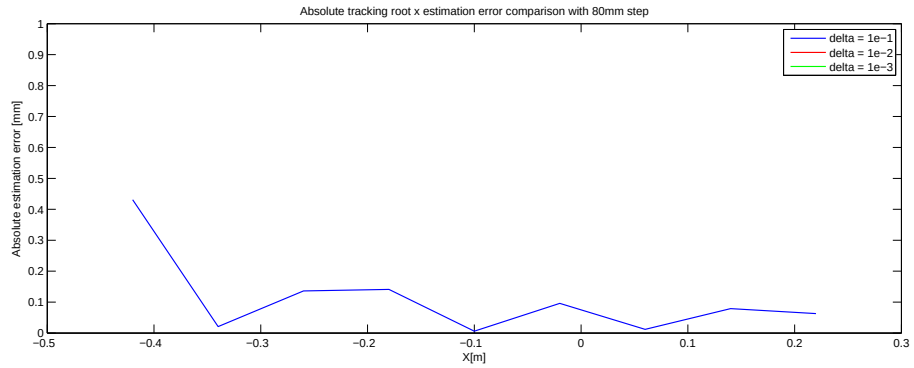


b) Stima di Y

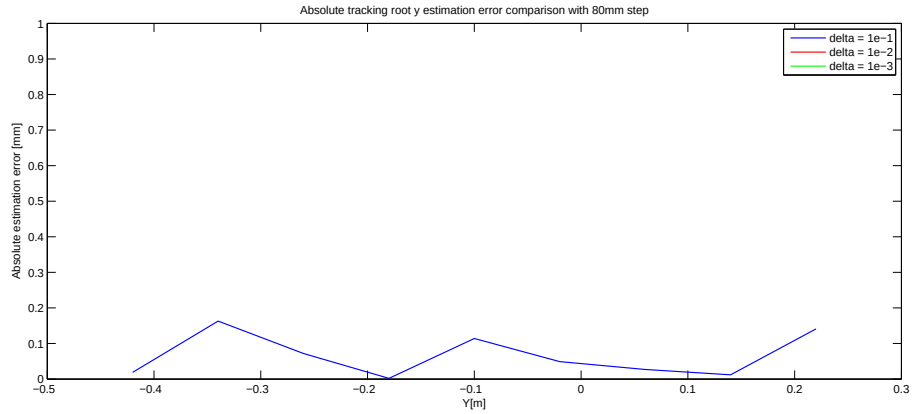


c) Stima di Z

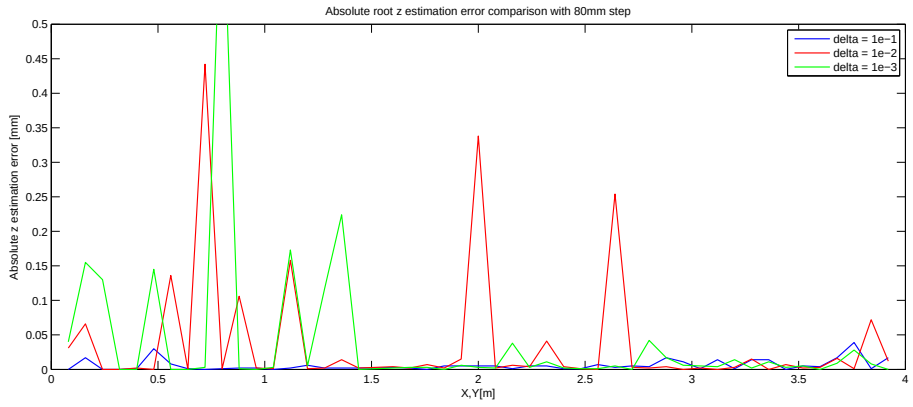
Figura A.24: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 80mm in modalità tracking.



a) Stima di X



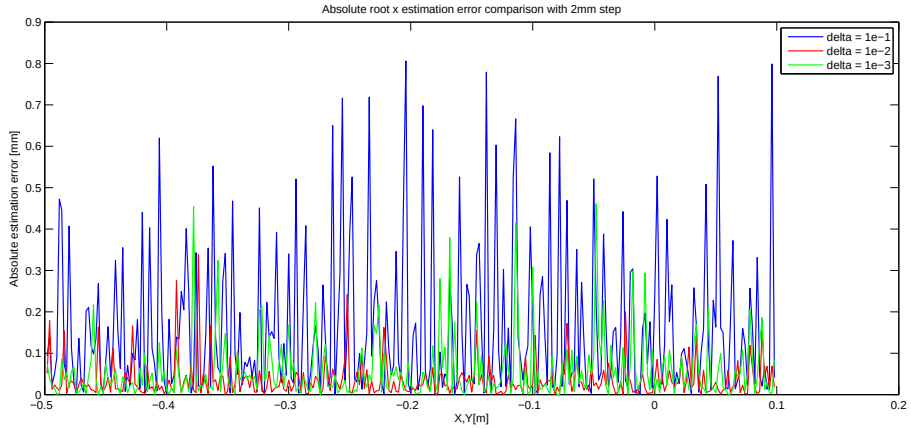
b) Stima di Y



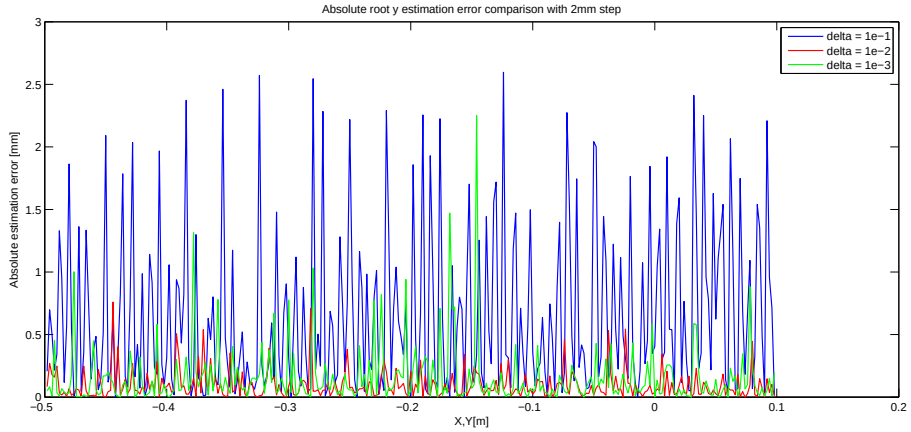
c) Stima di Z

Figura A.25: Confronto errore assoluto di stima di X, Y e Z per tre diversi valori di Δ e spostamenti di 80mm in modalità tracking (ingrandimento).

A.5 Grafici per il Test 5

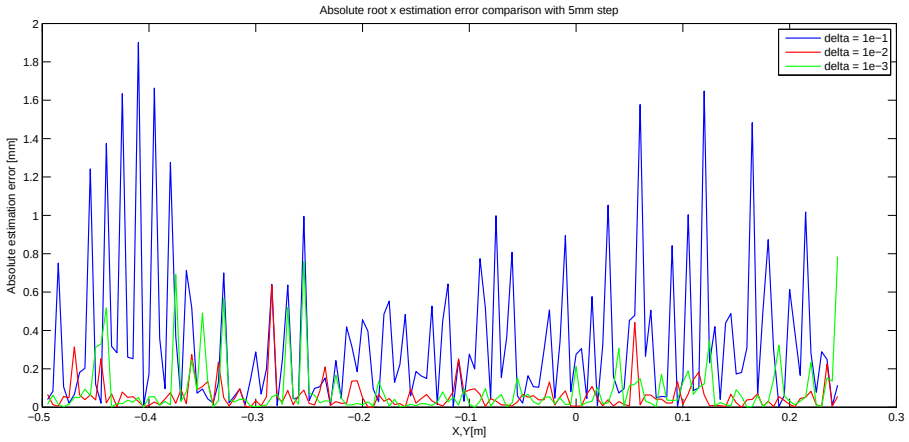


a) Stima di X

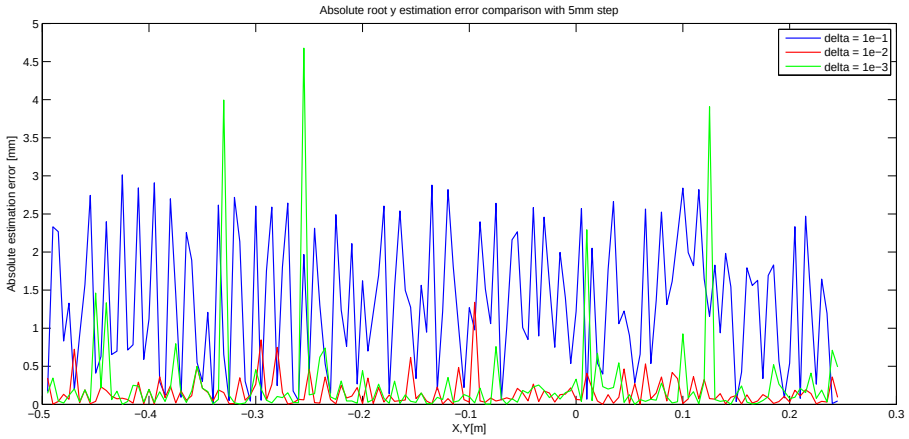


b) Stima di Y

Figura A.26: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 2mm.

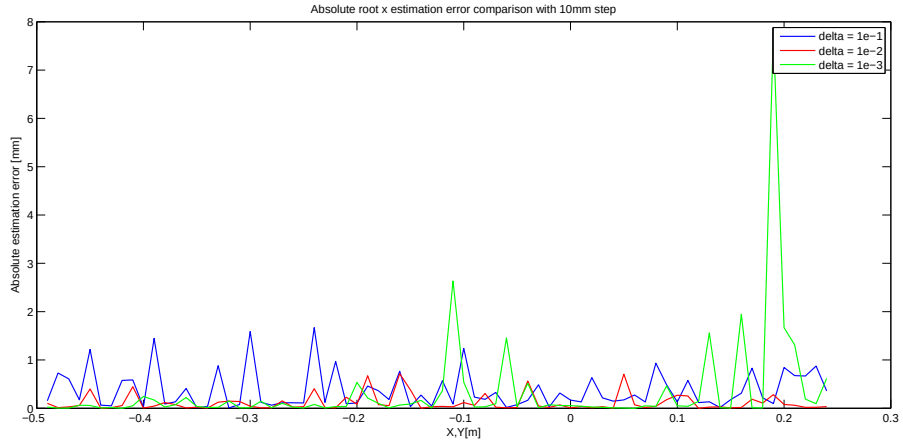


a) Stima di X

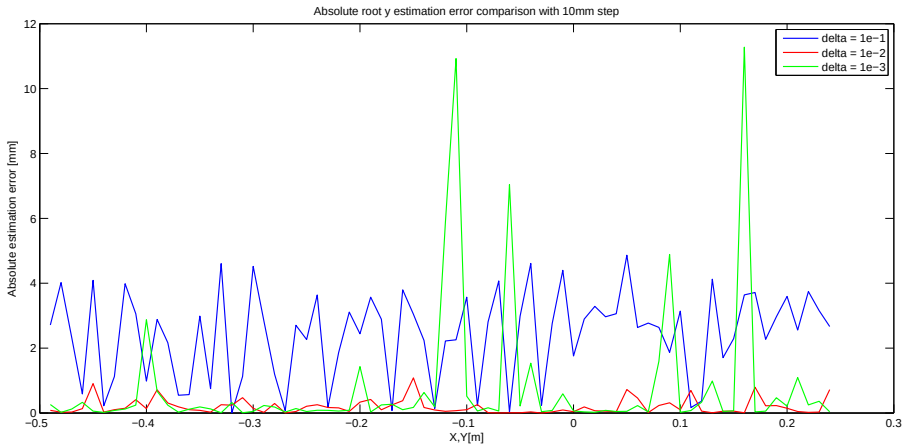


b) Stima di Y

Figura A.27: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 5mm.

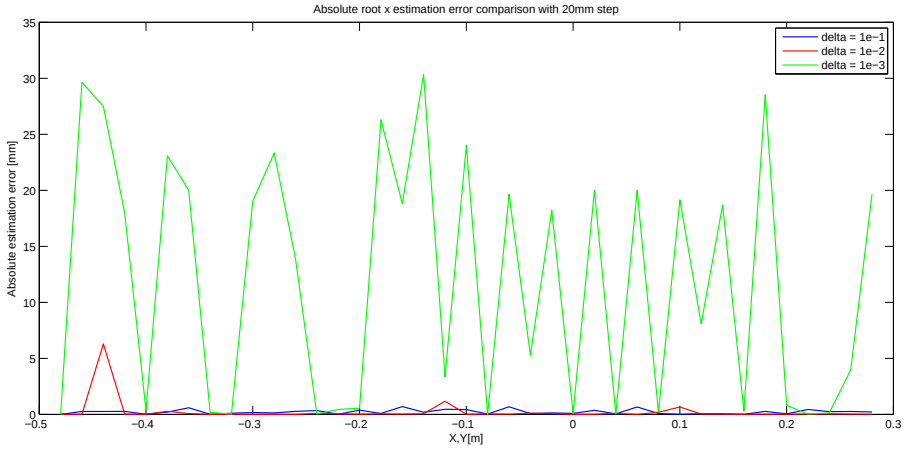


a) Stima di X

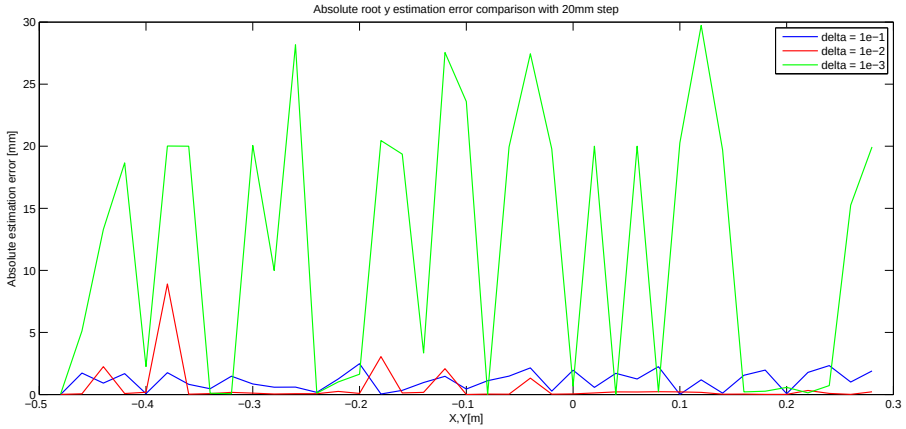


b) Stima di Y

Figura A.28: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 10mm.

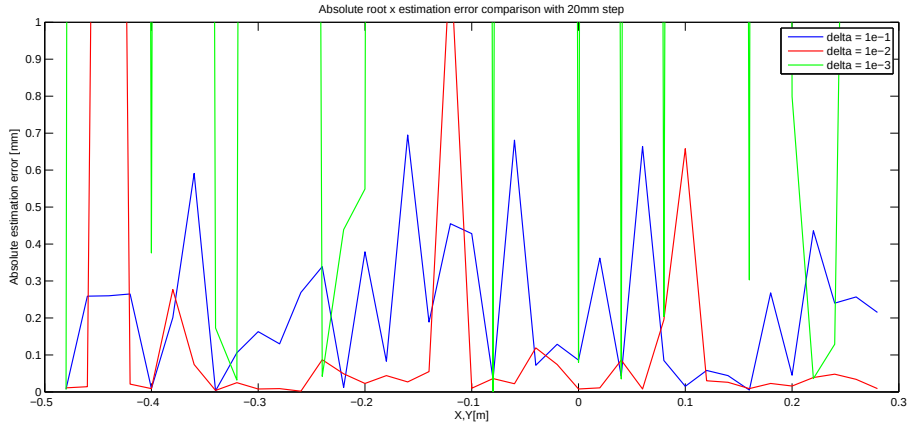


a) Stima di X

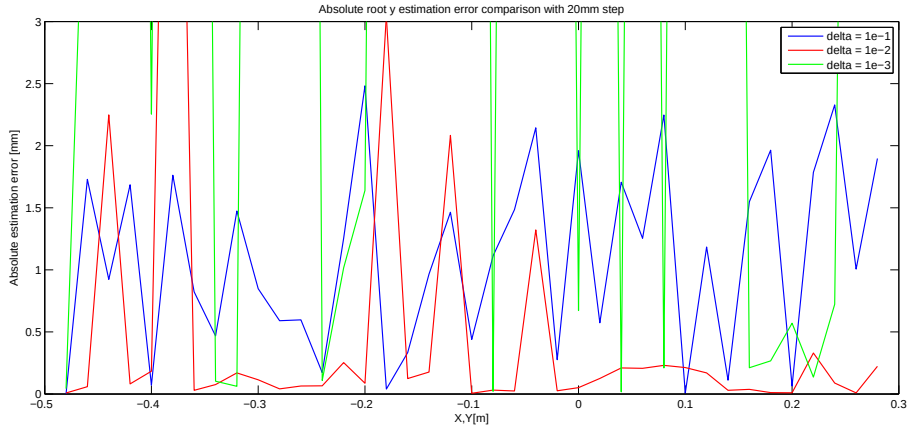


b) Stima di Y

Figura A.29: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 20mm.

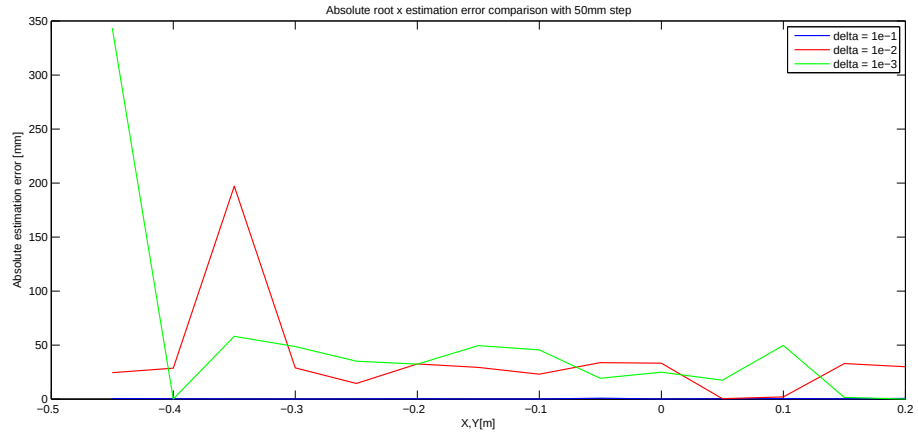


a) Stima di X

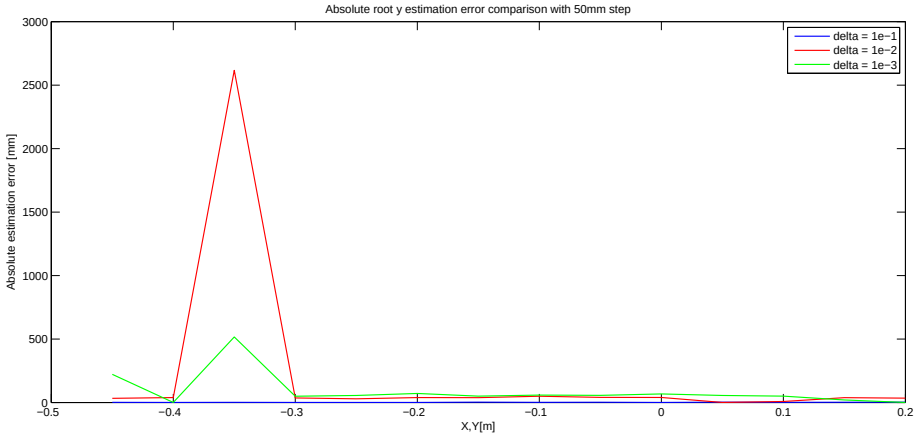


b) Stima di Y

Figura A.30: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 20mm (ingrandimento).

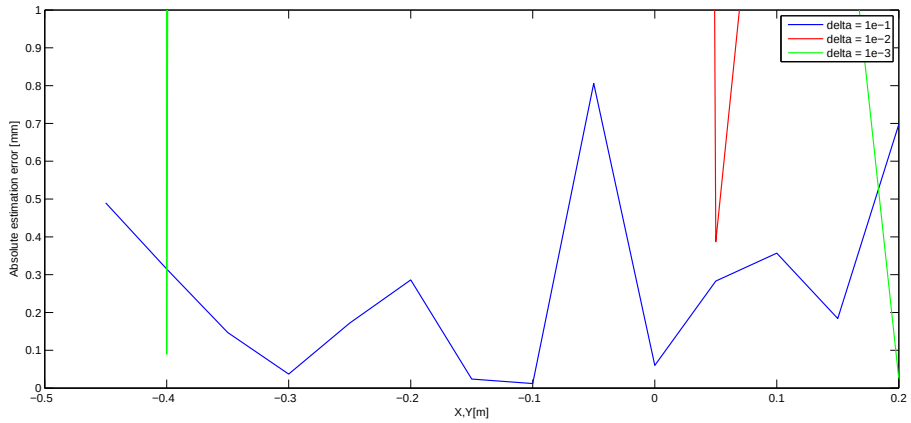


a) Stima di X

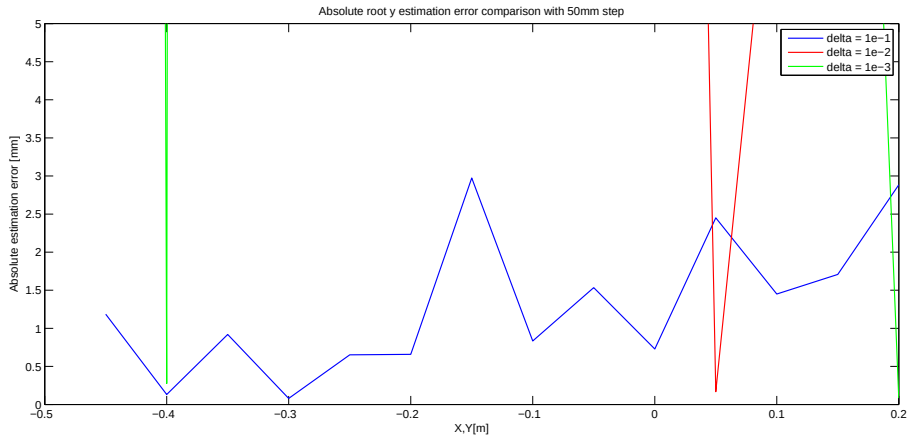


b) Stima di Y

Figura A.31: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 50mm.

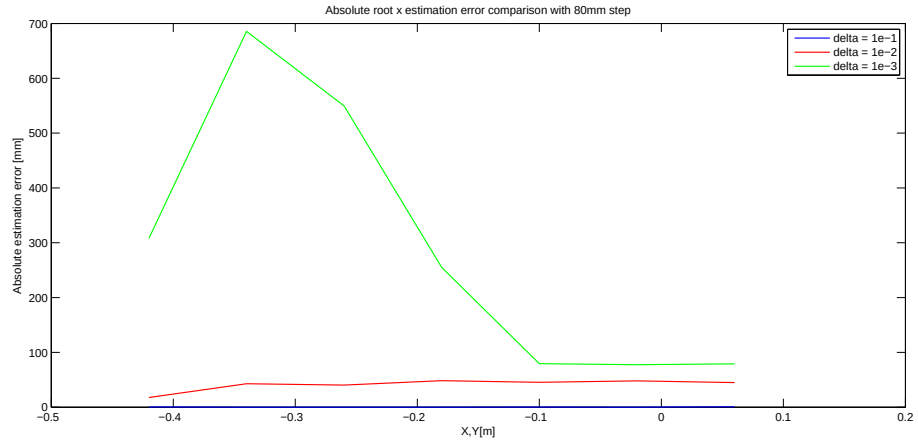


a) Stima di X

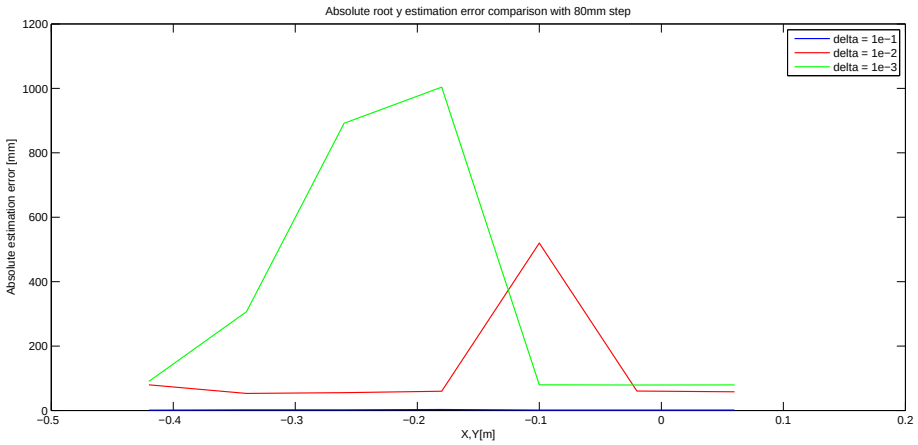


b) Stima di Y

Figura A.32: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 50mm (ingrandimento).

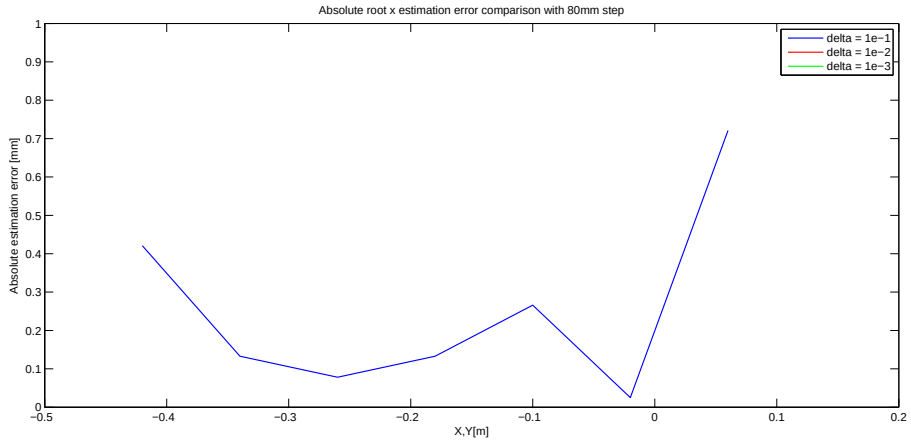


a) Stima di X

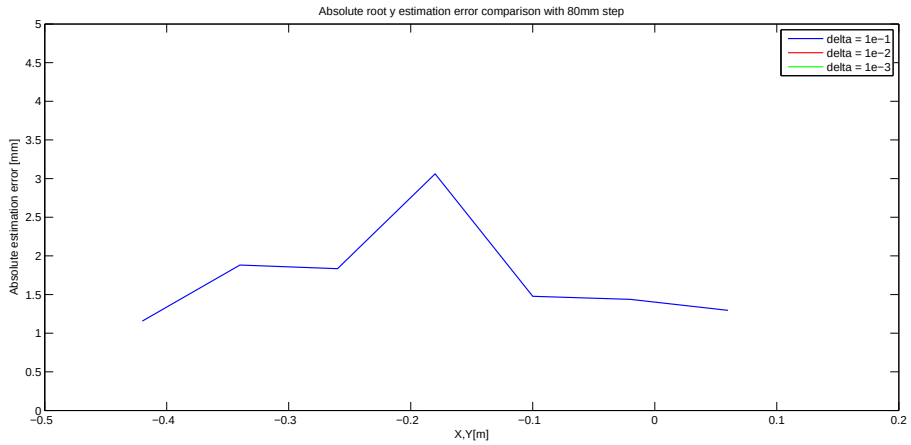


b) Stima di Y

Figura A.33: Confronto errore assoluto di stima di X (a)) e di Y (b))per tre diversi valori di Δ e spostamenti di 80mm.



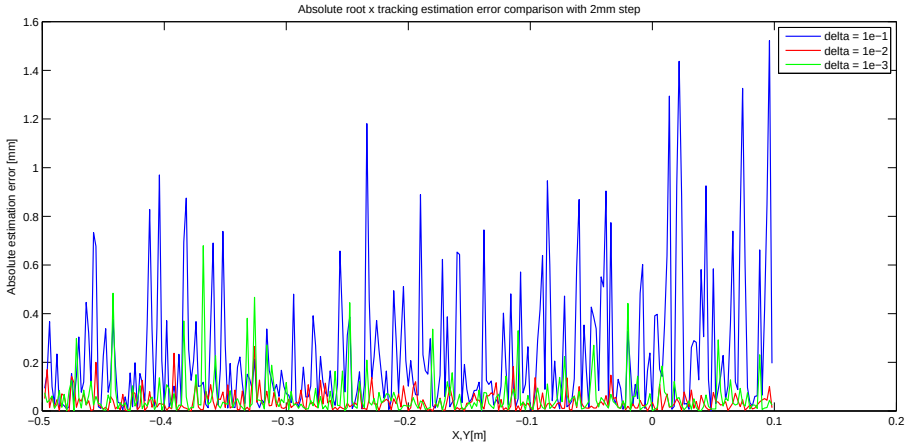
a) Stima di X



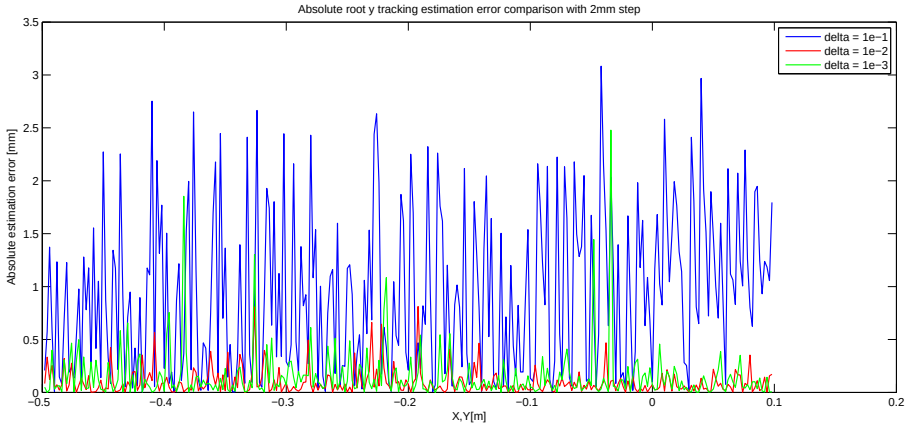
b) Stima di Y

Figura A.34: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 80mm (ingrandimento).

A.6 Grafici per il Test 6

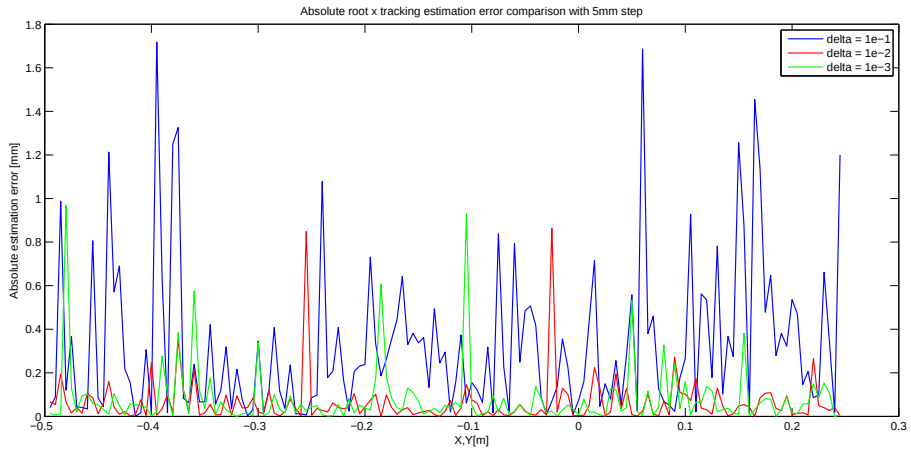


a) Stima di X

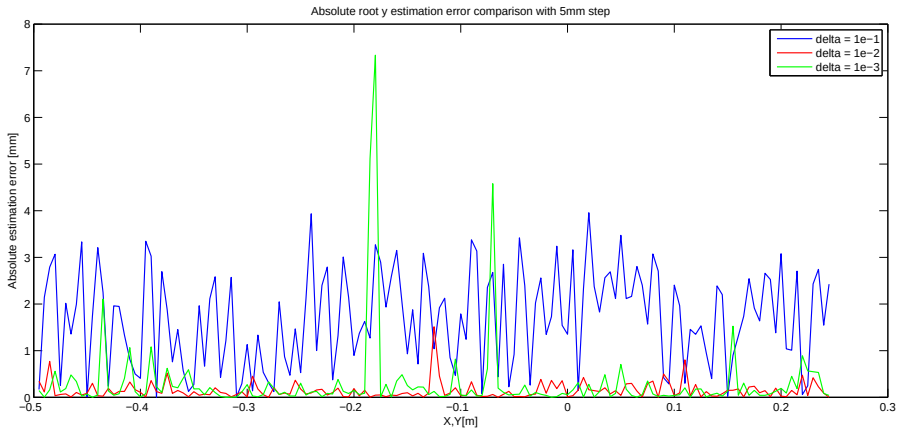


b) Stima di Y

Figura A.35: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 2mm in modalità tracking.

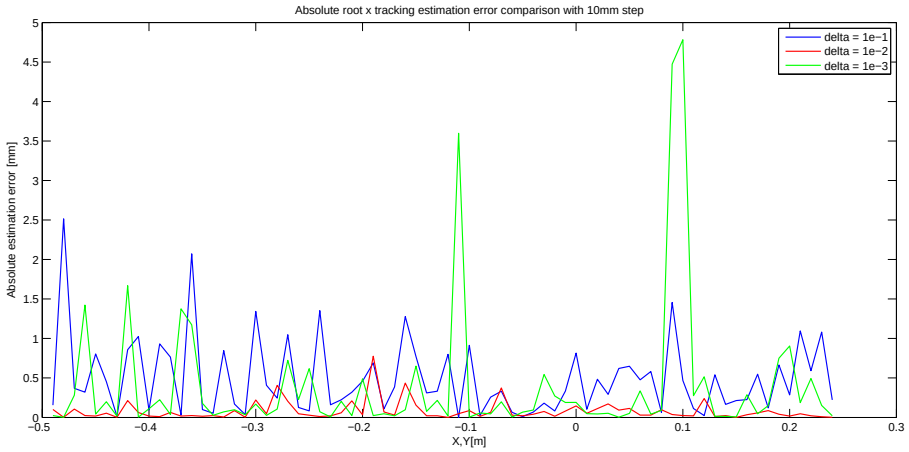


a) Stima di X

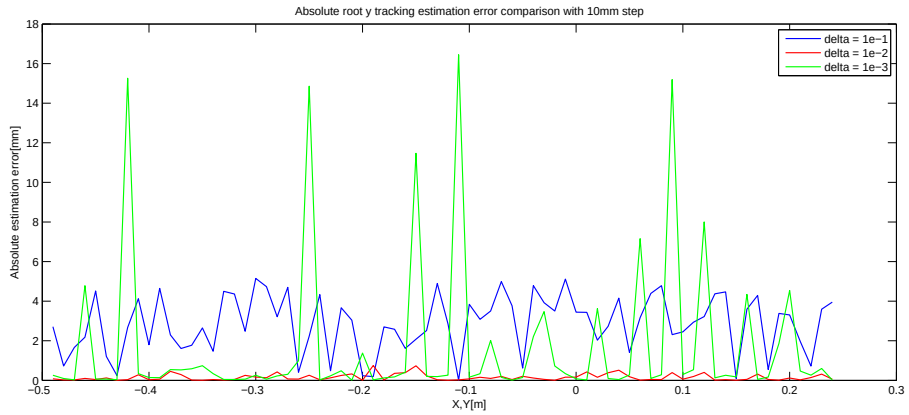


b) Stima di Y

Figura A.36: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 5mm in modalità tracking.

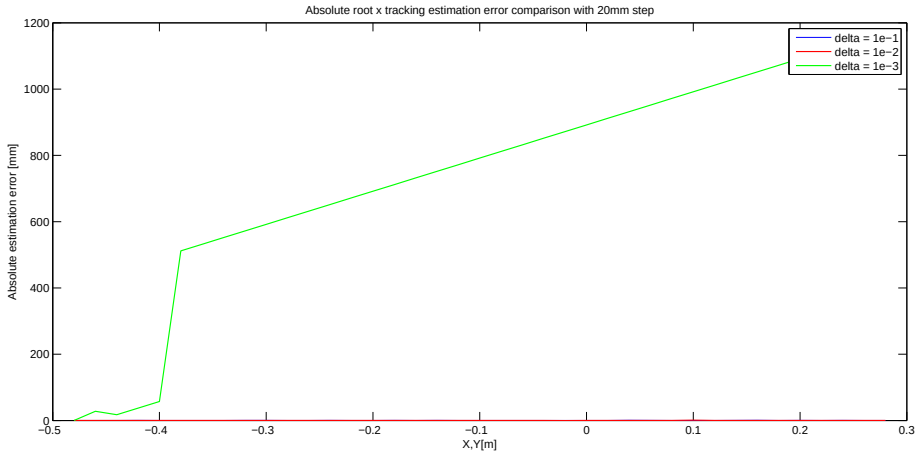


a) Stima di X

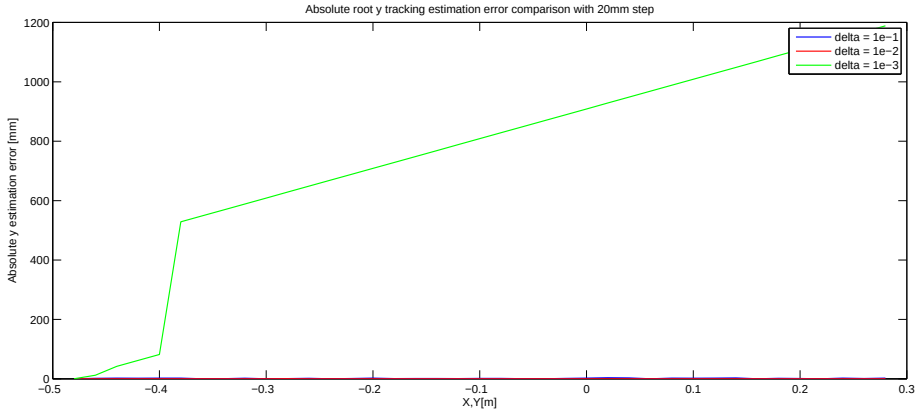


b) Stima di Y

Figura A.37: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 10mm in modalità tracking.

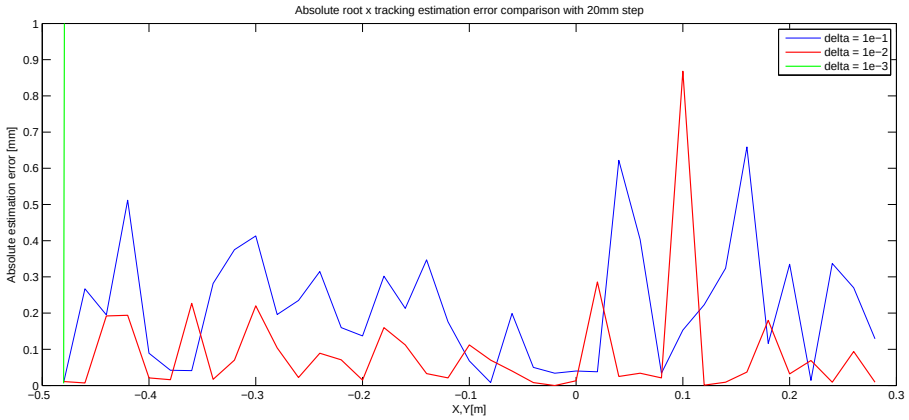


a) Stima di X

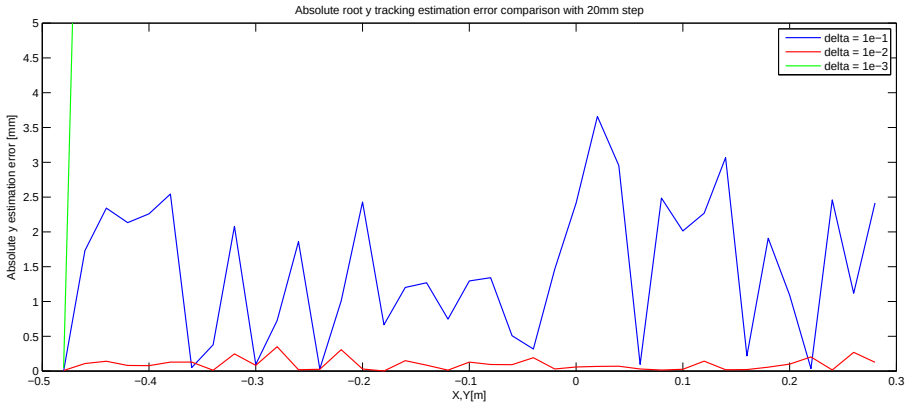


b) Stima di Y

Figura A.38: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 20mm in modalità tracking.

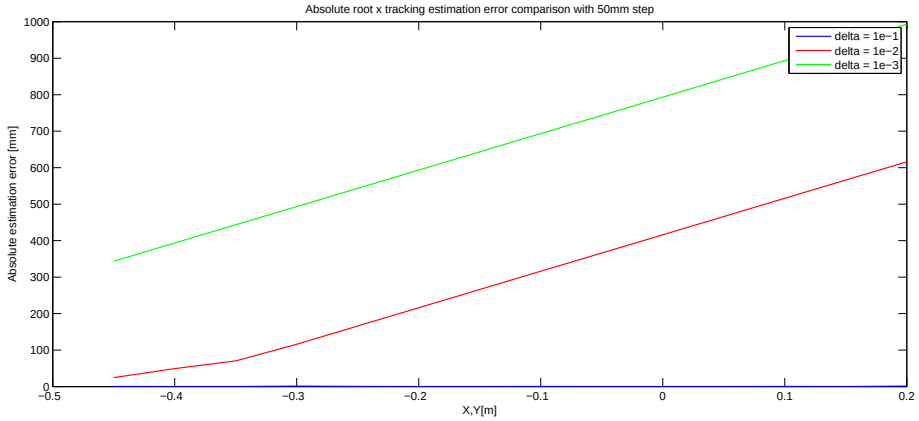


a) Stima di X

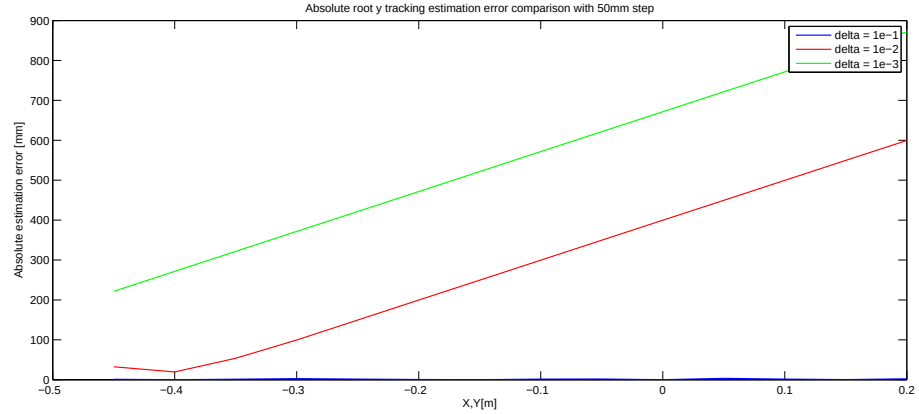


b) Stima di Y

Figura A.39: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 20mm in modalità tracking (ingrandimento).

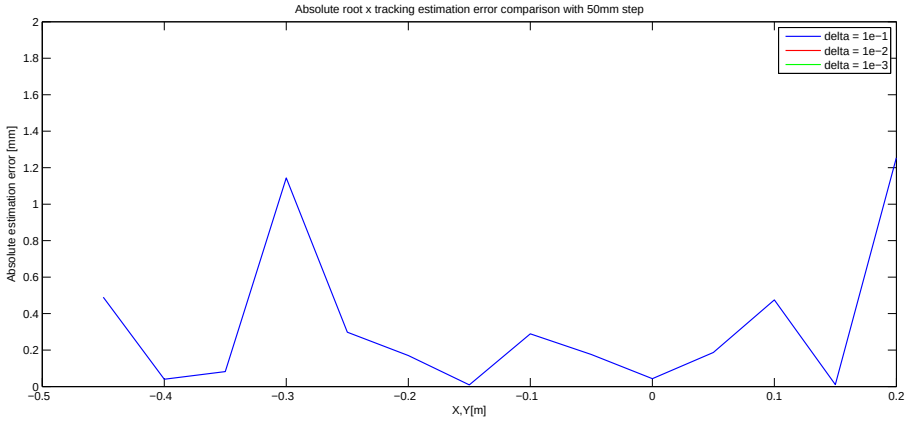


a) Stima di X

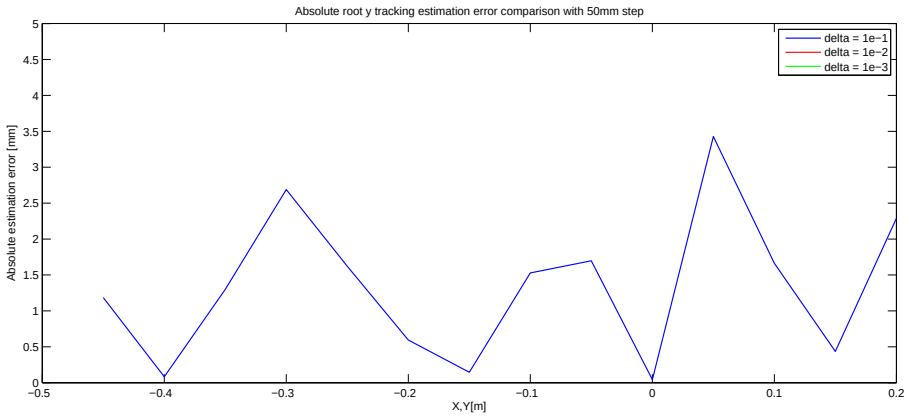


b) Stima di Y

Figura A.40: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 50mm in modalità tracking.

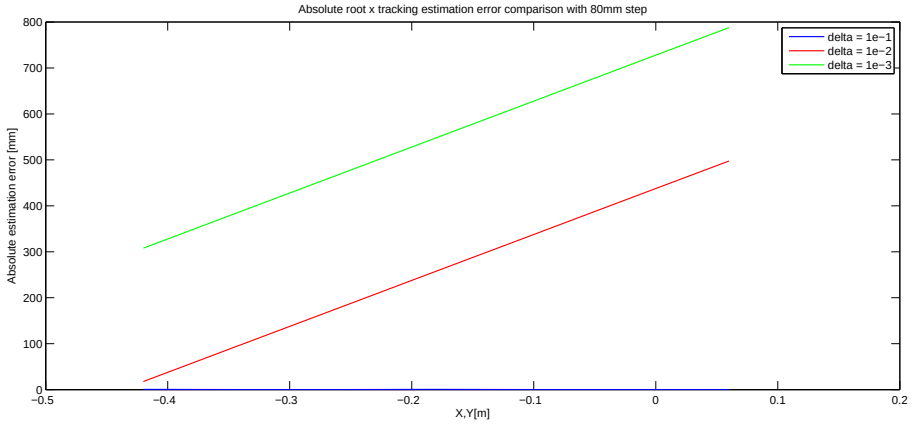


a) Stima di X

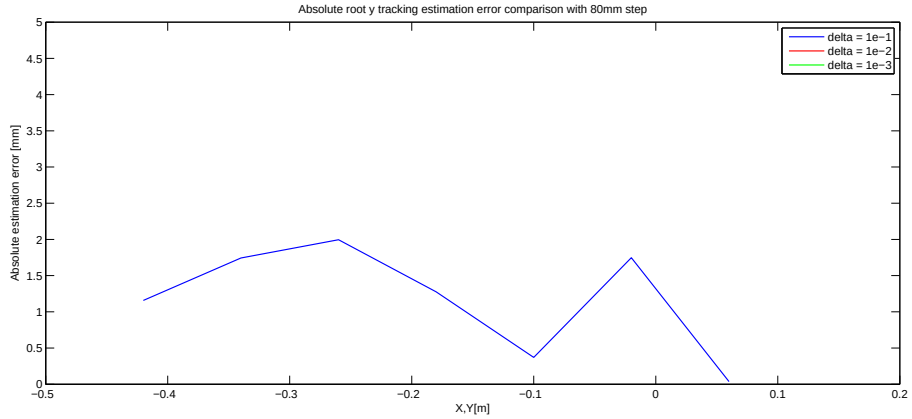


b) Stima di Y

Figura A.41: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 50mm in modalità tracking (ingrandimento).

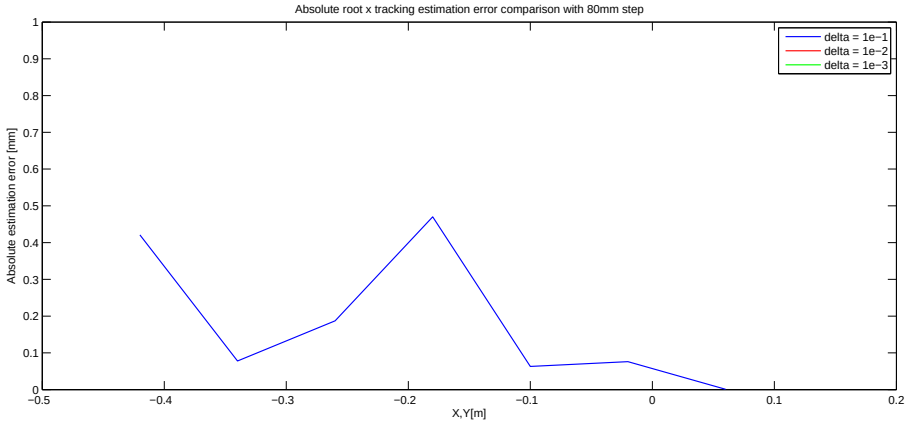


a) Stima di X

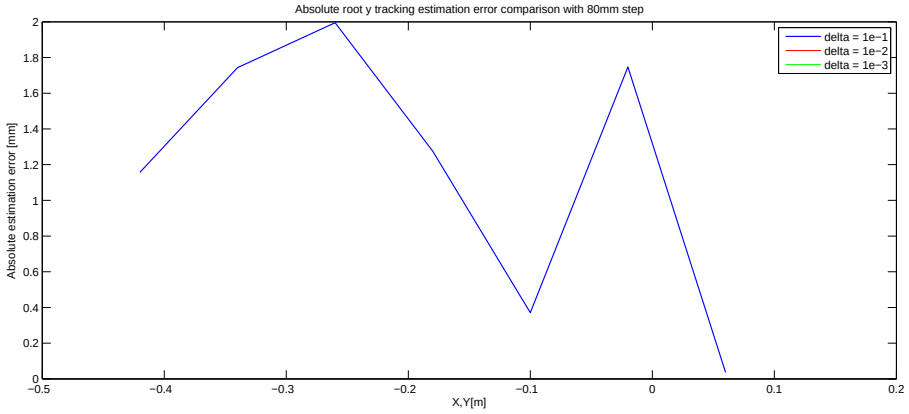


b) Stima di Y

Figura A.42: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 80mm in modalità tracking.



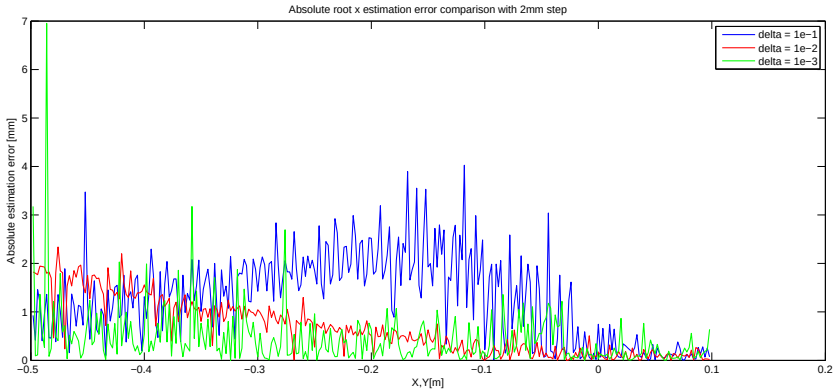
a) Stima di X



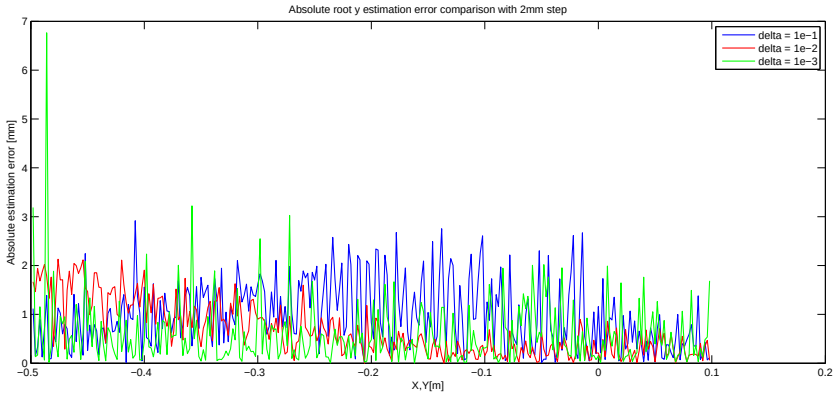
b) Stima di Y

Figura A.43: Confronto errore assoluto di stima di X e di Y per tre diversi valori di Δ e spostamenti di 80mm in modalità tracking (ingrandimento).

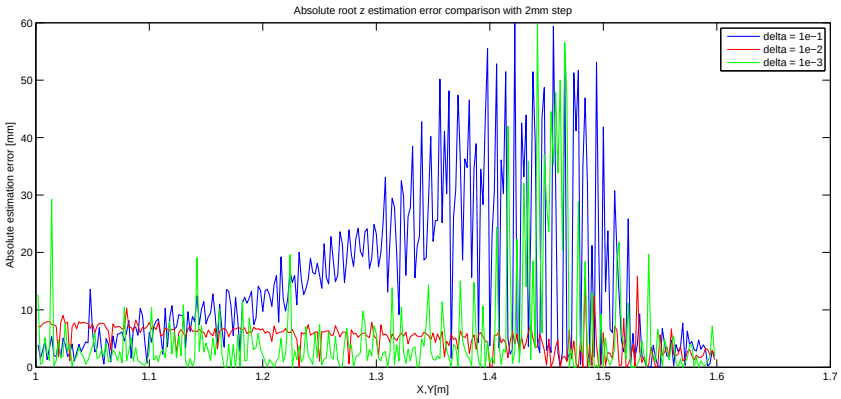
A.7 Grafici per il Test 7



a) Stima di X

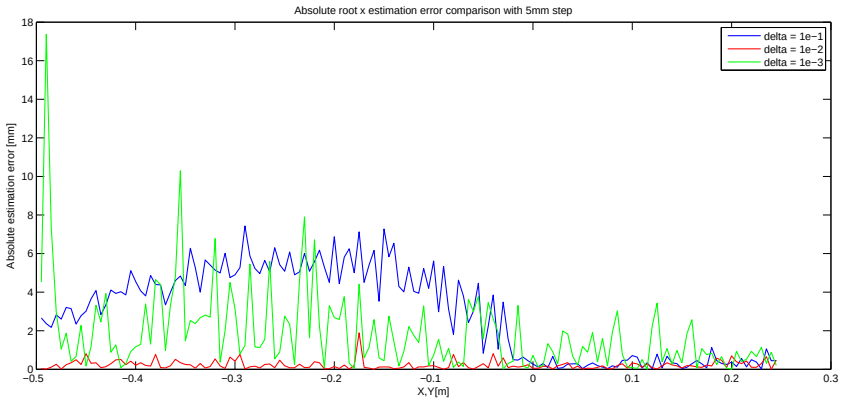


b) Stima di Y

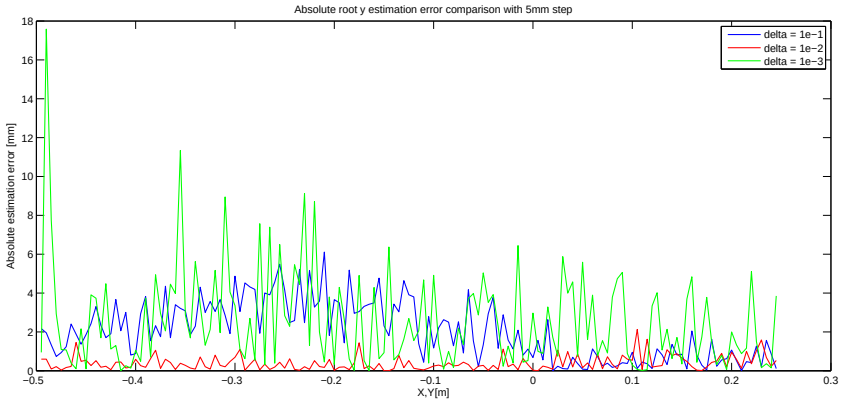


c) Stima di Z

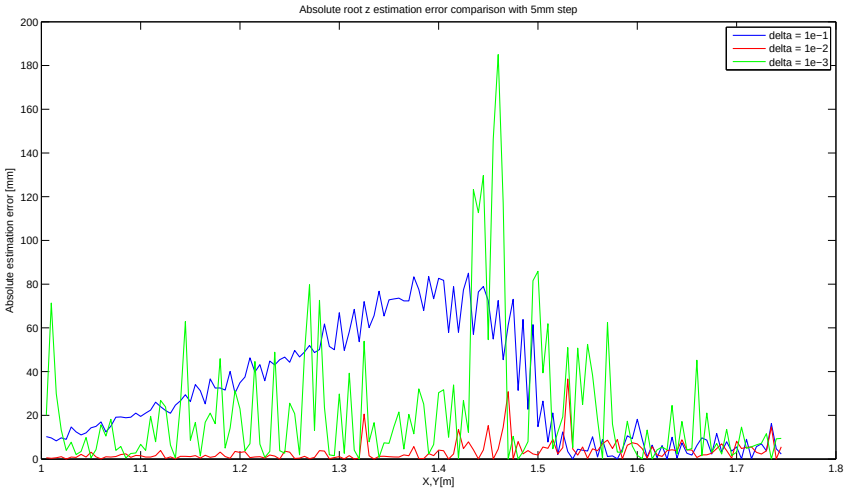
Figura A.44: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 2mm.



a) Stima di X

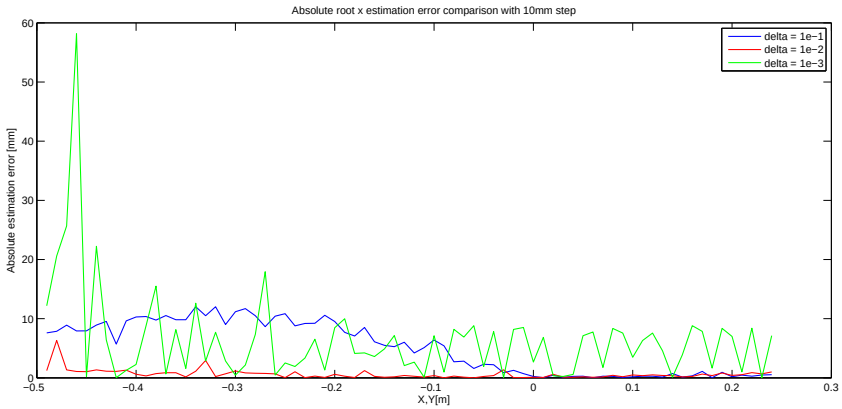


b) Stima di Y

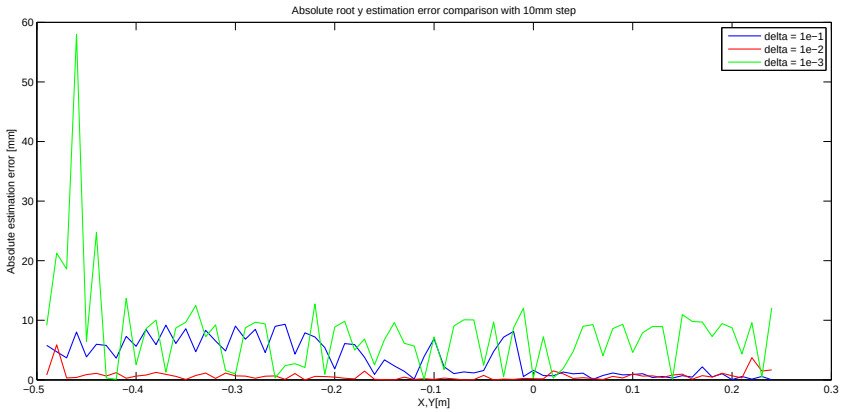


c) Stima di Z

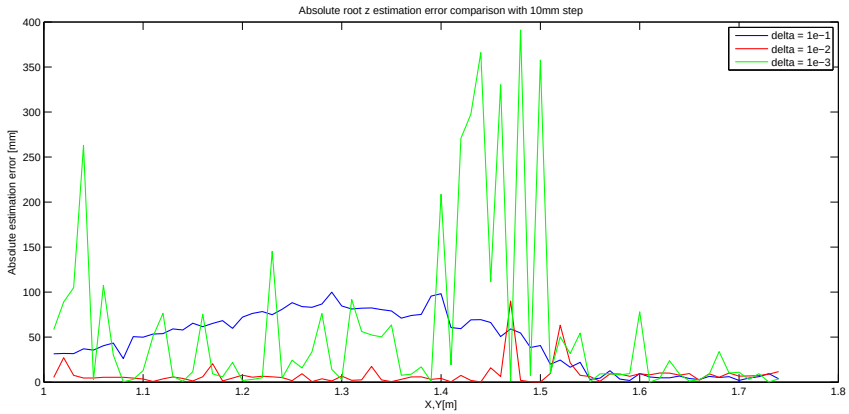
Figura A.45: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 5mm.



a) Stima di X

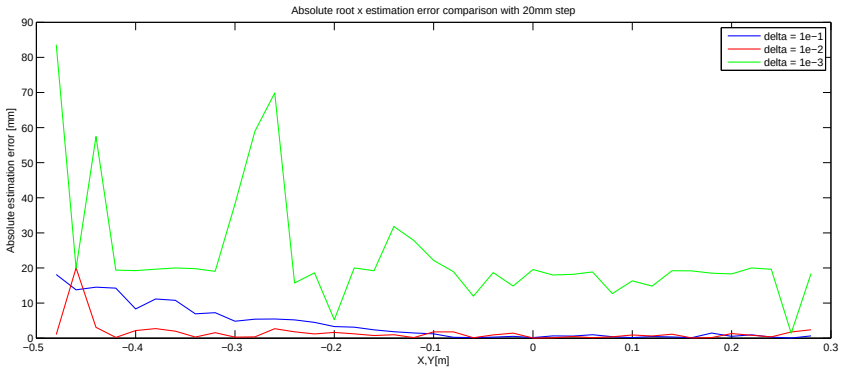


b) Stima di Y

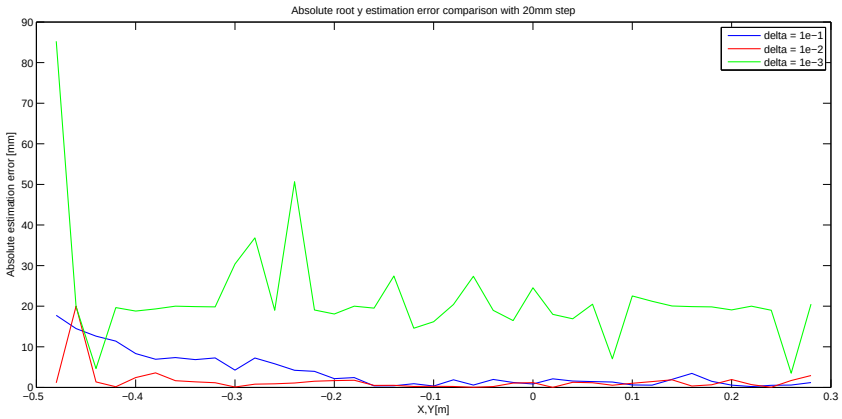


c) Stima di Z

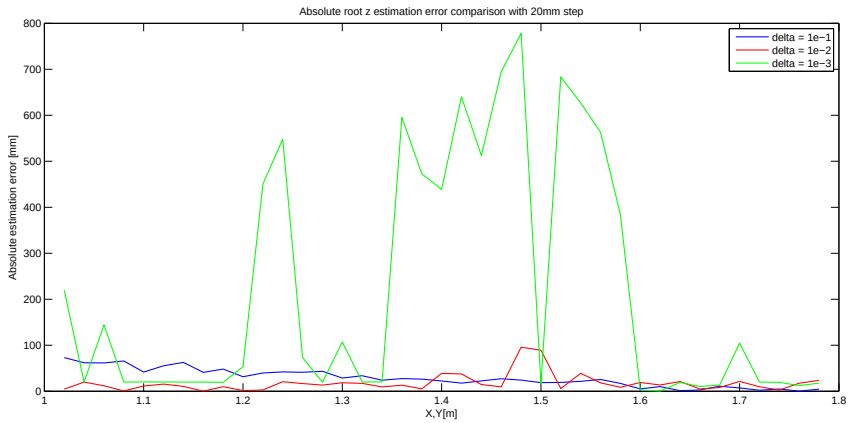
Figura A.46: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 10mm.



a) Stima di X

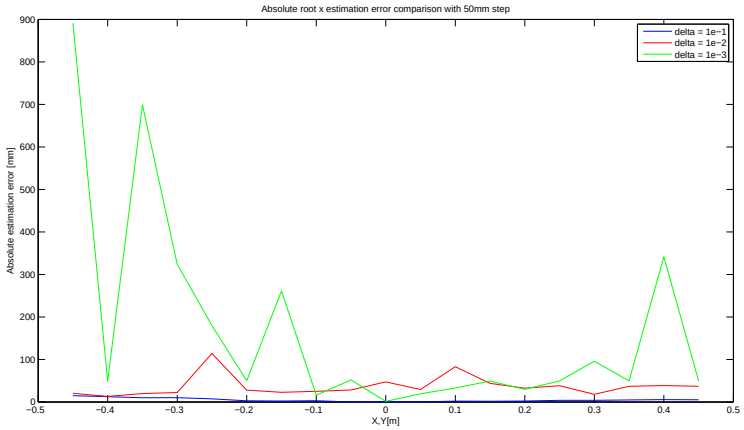


b) Stima di Y

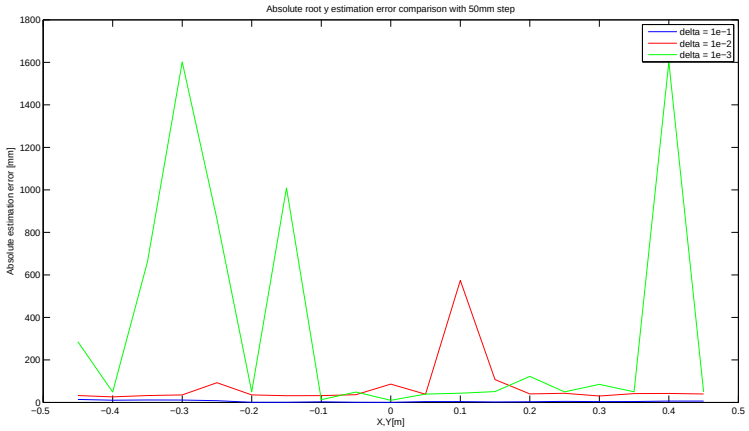


c) Stima di Z

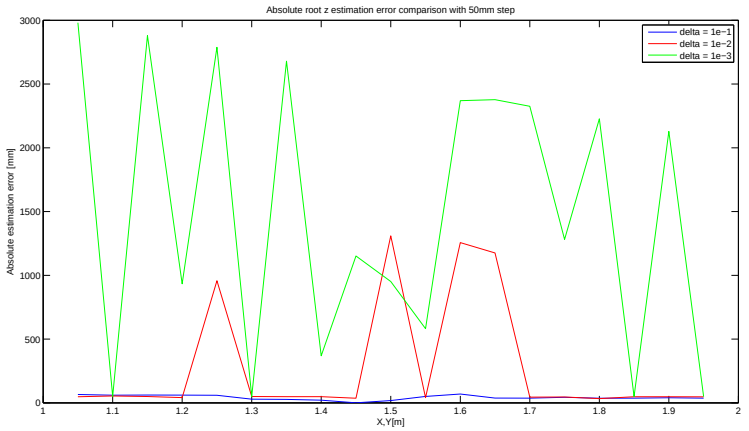
Figura A.47: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 20mm.



a) Stima di X

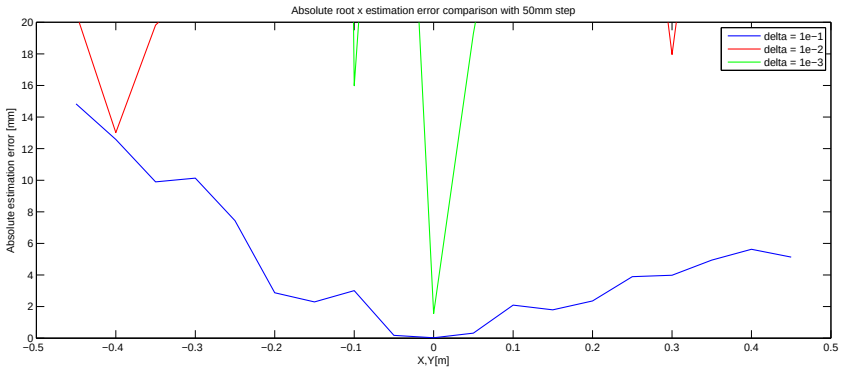


b) Stima di Y

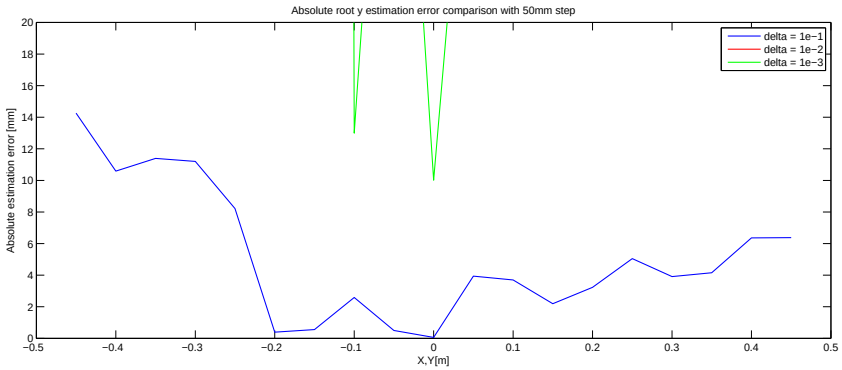


c) Stima di Z

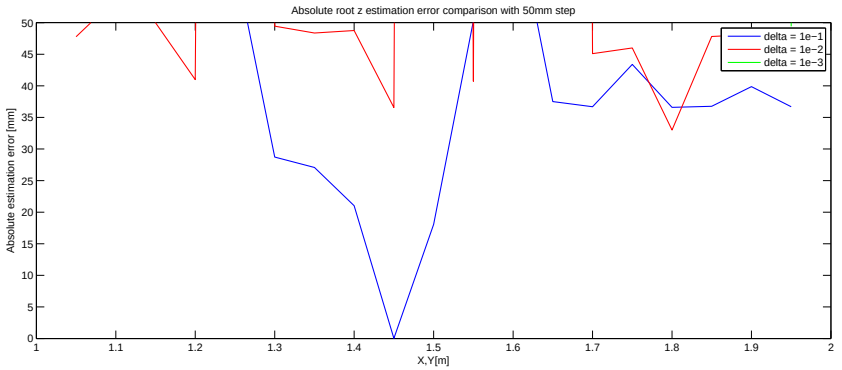
Figura A.48: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 50mm.



a) Stima di X

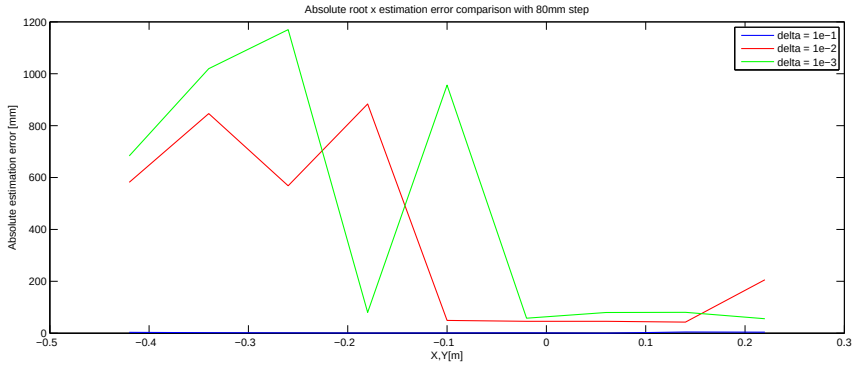


b) Stima di Y

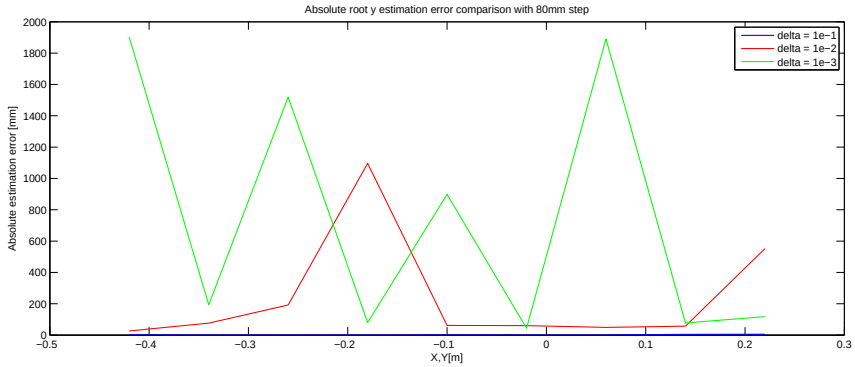


c) Stima di Z

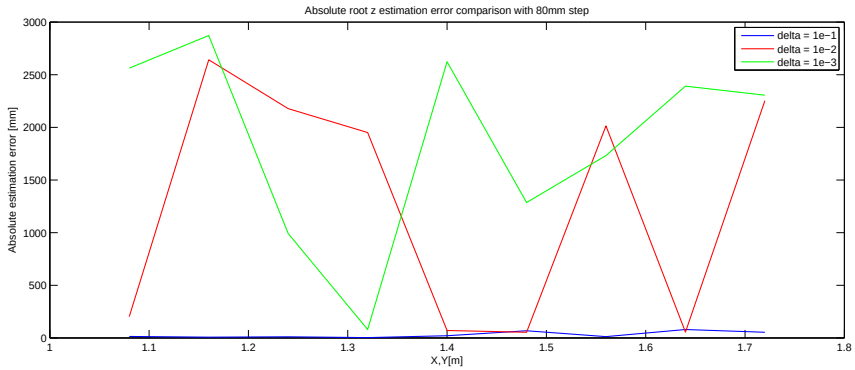
Figura A.49: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 50mm (ingrandimento).



a) Stima di X

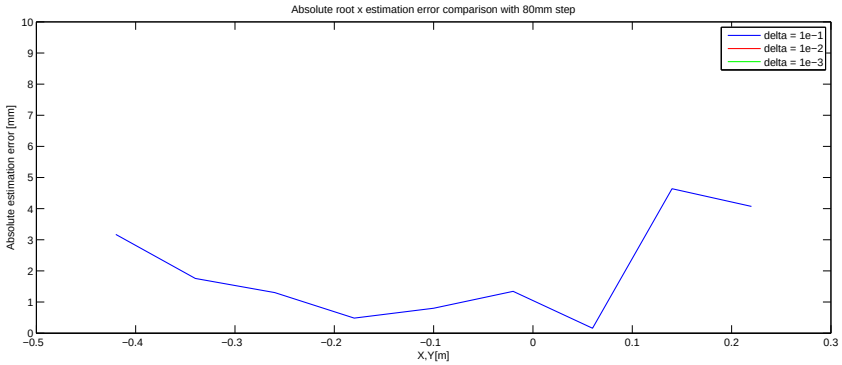


b) Stima di Y

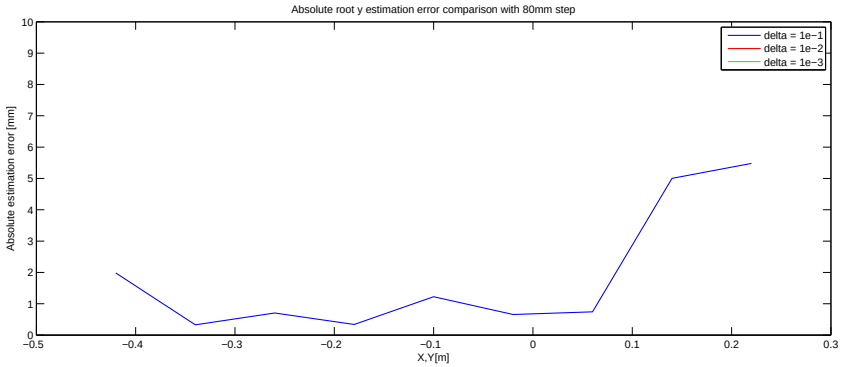


c) Stima di Z

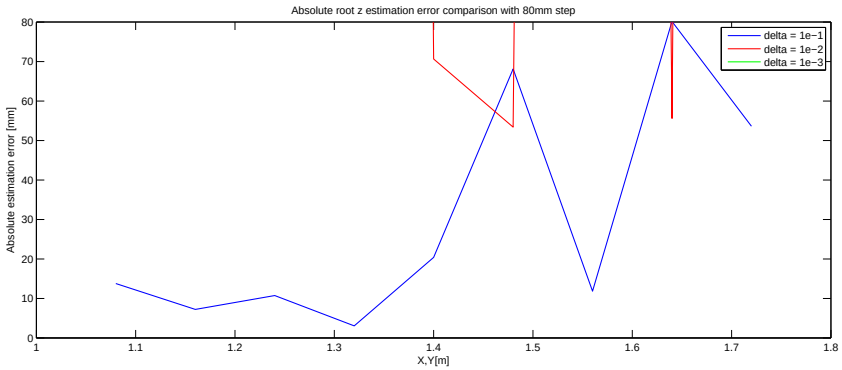
Figura A.50: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 80mm.



a) Stima di X



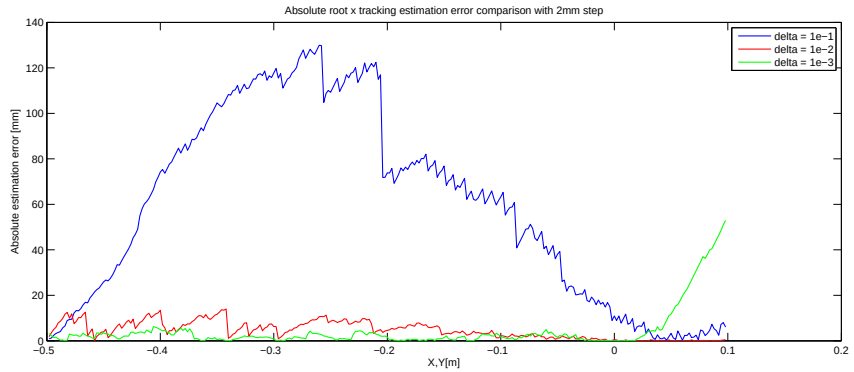
b) Stima di Y



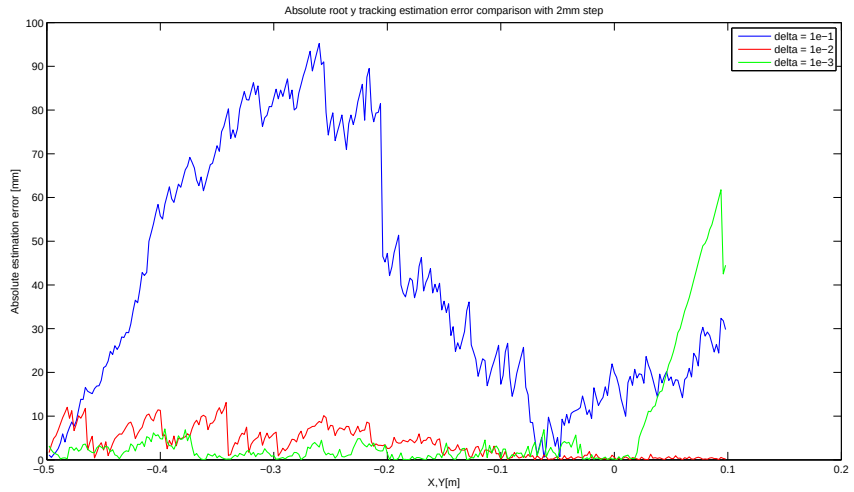
c) Stima di Z

Figura A.51: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 80mm (ingrandimento).

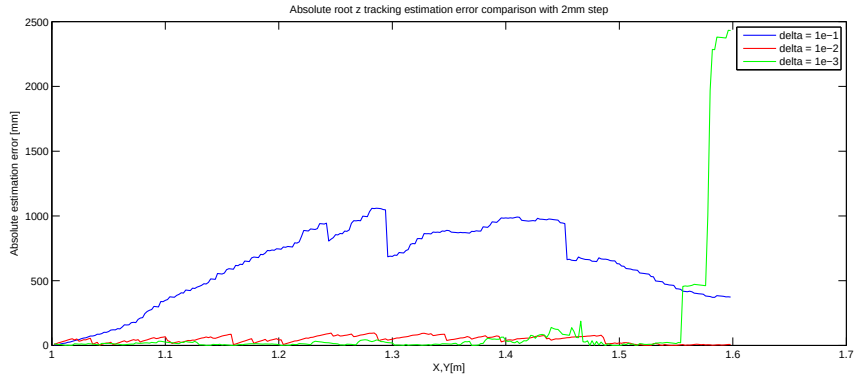
A.8 Grafici per il Test 8



a) Stima di X

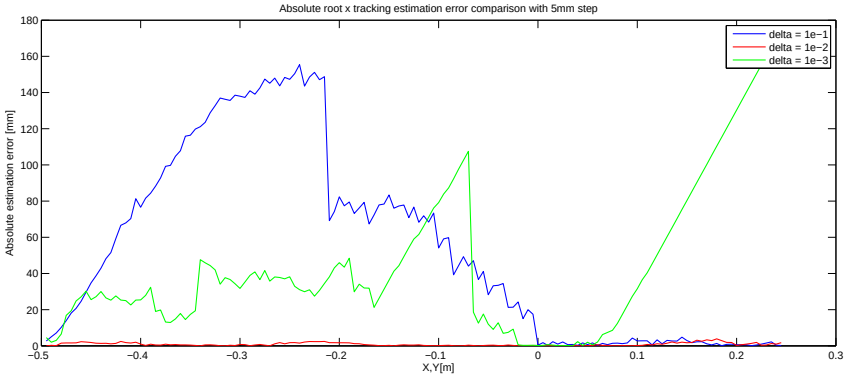


b) Stima di Y

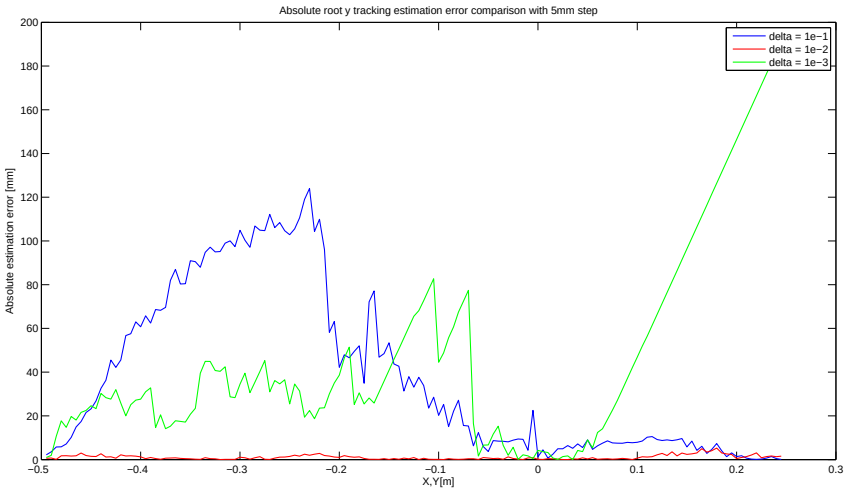


c) Stima di Z

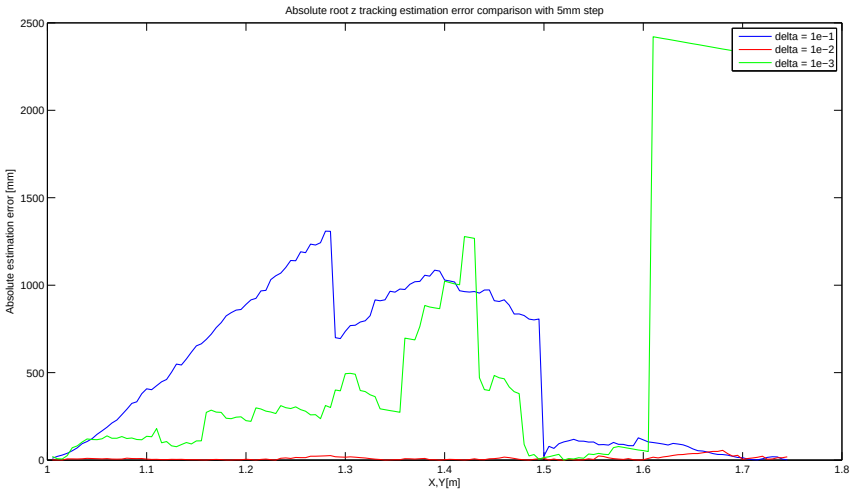
Figura A.52: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 2mm in modalità tracking.



a) Stima di X

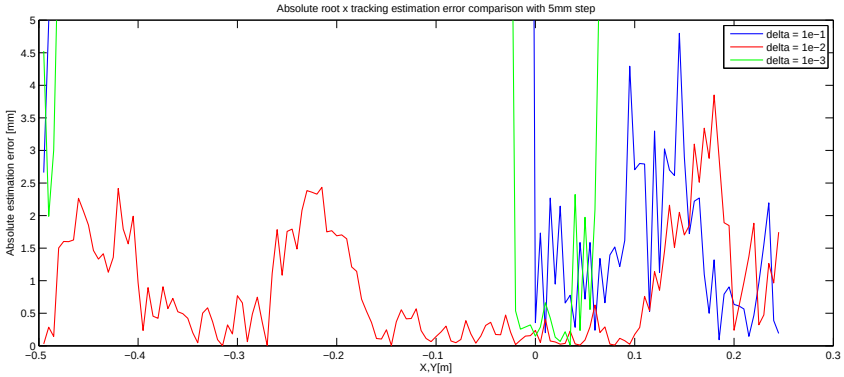


b) Stima di Y

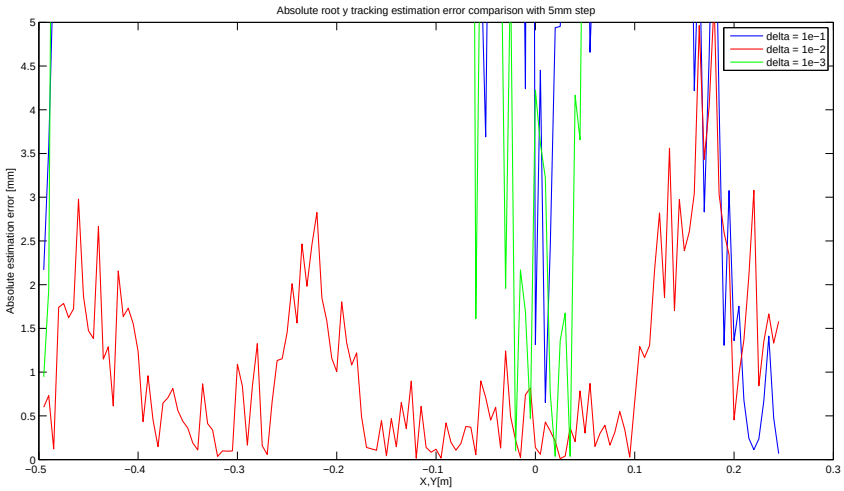


c) Stima di Z

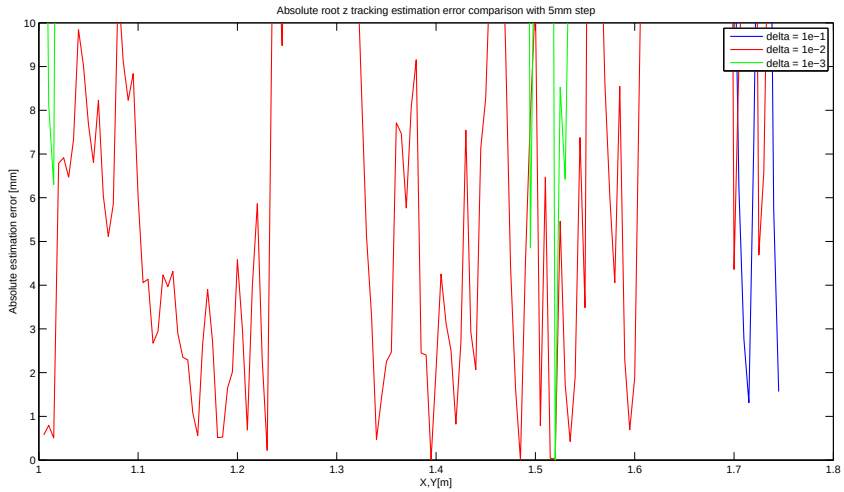
Figura A.53: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 5mm in modalità tracking.



a) Stima di X

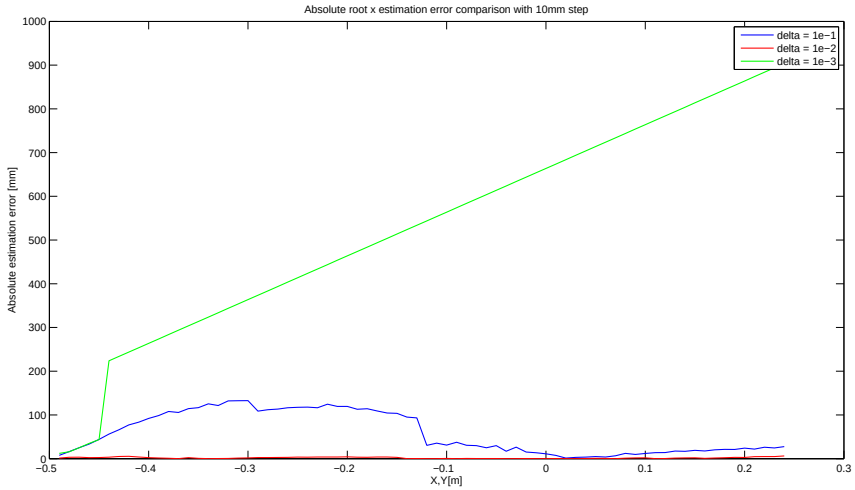


b) Stima di Y

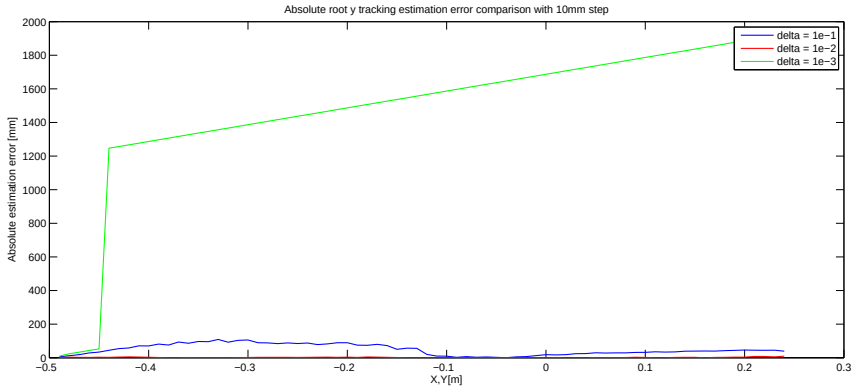


c) Stima di Z

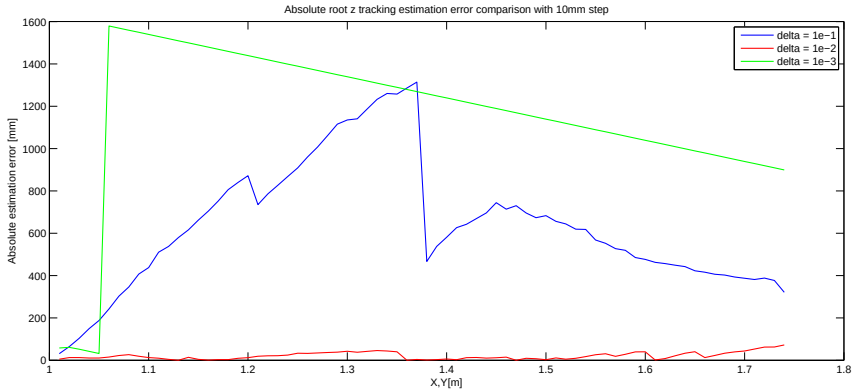
Figura A.54: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 5mm in modalità tracking (ingrandimento).



a) Stima di X

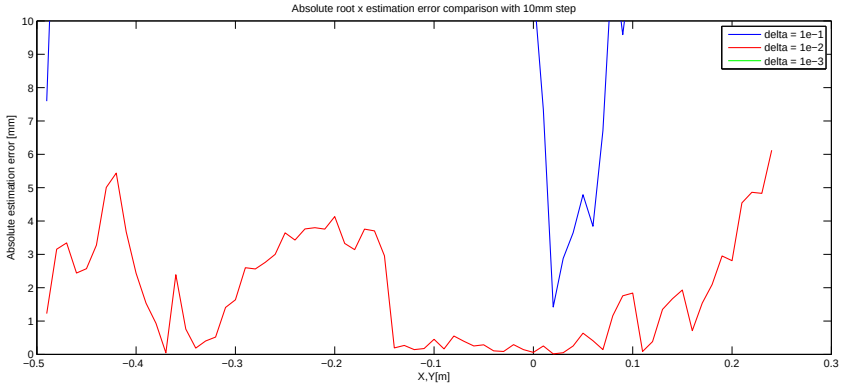


b) Stima di Y

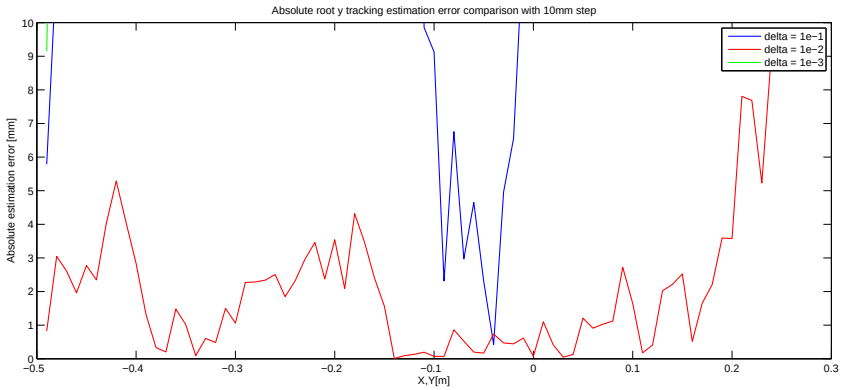


c) Stima di Z

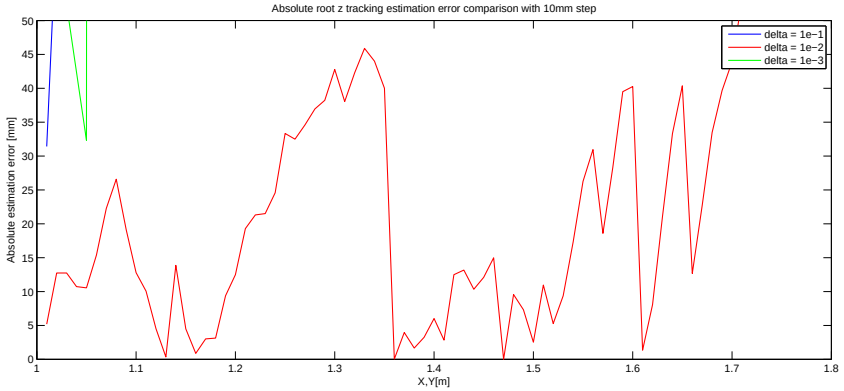
Figura A.55: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 10mm in modalità tracking.



a) Stima di X

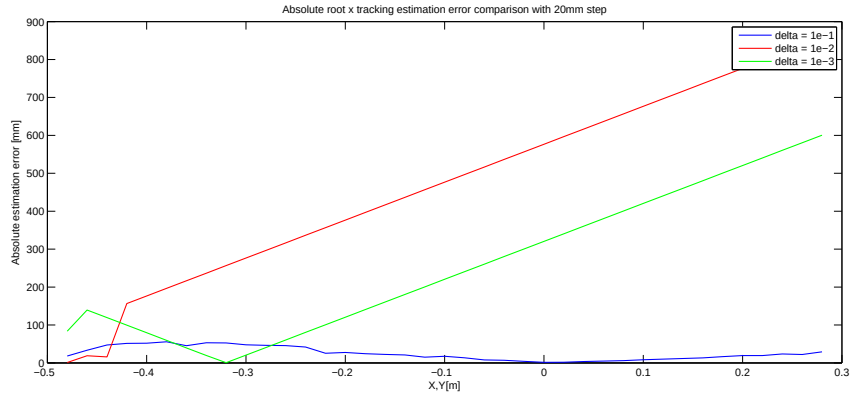


b) Stima di Y

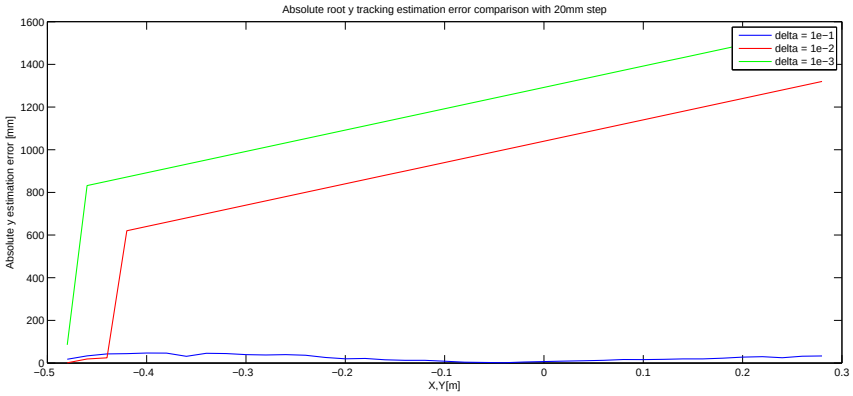


c) Stima di Z

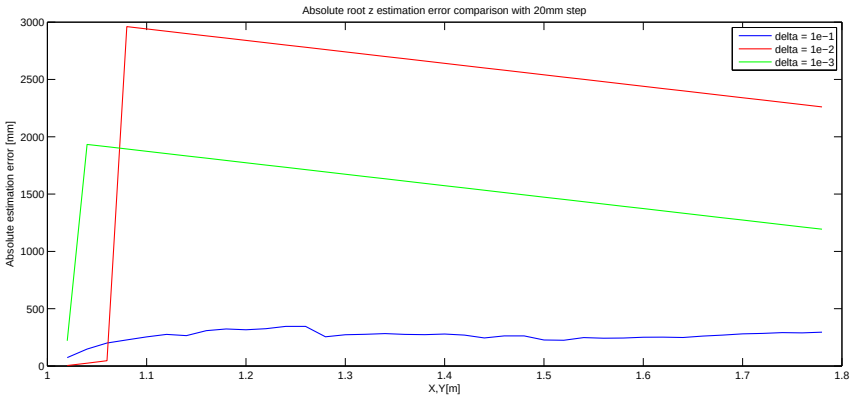
Figura A.56: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 10mm in modalità tracking (ingrandimento).



a) Stima di X

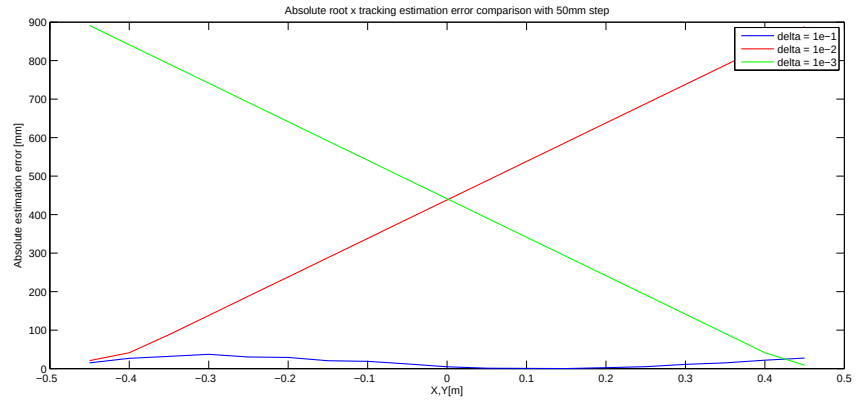


b) Stima di Y

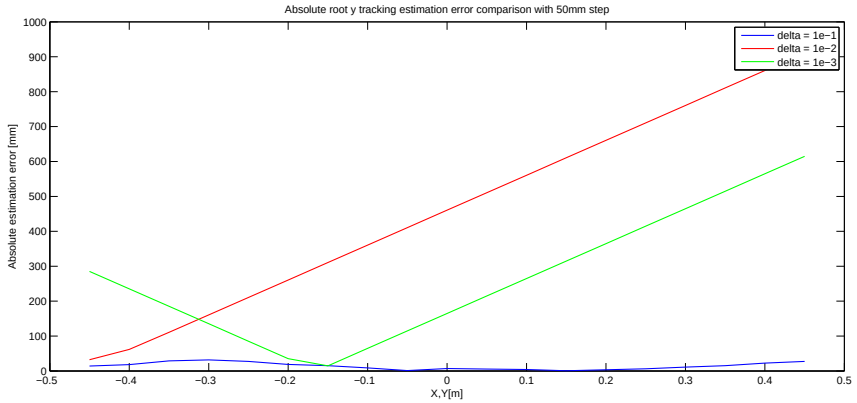


c) Stima di Z

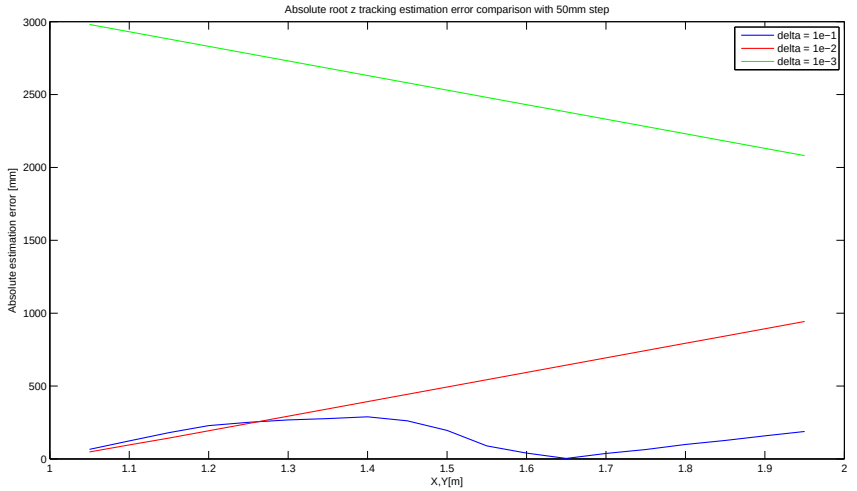
Figura A.57: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 20mm in modalità tracking.



a) Stima di X

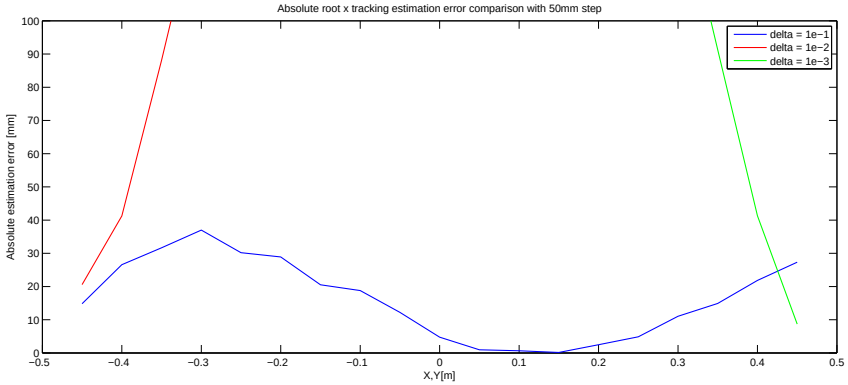


b) Stima di Y

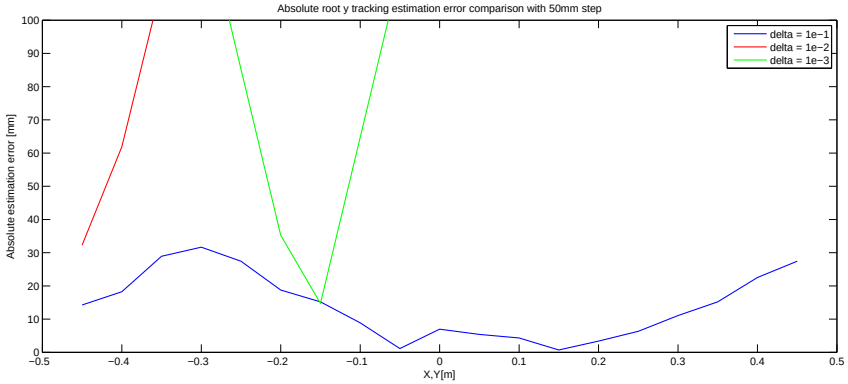


c) Stima di Z

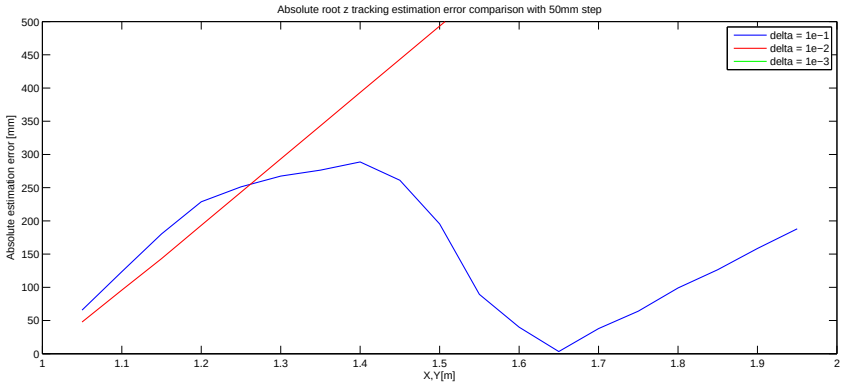
Figura A.58: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 50mm in modalità tracking.



a) Stima di X

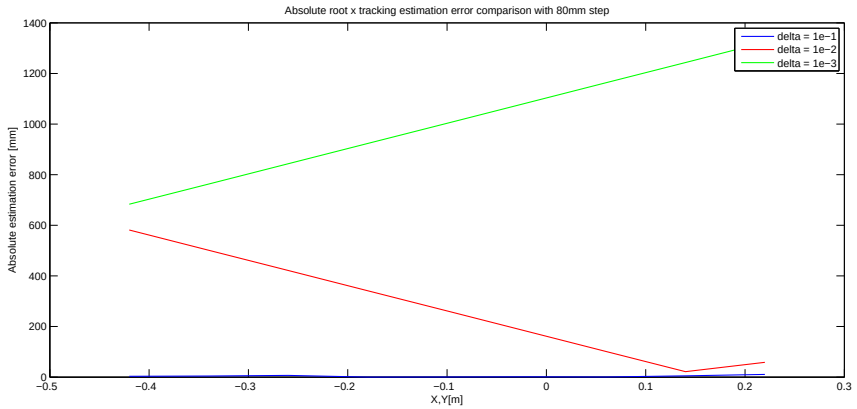


b) Stima di Y

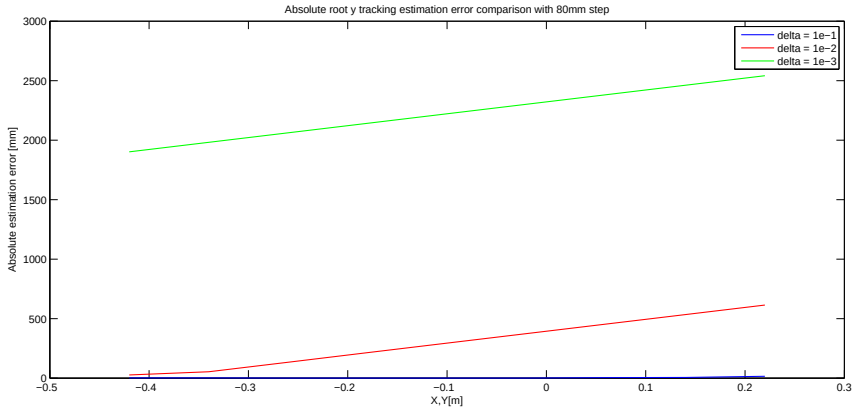


c) Stima di Z

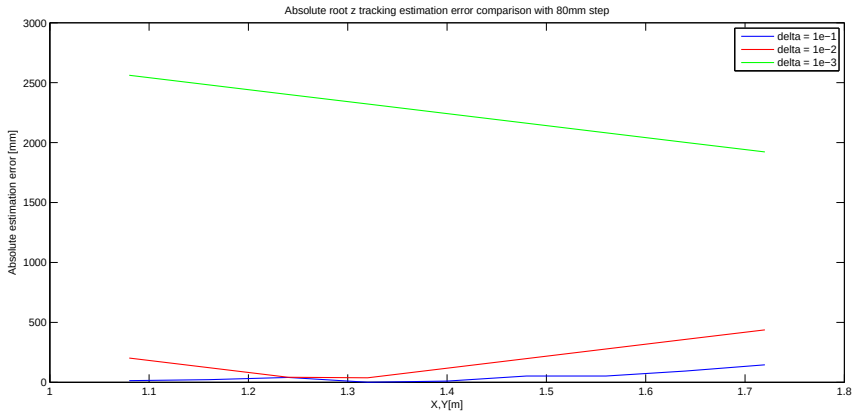
Figura A.59: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 50mm in modalità tracking (ingrandimento).



a) Stima di X

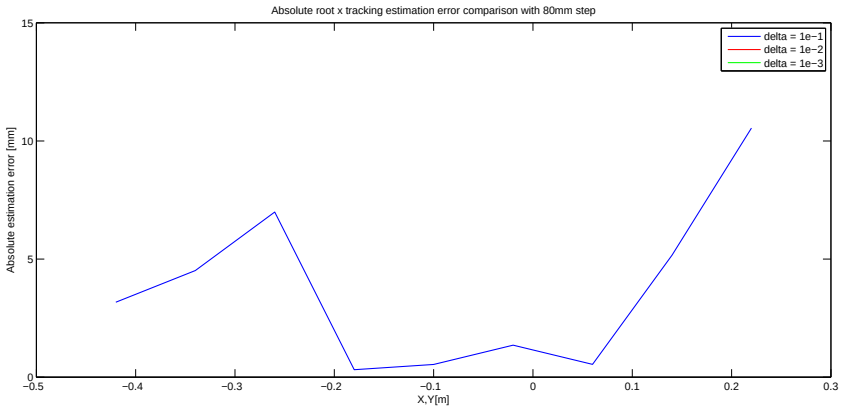


b) Stima di Y

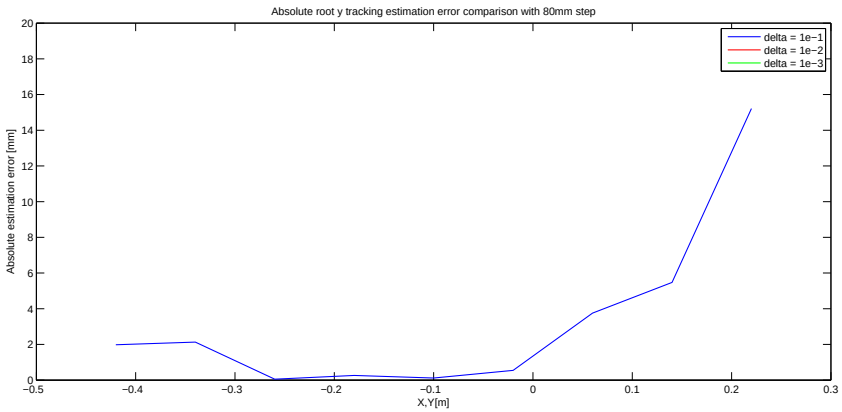


c) Stima di Z

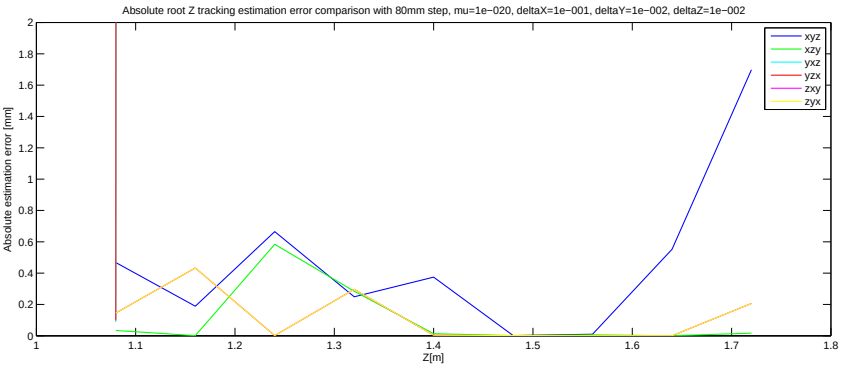
Figura A.60: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 80mm in modalità tracking.



a) Stima di X



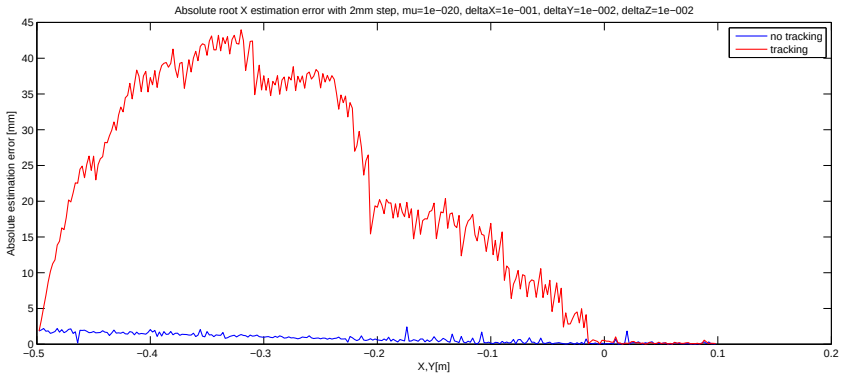
b) Stima di Y



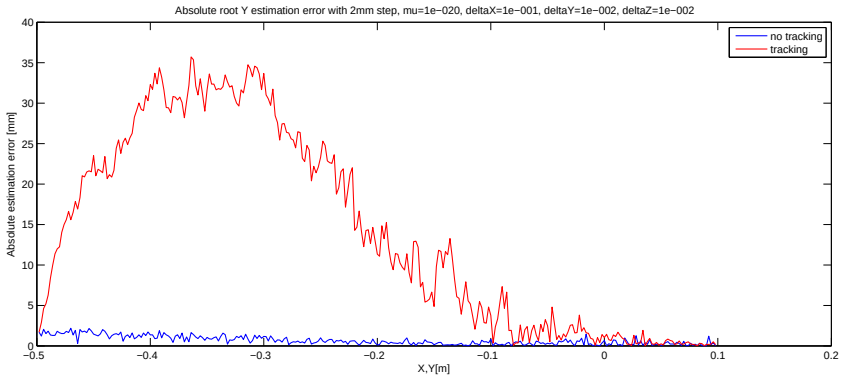
c) Stima di Z

Figura A.61: Confronto errore assoluto di stima di X, di Y e di Z per tre diversi valori di Δ e spostamenti di 80mm in modalità tracking (ingrandimento).

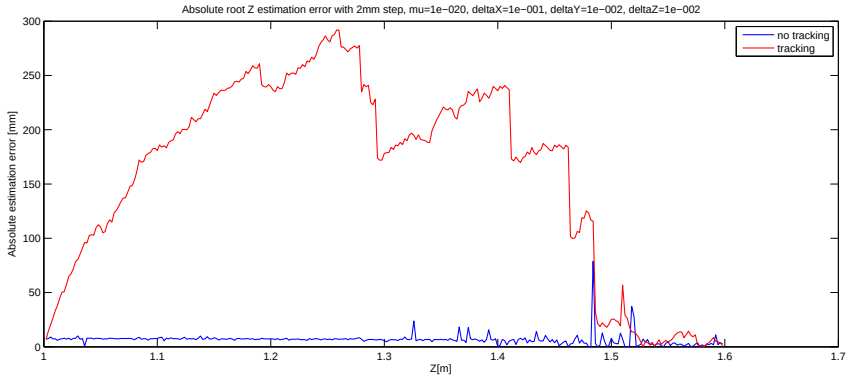
A.9 Grafici per il Test 9



a) Stima di X

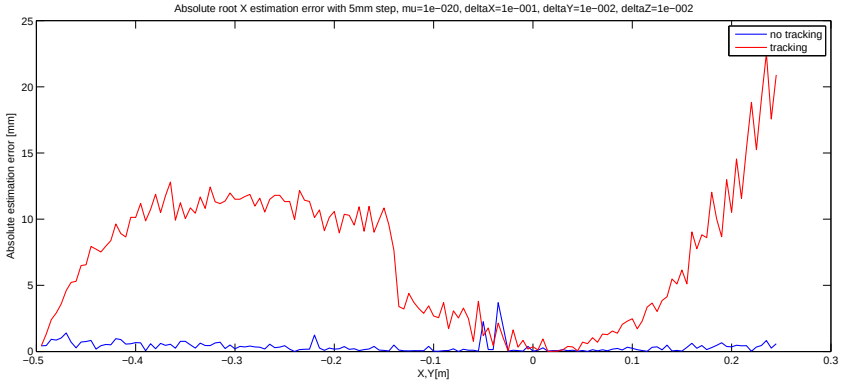


b) Stima di Y

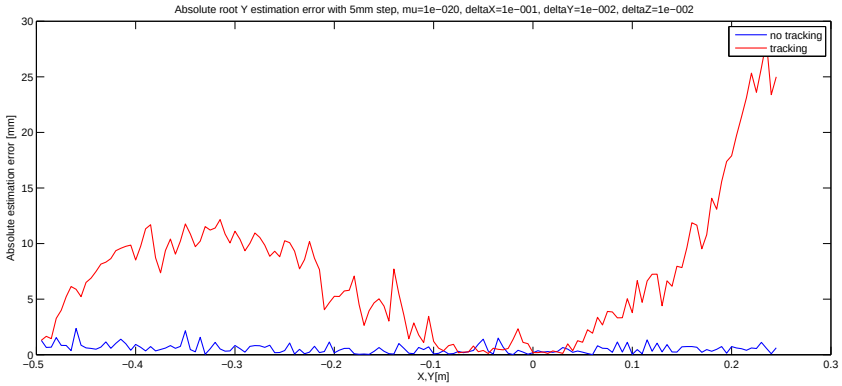


c) Stima di Z

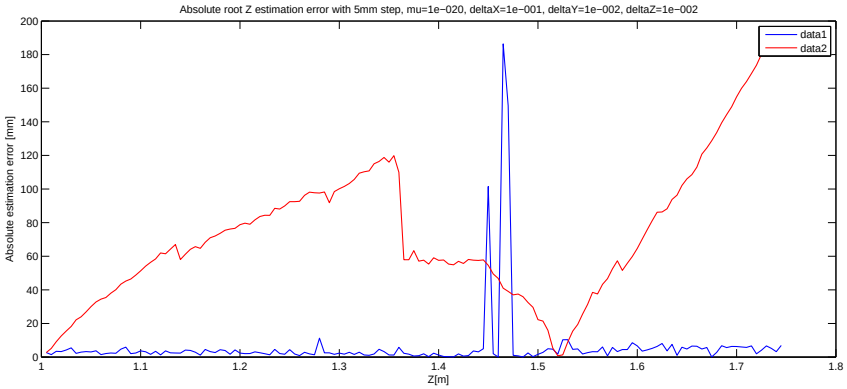
Figura A.62: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 2mm.



a) Stima di X

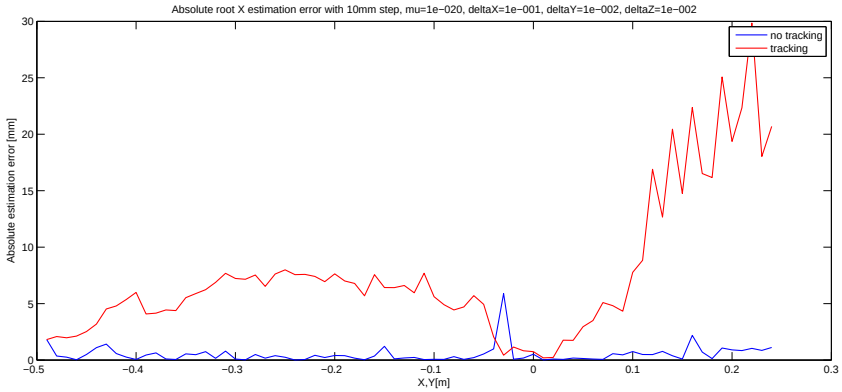


b) Stima di Y

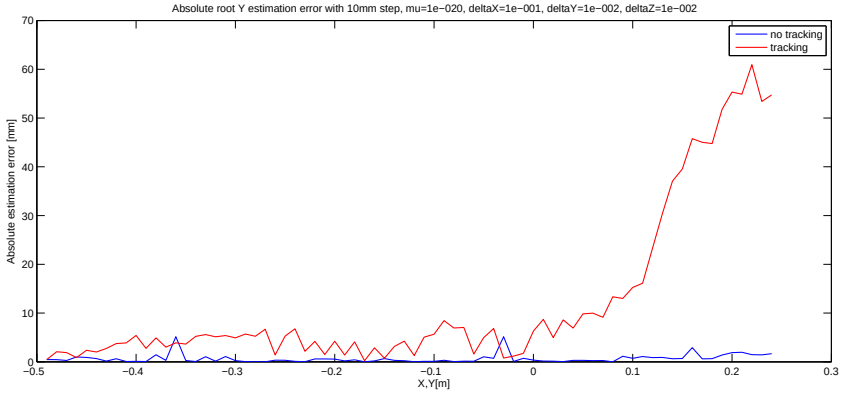


c) Stima di Z

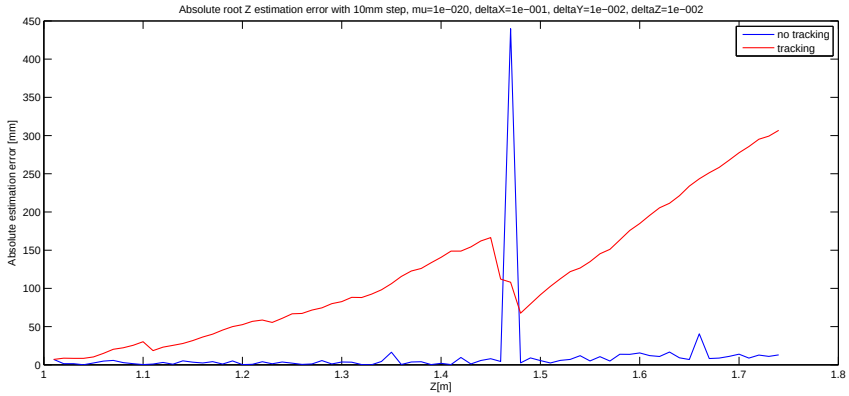
Figura A.63: Confronto errore assoluto di stima di X, Y e Z per spostamenti di 5mm.



a) Stima di X

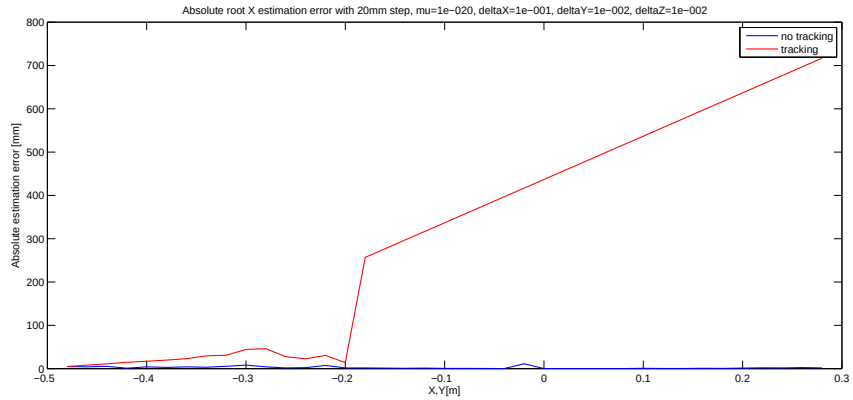


b) Stima di Y

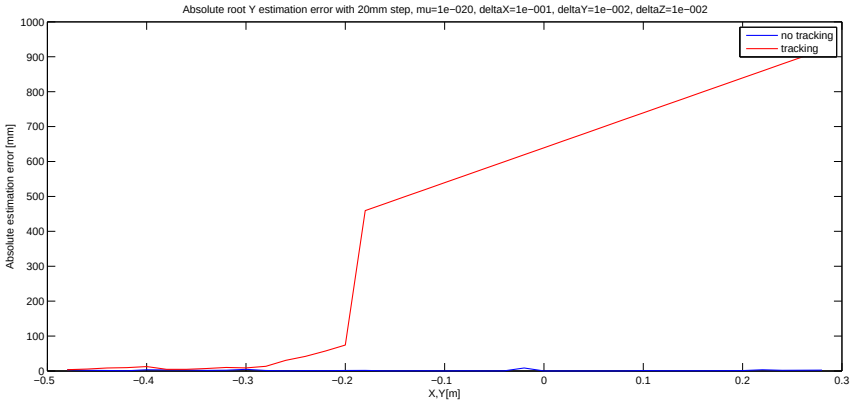


c) Stima di Z

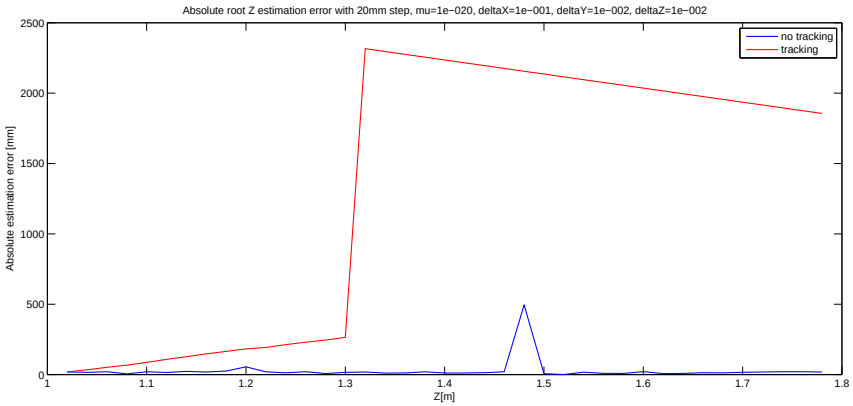
Figura A.64: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 10mm.



a) Stima di X

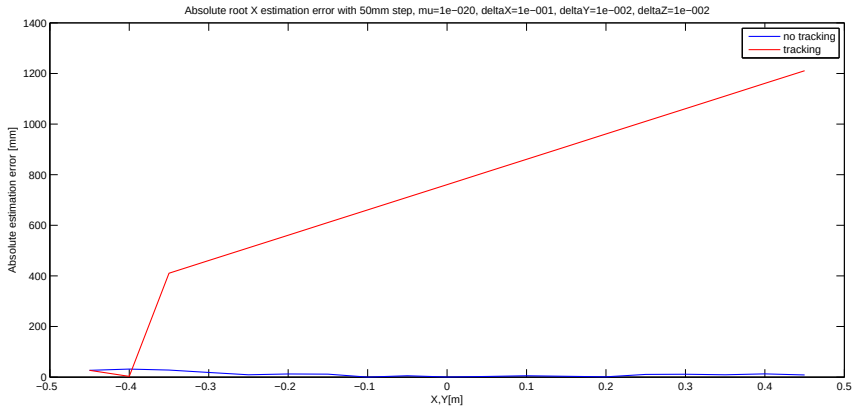


b) Stima di Y

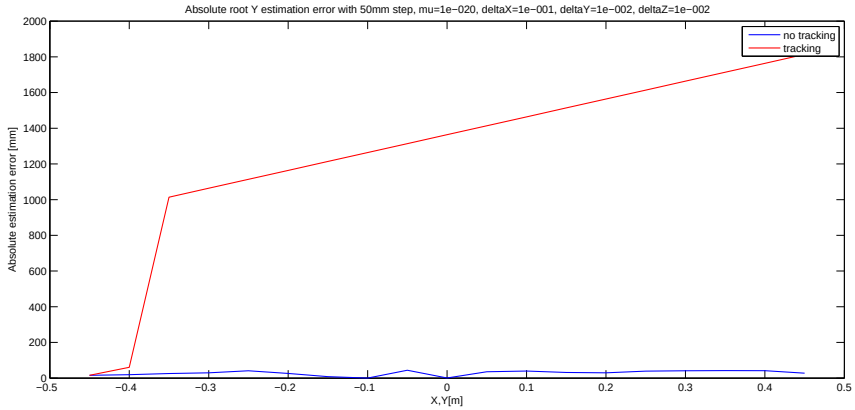


c) Stima di Z

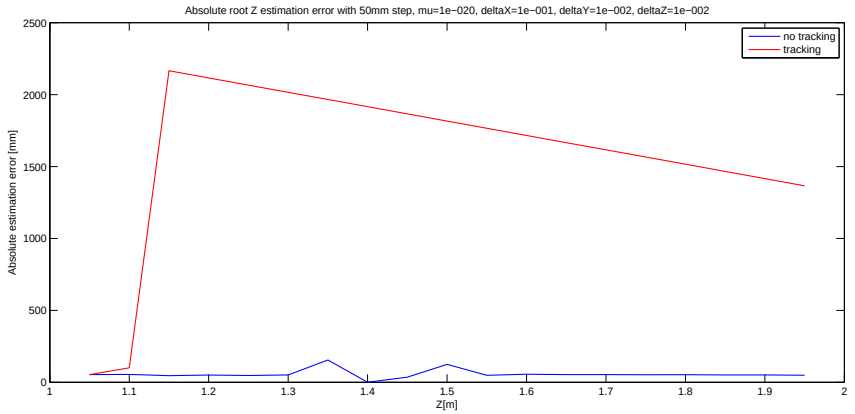
Figura A.65: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 20mm.



a) Stima di X

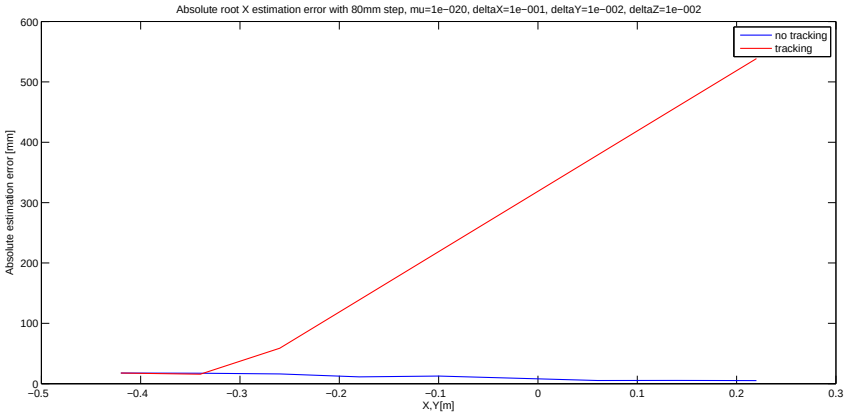


b) Stima di Y

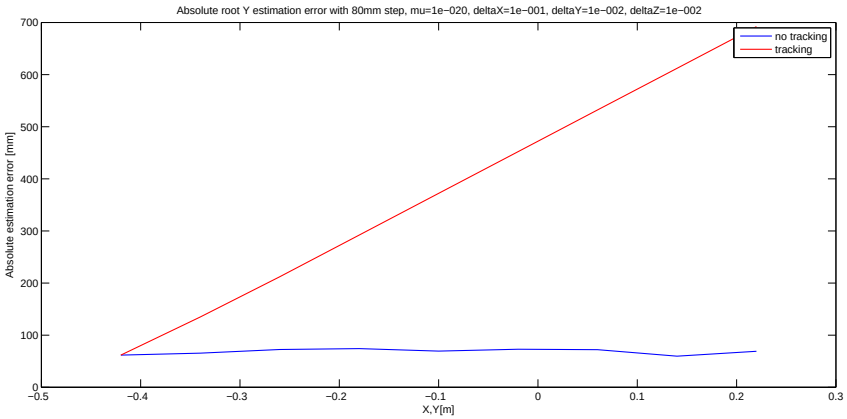


c) Stima di Z

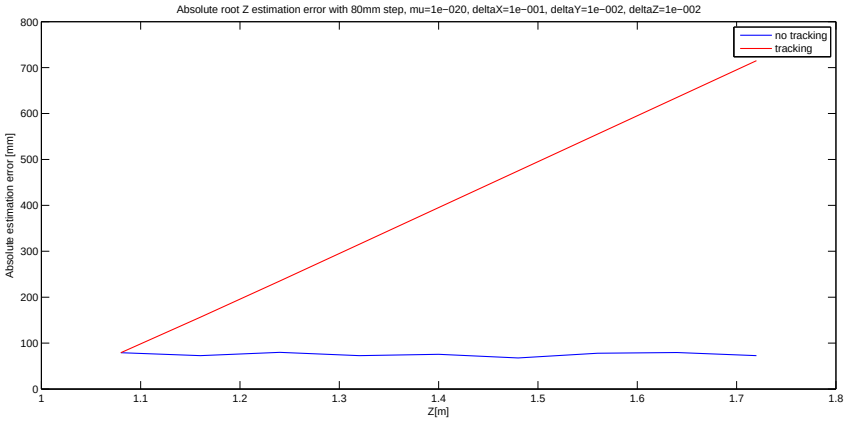
Figura A.66: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 50mm.



a) Stima di X



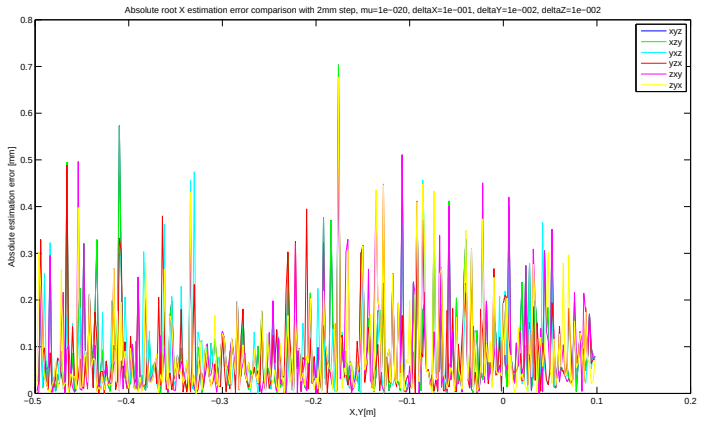
b) Stima di Y



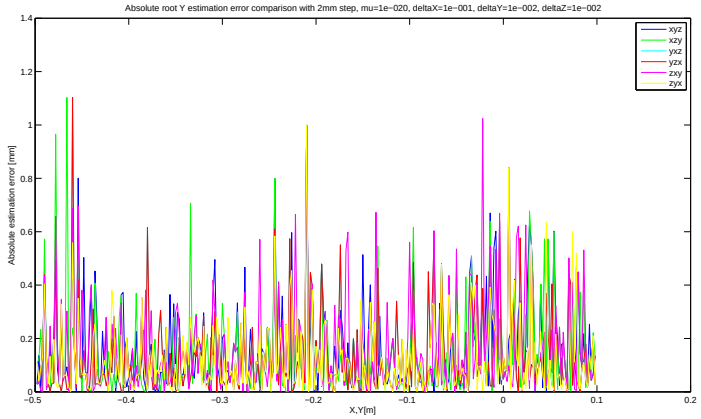
c) Stima di Z

Figura A.67: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 80mm.

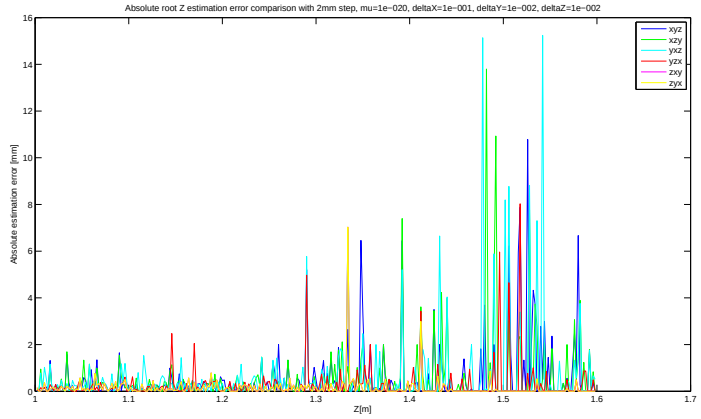
A.10 Grafici per il Test 10



a) Stima di X

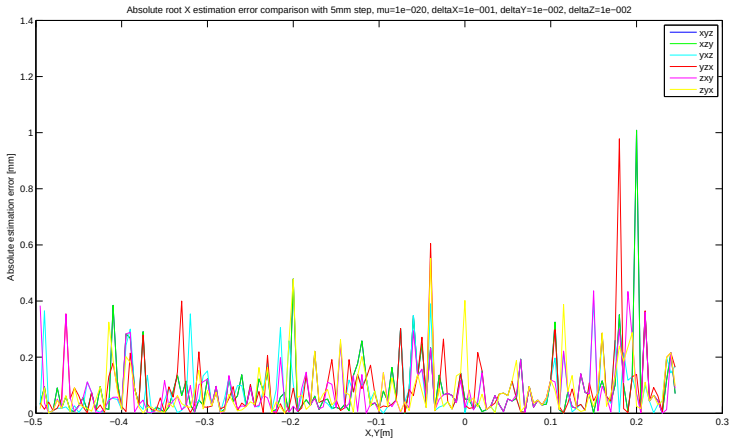


b) Stima di Y

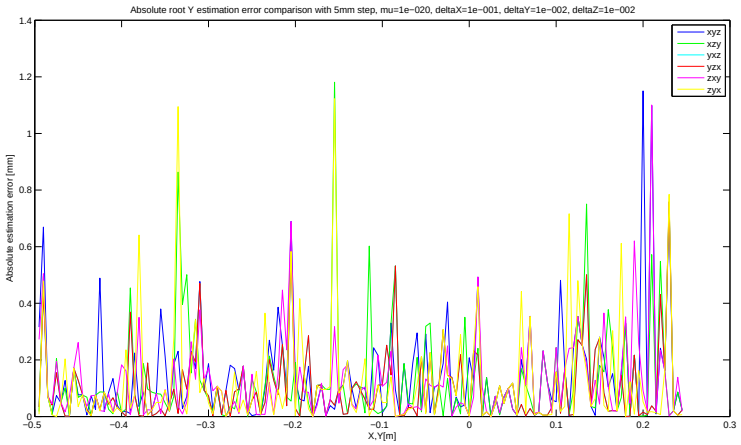


c) Stima di Z

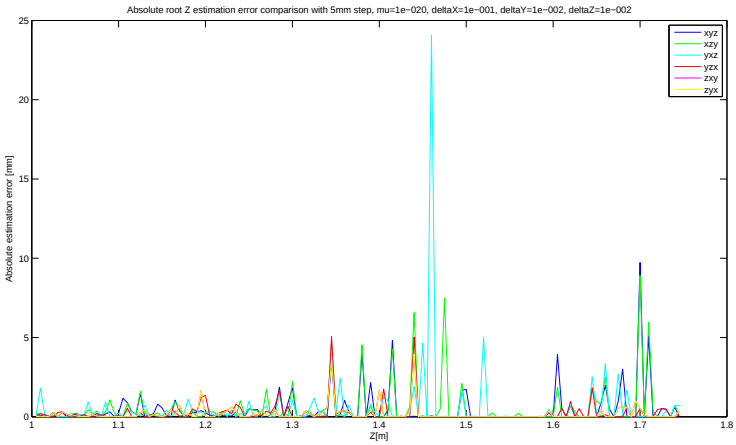
Figura A.68: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 2mm.



a) Stima di X

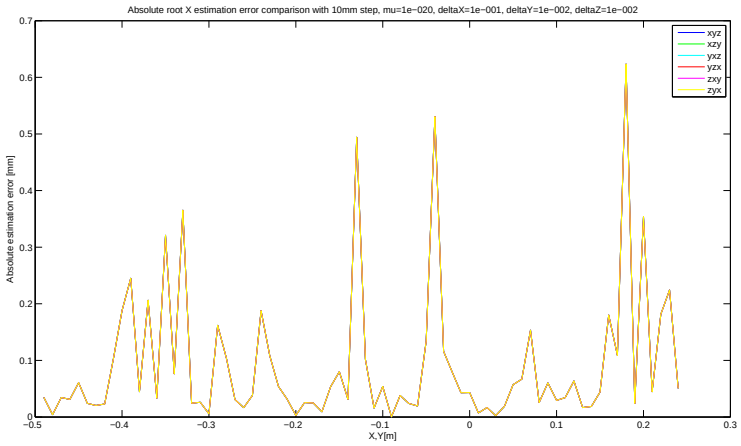


b) Stima di Y

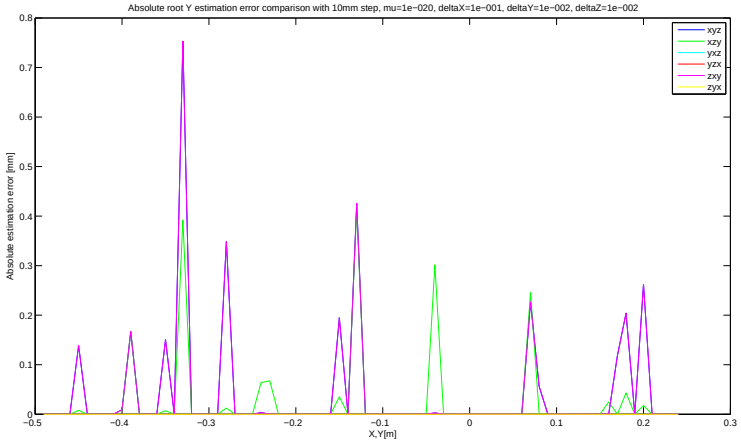


c) Stima di Z

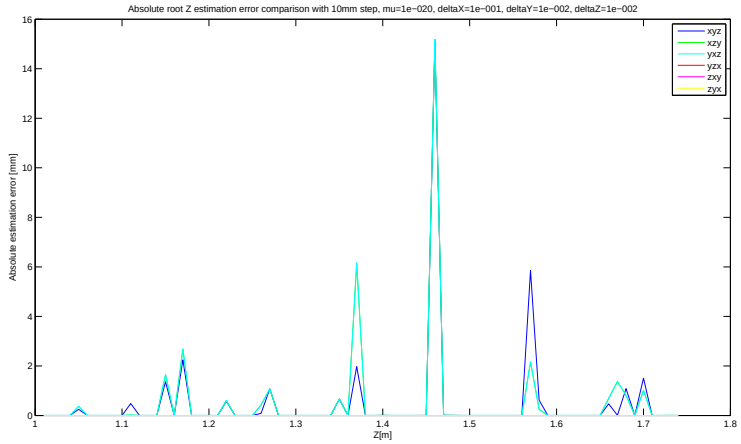
Figura A.69: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 5mm.



a) Stima di X

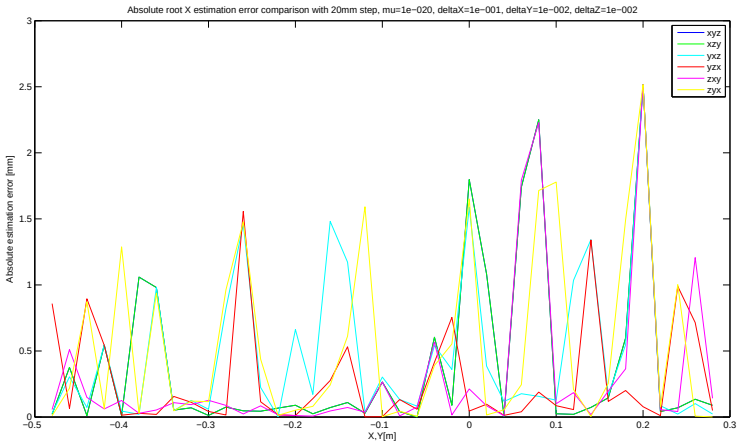


b) Stima di Y

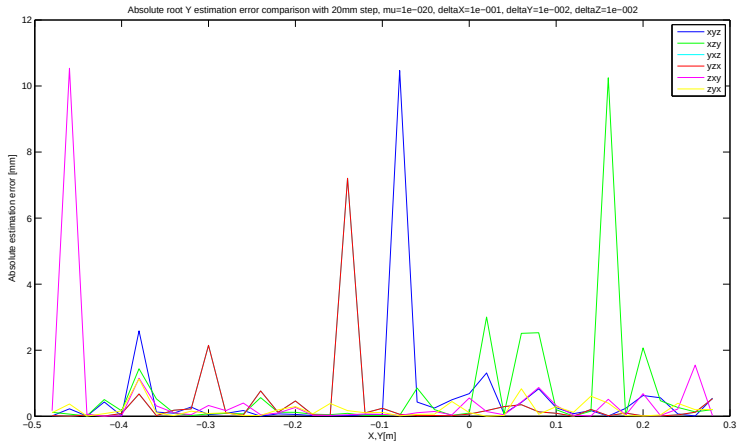


c) Stima di Z

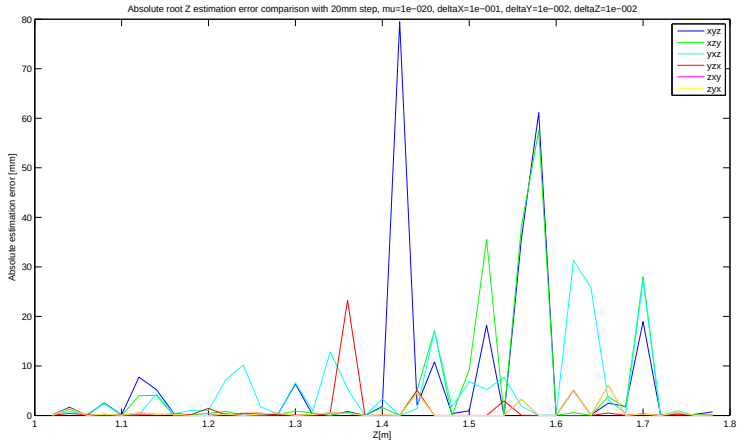
Figura A.70: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 10mm.



a) Stima di X

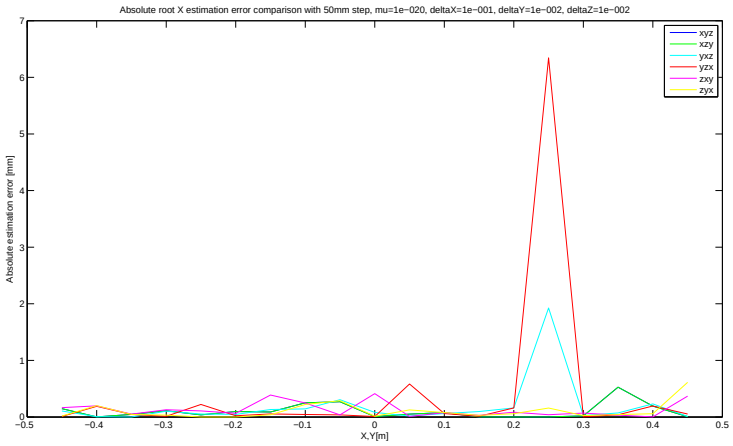


b) Stima di Y

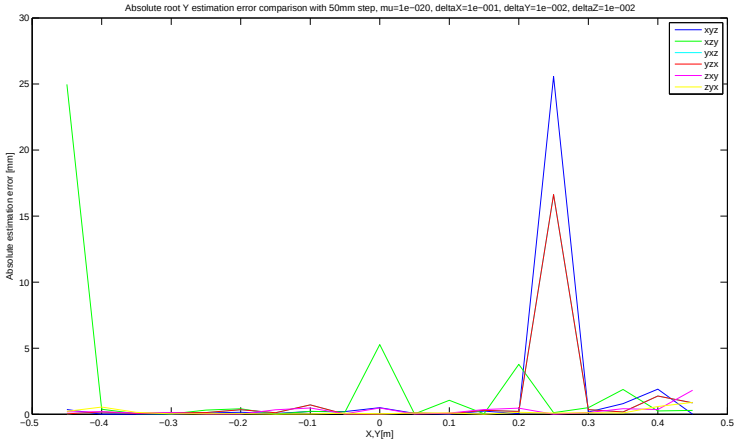


c) Stima di Z

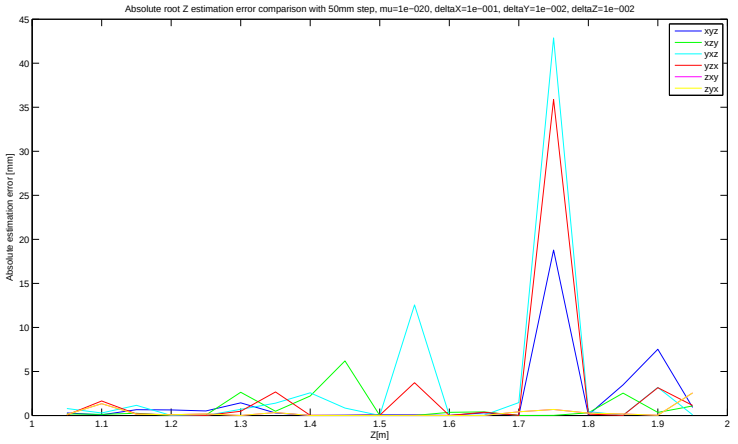
Figura A.71: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 20mm.



a) Stima di X

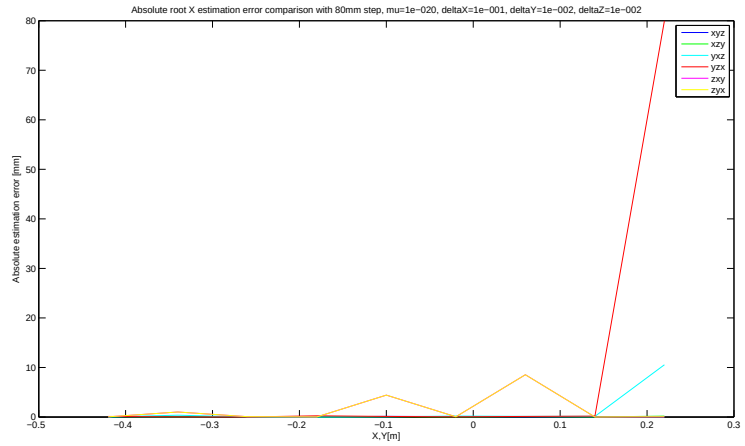


b) Stima di Y

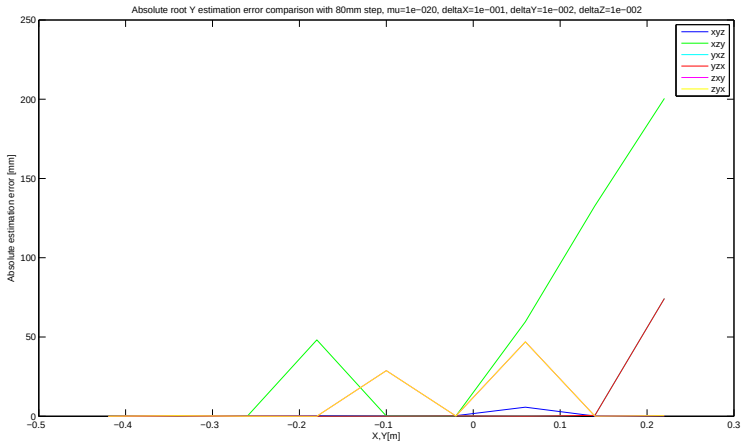


c) Stima di Z

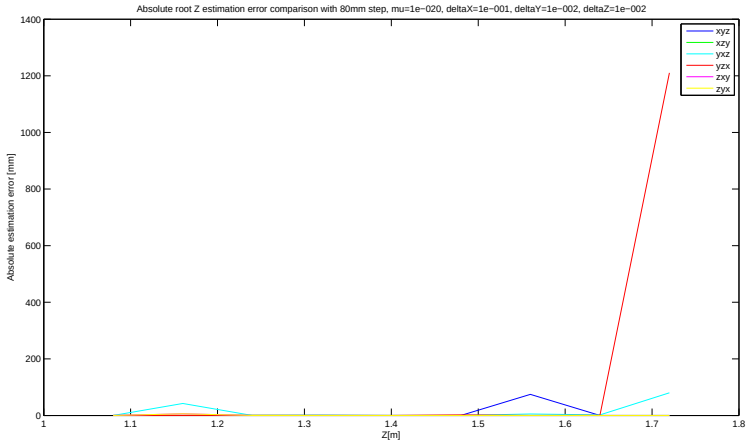
Figura A.72: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 50mm.



a) Stima di X

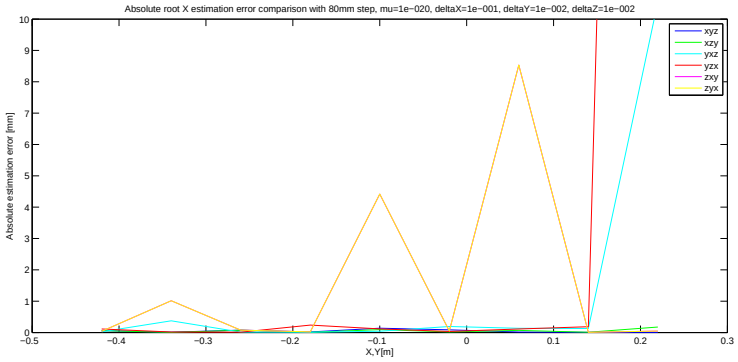


b) Stima di Y

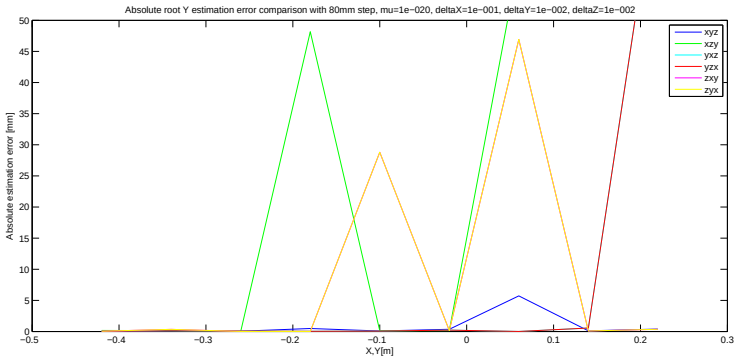


c) Stima di Z

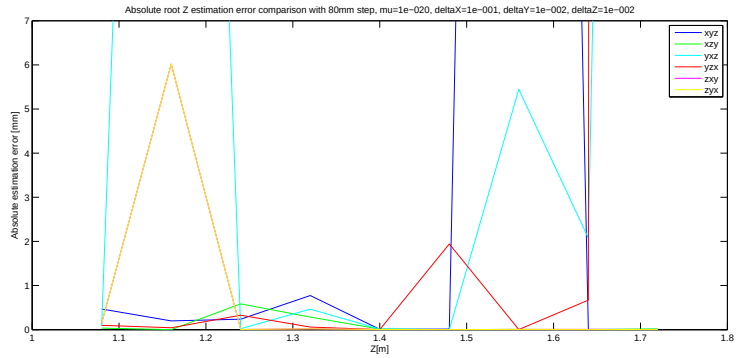
Figura A.73: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 80mm.



a) Stima di X



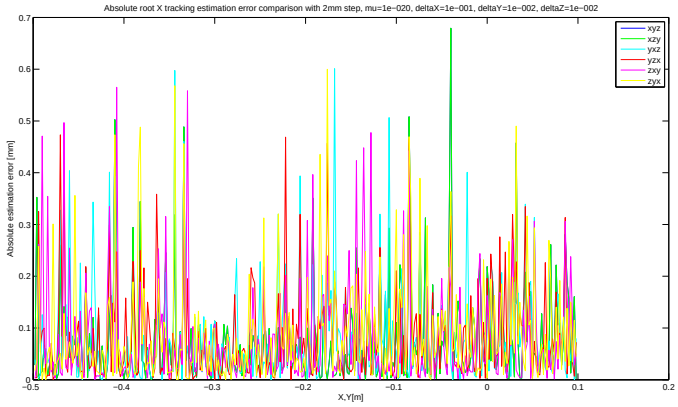
b) Stima di Y



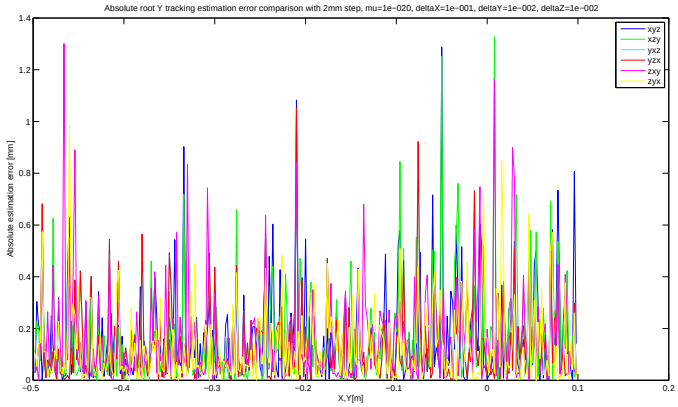
c) Stima di Z

Figura A.74: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 80mm (ingrandimento).

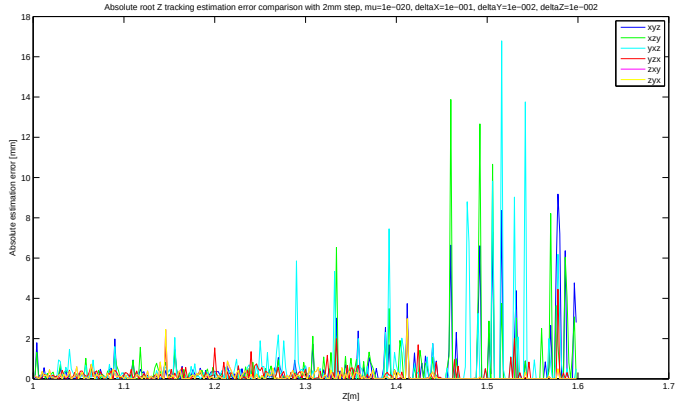
A.11 Grafici per il Test 11



a) Stima di X

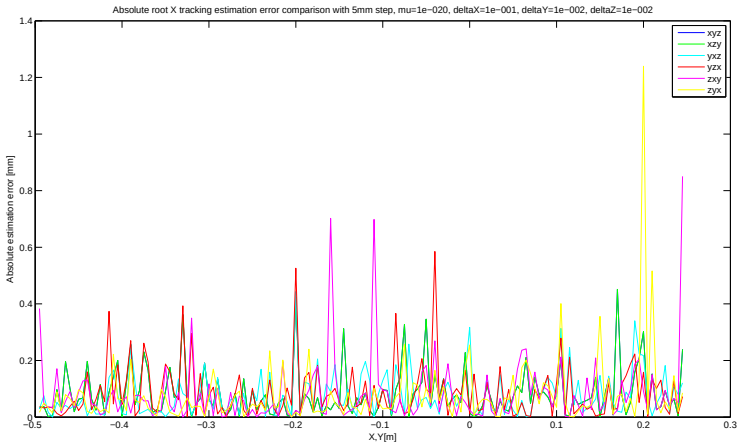


b) Stima di Y

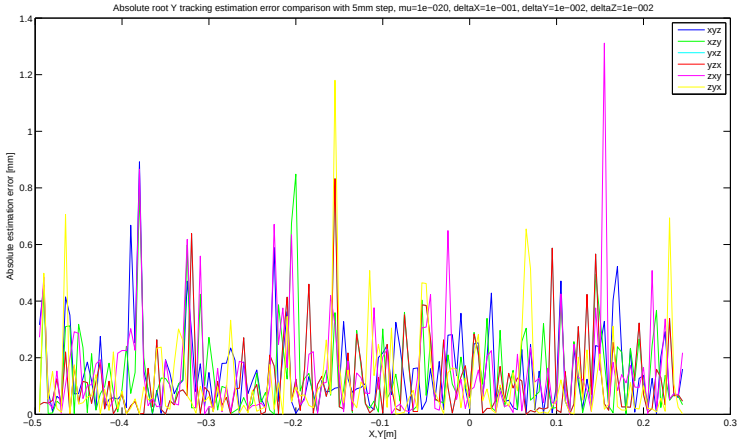


c) Stima di Z

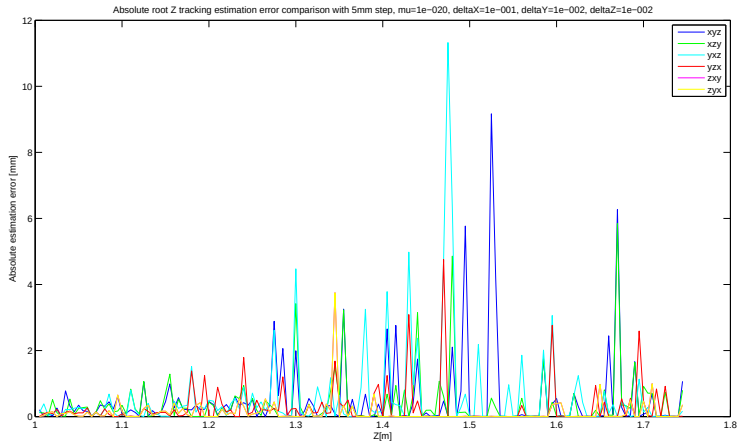
Figura A.75: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 2mm in modalità tracking.



a) Stima di X

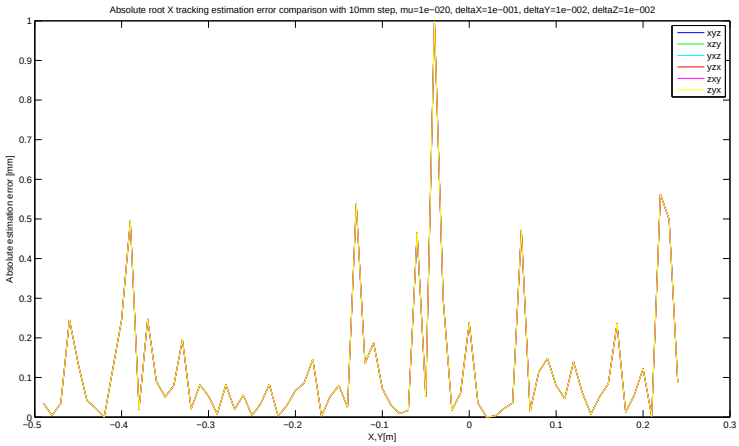


b) Stima di Y

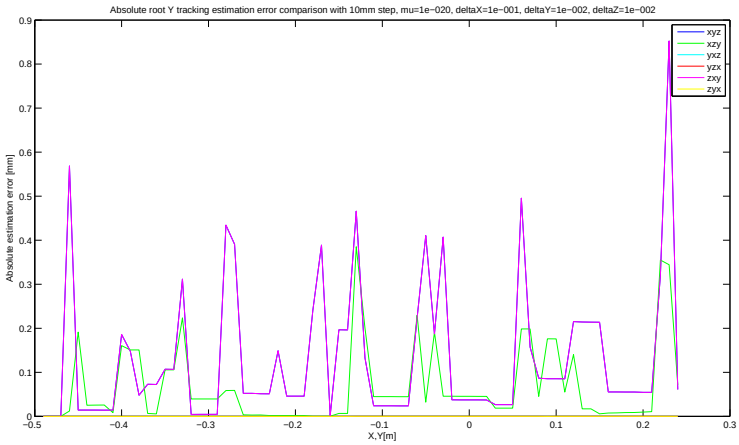


c) Stima di Z

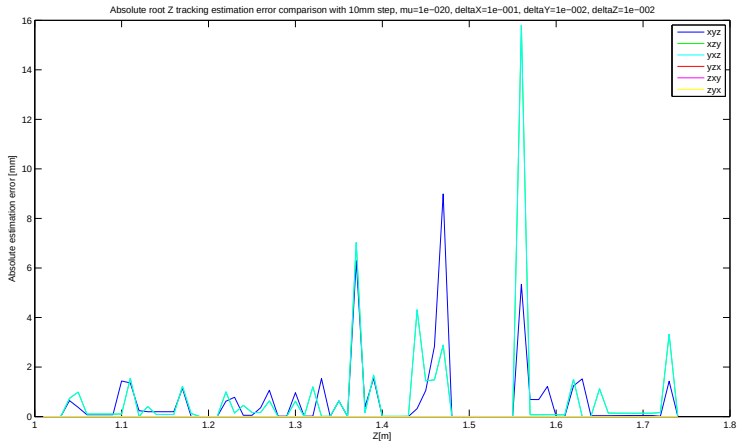
Figura A.76: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 5mm in modalità tracking.



a) Stima di X

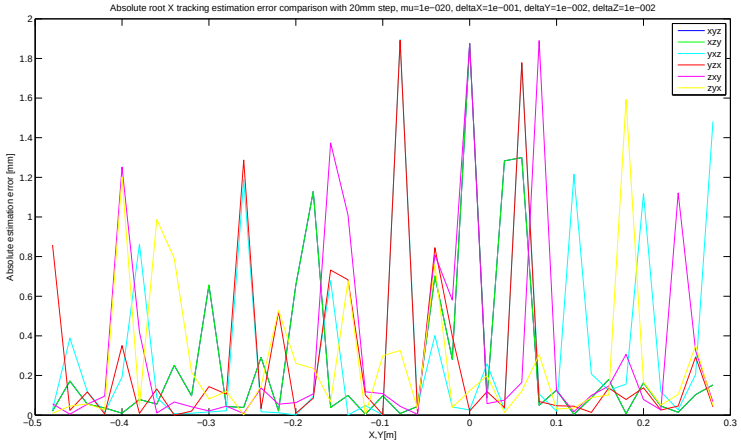


b) Stima di Y

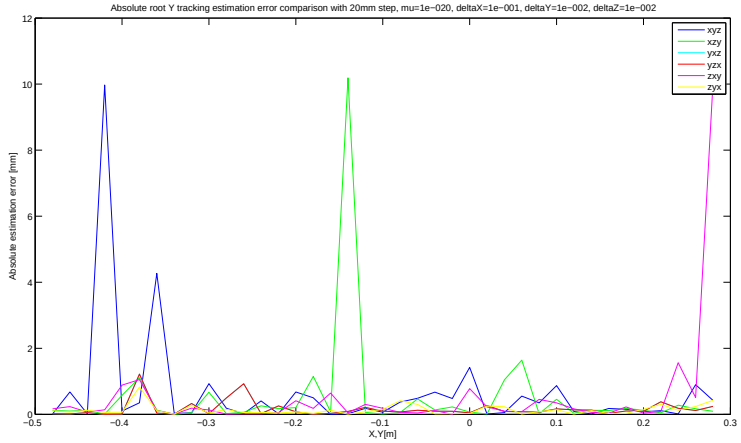


c) Stima di Z

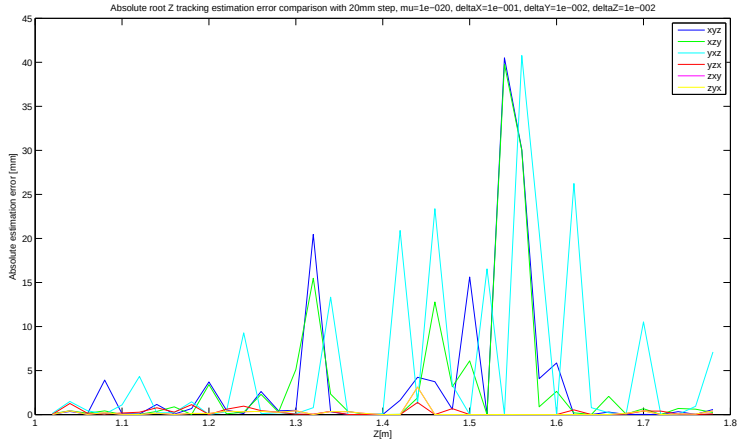
Figura A.77: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 10mm in modalità tracking.



a) Stima di X

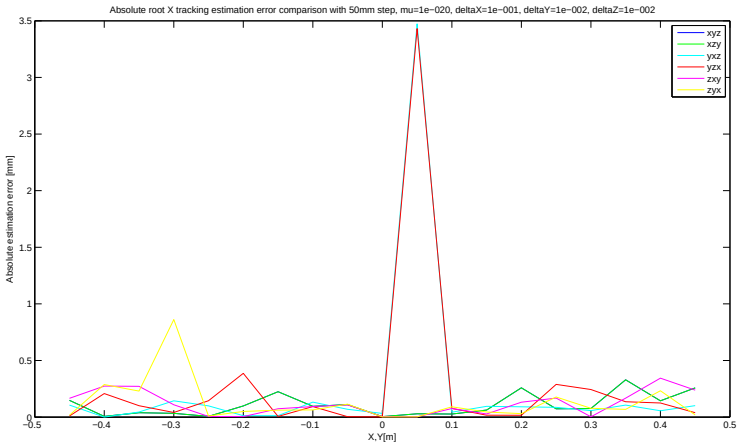


b) Stima di Y

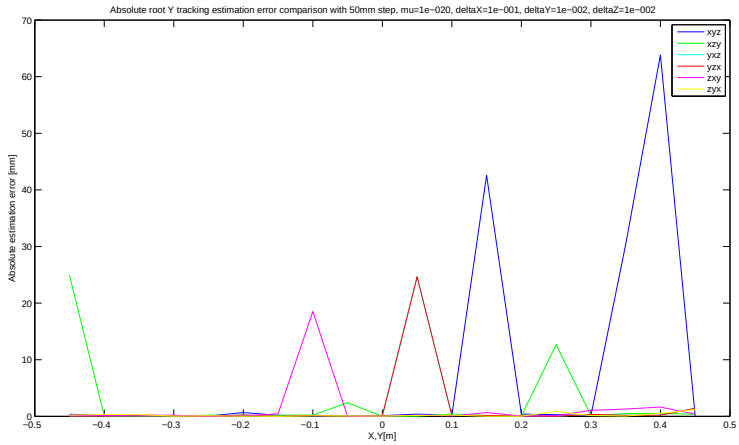


c) Stima di Z

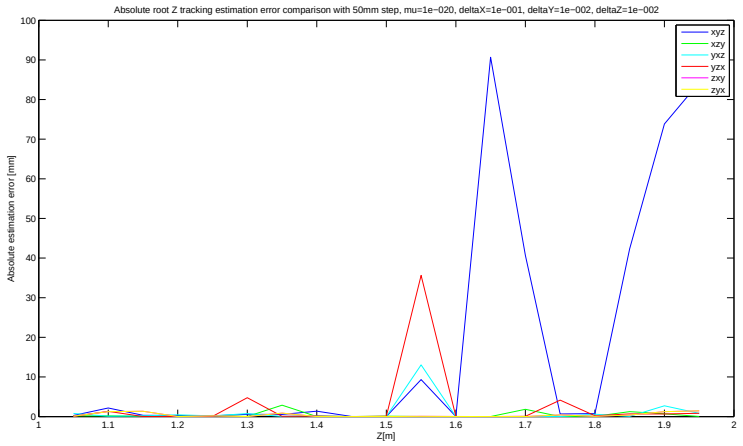
Figura A.78: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 20mm in modalità tracking.



a) Stima di X

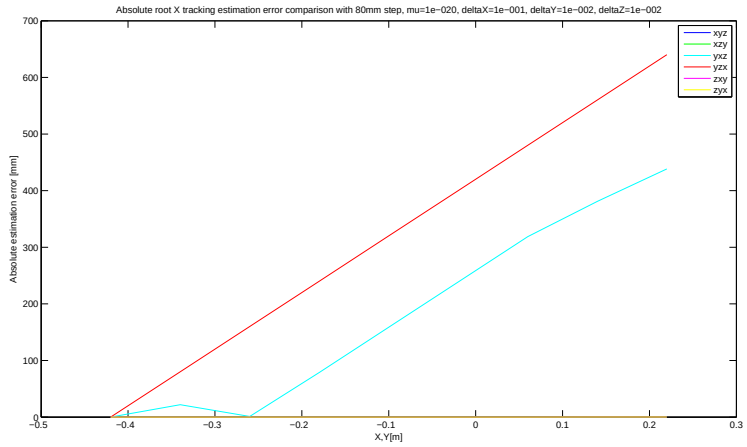


b) Stima di Y

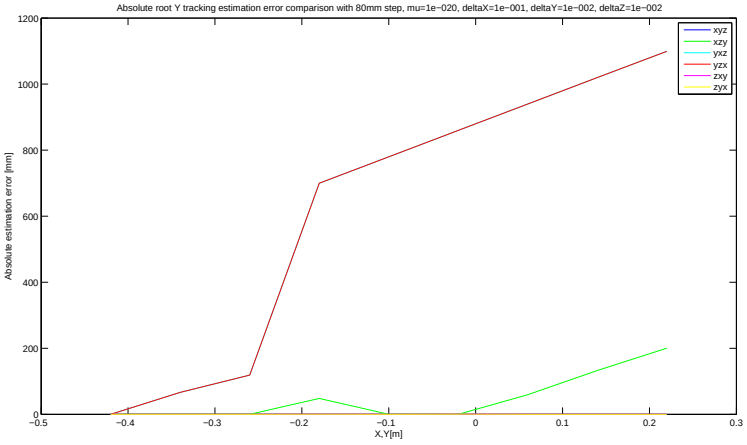


c) Stima di Z

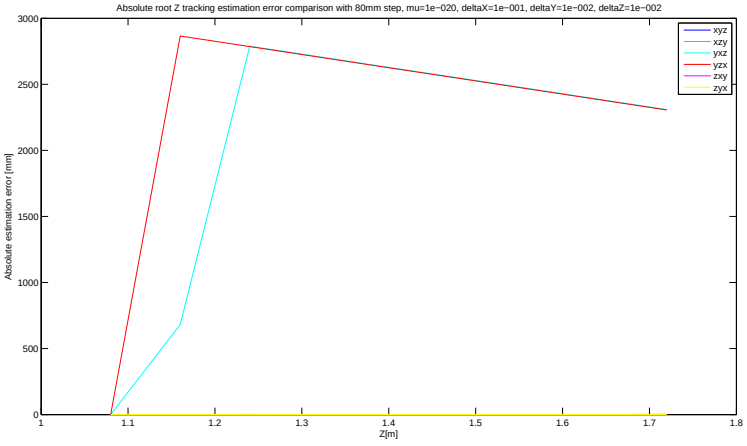
Figura A.79: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 50mm in modalità tracking.



a) Stima di X

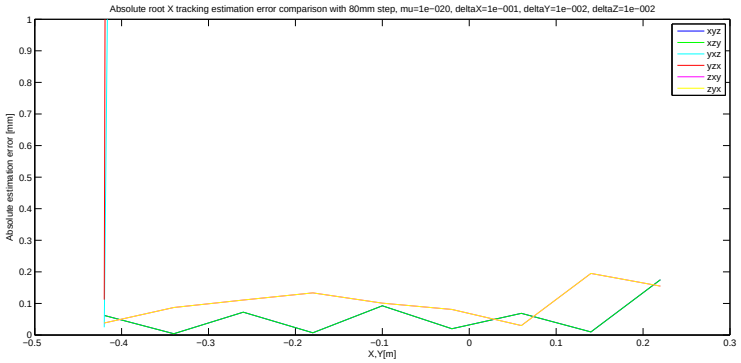


b) Stima di Y

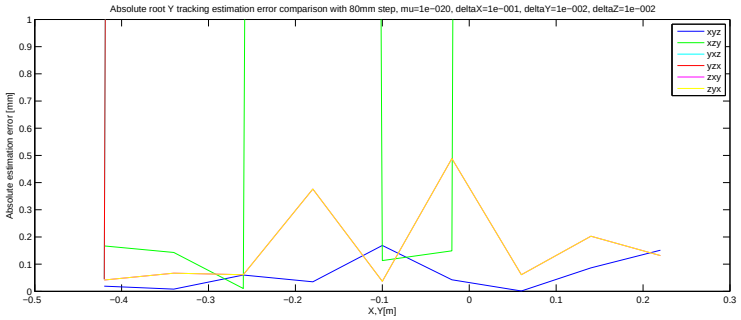


c) Stima di Z

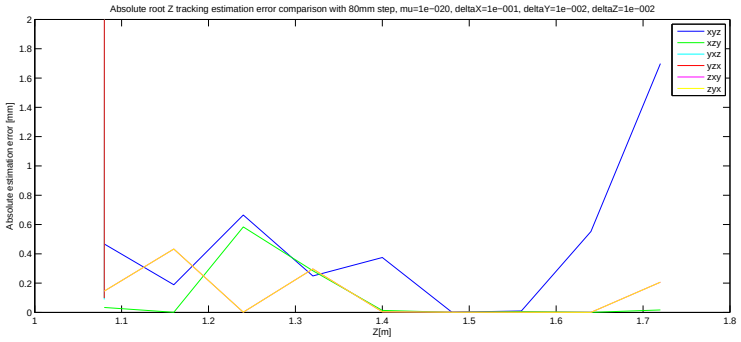
Figura A.80: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 80mm in modalità tracking.



a) Stima di X



b) Stima di Y



c) Stima di Z

Figura A.81: Confronto errore assoluto di stima di X, di Y e di Z per spostamenti di 80mm in modalità tracking (ingrandimento).

Bibliografia

[1] Broyden's method. = http://www.iue.tuwien.ac.at/phd/khalil/node14.html. 62

[2] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammerling, J. Demmel, C. Bischof, and D. Sorensen. Lapack: a portable linear algebra library for high-performance computers. In *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, Supercomputing '90, pages 2–11, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press. 61

[3] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. In *Proceedings of the SIGGRAPH Conference*, 2005. 3, 21

[4] Luca Ballan. *Acquiring shape and motion of interacting people from videos.* Gennaio 2009. 47, 171

[5] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26, July 2007. 44, 47

[6] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14:239–256, February 1992. 24

[7] Gary Bradski and Adrian Kaehler. *Learning OpenCV.* O'Reilly Media Inc., 2008. 11

[8] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593, March 1965. 62

[9] J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations (Classics in Applied Mathematics, 16).* Soc for Industrial & Applied Math, 1996. 60

[10] Andrea Fusiello. *Visione computazionale. Appunti delle lezioni.* Giugno 2008. 11, 53

[11] Andrea Fusiello, Umberto Castellani, Luca Ronchetti, and Vittorio Murino. Model acquisition by registration of multiple acoustic range views. In *Computer Vision - ECCV 2002, LNCS*, pages 805–819. Springer, 2002. 30

[12] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. 12

[13] Paul S. Heckbert and Michael Garland. Optimal triangulation and quadric-based surface simplification. *Journal of Computational Geometry: Theory and Applications*, 14:49–65, 1999. 35, 36, 164

[14] David Jacka, Ashley Reid, and Bruce Merry. A comparison of linear skinning techniques for character animation. In *In Afrigraph*, pages 177–186. ACM, 2007. 46

[15] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. 33, 34, 164

[16] Kitware. Vtk. = http://www.vtk.org/. 54

[17] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for fortran usage. *ACM Trans. Math. Softw.*, 5:308–323, September 1979. 61

[18] Peter Liepa. Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '03, pages 200–205, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. 33

[19] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21:163–169, August 1987. 33

[20] M.I.A. Lourakis. levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++. =http://www.ics.forth.gr/ lourakis/levmar/. 61

[21] Giulio Marin. *Modelli deformabili della mano a partire da dati acquisiti tramite laser scanner*. Tesi di laurea edition, 2011. 23

[22] Steve Marschner, James Davis, Matt Garr, and Marc Levoy. Filling holes in complex surfaces using volumetric diffusion, 2001. 33

[23] MESA. Sr4000. = http://www.mesa-imaging.ch. 5

[24] MESA. Sr4000 manual. = http://www.mesa-imaging.ch. 6, 7

[25] C. Dal Mutto, P. Zanuttigh, and G.M. Cortelazzo. A probabilistic approach to tof and stereo data fusion. In *Proceedings of 3DPVT 10*, Paris (France), May 2010. 13

[26] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer series in operations research. Springer, 1999. 60

[27] L. Sigal, A. O. Balan, and M. J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87:4–27, March 2010. 3

Elenco delle figure

Elenco delle tabelle

Lista delle Formule

Sono passati ormai due anni dalla laurea triennale, ma sento ancora la trepidazione di quel giorno come fosse ieri. Il tempo è volato, e sono giunto al termine di un nuovo ciclo di studi quasi senza accorgermene.

Le persone a cui devo dei sentiti ringraziamenti sono molte, e queste poche righe non sono sufficienti per citarle tutte. Vorrei ringraziare innanzitutto il prof. Cortelazzo per la sua disponibilità anche nei periodi di meritato riposo e per la fiducia che ha riposto in me, una fiducia di cui spesso sono privo. Ringrazio il prof. Zanuttigh e l'ing. Dal Mutto per avermi guidato nell'arduo lavoro di tesi e per i loro preziosi consigli che, più di una volta, mi hanno aiutato a uscire da situazioni di stallo. Un grazie sentito anche a Giulio Marin per avermi assistito nella prima parte del progetto, a tutti i volontari che si sono prestati ai miei tediosi esperimenti e ai compagni tesisti del laboratorio LTTM che mi hanno fatto sentire parte di una grande famiglia. Ringrazio anche i docenti che mi hanno accompagnato attraverso questo lungo e tortuoso percorso di studi e i compagni di corso con cui ho condiviso indimenticabili momenti di vita universitaria.

Ringrazio i miei genitori che mi hanno sostenuto sia economicamente che moralmente sin dal primo giorno di università, i miei familiari, gli amici e i conoscenti che hanno creduto in me e mi hanno esortato a proseguire gli studi dando sempre il meglio. Un grazie particolare ai parenti venuti da lontano per partecipare alla mia laurea, e a quelli che sarebbero voluti venire ma non hanno potuto per motivi di forza maggiore. Un grazie alle persone che continuano a essermi amiche e a sopportarmi dopo tanti anni, e alle persone che sono state vicino a me e alla mia famiglia durante i forti momenti di dolore di quest'anno. Ringrazio, infine, tutte le persone che non ho citato, ma a cui devo un sentito grazie.