

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

Sviluppo di una applicazione di test per un sistema di localizzazione

PROVA FINALE IN

CONTROLLI AUTOMATICI

RELATORE:

Carlo Rossi

CORRELATORE:

Matteo De Angeli

PRESENTATA DA:

Daniele Mazzotti

SESSIONE III

ANNO ACCADEMICO 2014/2015

Indice

1 Introduzione	5
1.1 Progetto Horvs.....	5
1.2 Ultra Wide Band.....	8
1.3 Classificazione di tecniche di localizzazione.....	9
1.4 Trilaterazione.....	10
1.5 Altre tecniche di localizzazione.....	11
2 Analisi dell'algorithm	15
2.1 Java Native Interface.....	15
2.2 Algoritmo.....	17
2.3 Geometric Dilution of Precision.....	25
2.4 Aggiunte e modifiche.....	28
3 Applicazione di testing	33
3.1 Componenti dell'interfaccia.....	33
3.2 Funzioni di traslazione dei tag.....	45
3.3 Vaadin.....	53
4 Principali funzionalità	57
4.1 Confronto di due algoritmi.....	58
4.2 Analisi di log reali.....	65
5 Rumore e filtri	71
5.1 Filtri.....	76
6 Server testing	81
7 Conclusione	87
Elenco delle figure	91
Glossario	97
Bibliografia	99

1 Introduzione

Argomento di questa tesi sarà un sistema di localizzazione basato sulla comunicazione in Ultra Wide Band (UWB) e sulla tecnica di trilaterazione.

In particolar modo come l'algoritmo è stato implementato e ottimizzato, e lo sviluppo di una applicazione di testing per valutarlo.

1.1 Progetto Horvs

Horvs è il nome in codice di un sistema RTLS (Real Time Localization System), attualmente in sviluppo presso Engynya s.r.l., che tramite la localizzazione di un “tag” può fornire diverse funzioni utili in ambito di sicurezza in ambiente industriale, come il controllo di macchinari in movimento, predizione della traiettoria e prevenzione di collisioni.

La libreria per il calcolo della trilaterazione e le schede elettroniche utilizzate sono basate su codice e componenti DecaWave, azienda irlandese nel campo della localizzazione inDoor^[1].

Il sistema è composto da tre componenti principali: ancore, tag e server.

Ancora:

è una scheda montata in una posizione fissa nell'ambiente in cui è installato il sistema.

Al momento dell'installazione memorizza le proprie coordinate.

Le ancore scambiano messaggi con i tag, per calcolare la distanza da esso, ovvero il range (nel modo che verrà spiegato più avanti).

Si dividono in ancora *Master* e ancore *Slave*.

L'ancora Master è l'unica collegata direttamente al server che gestisce l'intero sistema, il *Central Localization Engine (CLE)*^[2].

Riceve i dati dalle ancore slave e le trasmette al CLE che esegue la trilaterazione.

Il sistema completo è composto da più “celle”, da quattro ancore, tre slave e una master, concatenate tra loro, in modo quindi che quattro celle consistano in nove ancore (di cui solitamente, la master è in comune, quindi quella al centro).

Ogni cella localizza un tag al suo interno, permettendo così di coprire una area maggiore.

Tag:

è il dispositivo che viene localizzato. Comunica con le ancore per il ranging e può ricevere delle istruzioni, come ad esempio far suonare un buzzer quando è in prossimità di un altro tag.

Il sistema può gestire un gran numero di tag, che vengono interrogati a rotazione. Il tempo tra due interrogazioni dipende quindi dal numero di tag presenti.

Dopo uno scambio di messaggi con le ancore il tag entra in *sleep mode*, per una finestra temporale definita, in cui vengono interrogati gli altri tag.

Quando il tag si risveglia, manda un messaggio di poll alle ancore per eseguire il ranging.

Periodicamente vengono eseguite delle comunicazioni per sincronizzare i tag, che aggiustano il loro tempo di sleep per rientrare nella giusta finestra temporale.

Server:

Il CLE, ovvero *Central Localization Engine*, è la macchina su cui gira l'applicazione.

Riceve i dati di ranging dalle ancore master e li usa per eseguire la trilaterazione, calcolando la posizione del tag, comunicando poi alle ancore come agire.

è collegato a una interfaccia utente per visualizzare graficamente le posizioni dei tag.



Immagine 1.1.1 : Prototipo di ancora. (Modello 3D)

Le comunicazioni sono eseguite tramite UltraWideBand.

1.2 Ultra Wide Band

L'Ultra Wide Band (UWB) è una tecnologia di trasmissione dei segnali in radiofrequenza, progettata per lo scambio di dati su breve distanza (decine di metri), caratterizzata da una banda passante molto ampia^[3].

L'UWB opera nelle frequenze da 3 a 10 GHz, con canali di 500MHz.

Le comunicazioni in UWB sono a bassissima potenza (decimi di milliwatt), e sfruttano una serie di impulsi molto rapidi per lo scambio di informazioni. Queste caratteristiche permettono di evitare interferenza con altri tipi di segnali a potenza più elevata; inoltre la trasmissione a impulsi permette di evitare le cosiddette interferenze autoprodotte, ovvero derivanti dai segnali di rimbalzo, problema che affligge comunicazioni molto lunghe e continue, in cui la prima parte del segnale, rimbalzando, interferisce con il resto della comunicazione. Cosa che non avviene in questo caso in quanto gli impulsi vengono ricevuti completamente prima che possano soffrire di questa interferenza.

Lavorando anche ad alte frequenze aumenta la capacità del segnale UWB di passare attraverso oggetti solidi, limitando però di molto la portata della comunicazione, che rimane limitata a poche decine di metri.

Tra le principali applicazioni dell'UWB troviamo^[4]:

- **PAN** : Grazie all'ampia banda, permette la trasmissione di grandi moli di dati, per cui viene utilizzata per le comunicazioni su scala ridotta, quindi in Personal Area Network (PAN) wireless.
- **Periferiche informatiche** : Grazie alla bassa potenza e quindi basso consumo, e bassa emissione di onde radio, potrebbe essere utilizzata la tecnologia UWB per la comunicazione tra periferiche, similmente al bluetooth.
- **Localizzazione** : Le comunicazioni UWB permettono una grande precisione nel localizzare un oggetto, ricevente o trasmittente, con errori anche di pochi centimetri; inoltre la capacità di attraversare senza troppe interferenze oggetti solidi permette la localizzazione in-door, ovvero all'interno di spazi chiusi, che possono anche essere divisi da muri.

1.3 Classificazione di tecniche di localizzazione

Per trilaterazione si intende un metodo che permette di calcolare la posizione di un punto sfruttando le proprietà dei triangoli, in particolar modo la proprietà di indeformabilità, per cui un triangolo può essere definito conoscendo i suoi lati.

Simile alla tecnica di triangolazione, che utilizza la misura degli angoli invece di quella dei lati.

Di seguito sono descritte le tipologie di tecniche per la localizzazione.

Tipologie di scenario:

- **Anchor based:** Esistono nodi “ancore” che conoscono la propria posizione, tutti gli altri nodi usano la posizione delle ancore per calcolare la propria. Si ottiene un sistema di coordinate assoluto.
- **Anchor free:** Nessun nodo conosce la propria posizione. Si ottiene un sistema di coordinate relativo.

Tipologie di tecniche:

- **Range based:** Si basa sulla distanza euclidea tra i nodi.
- **Angle based:** Utilizza la distanza angolare.
- **Range free:** Indipendente dalla distanza euclidea.

La trilaterazione è una tecnica Range based, in questo caso utilizzata in ambiente Anchor based.

1.4 Trilaterazione

La trilaterazione è una tecnica per determinare la posizione, appartenente quindi alla fase di *Positioning*, che è però preceduta dalla fase di *Ranging* in cui vengono calcolate le distanze; questo calcolo viene effettuato tramite ToA (Time of Arrival), per cui la stima della distanza è calcolata in base al tempo di propagazione del segnale, ovvero il *Time of Flight*.

Time of Flight : $T_f = d / c$

d = distanza.

c = velocità di propagazione (= 299702547 m/s , velocità della luce nell'aria)

(299792458 m/s , velocità della luce nel vuoto)

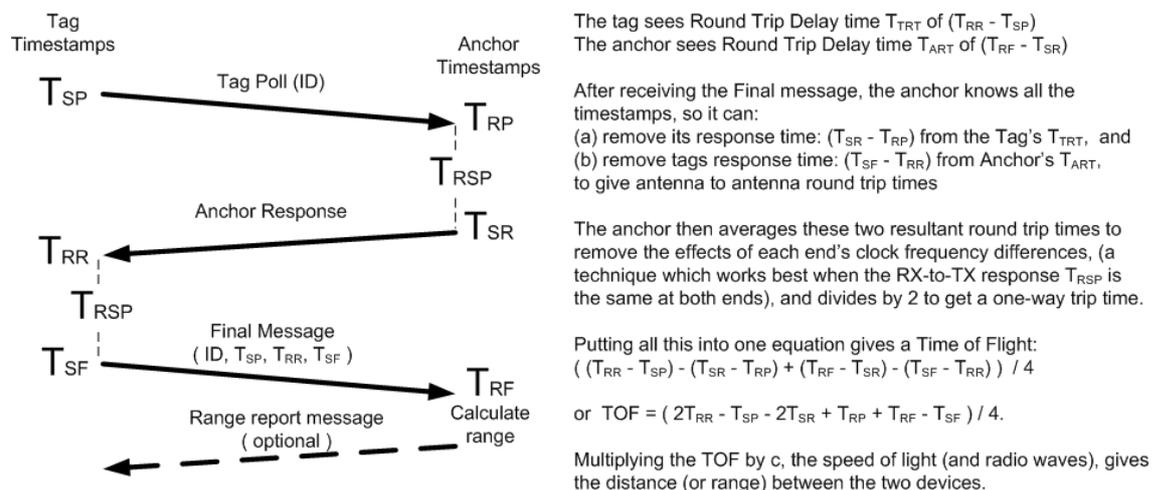


Immagine 1.4.1 : Calcolo del Time of Flight tramite TwoWay Ranging.

Il ToF viene calcolato tramite TwoWay Ranging^[5], ovvero un doppio scambio di messaggi tra l'ancora e il tag.

Nel momento in cui un tag è attivo (ovvero fuori dalla modalità *sleep*), inizia a mandare messaggi di poll per entrare in comunicazione con le ancore.

Nel caso non ottenga risposta da nessuna ancora, riprova a spedire il messaggio, se non riceve risposta di nuovo entra in *sleep*, saltando quindi un aggiornamento della posizione.

In caso contrario, invece, parte il TwoWay Ranging.

Il tag registra il momento in cui viene spedito il messaggio di poll, TSP.

Quando il messaggio arriva all'ancora questa salva il momento di arrivo TRP, dopodichè manda la risposta al Tag, e salva il tempo in cui è stato spedito, il tag salverà il momento in cui viene ricevuto, TSR e TRR.

Infine il tag manda il messaggio finale, al momento TSF, contenente tutti i tempi registrati.

L'ancora a questo punto salva il tempo di arrivo TRF e ha tutte le informazioni per calcolare il ToF, con la formula:

$$((TRR-TSP) - (TSR-TRP) + (TRF-TSR) - (TSF-TRR)) / 4$$

Il ToF, in millisecondi, va quindi moltiplicato per la costante c (velocità della luce), ottenendo il valore in metri.

1.5 Altre tecniche di localizzazione

Ranging:

Un altro modo per effettuare il ranging è il TdoA (Time Difference of Arrival), che utilizza la differenza tra i tempi di arrivo del messaggio alle varie ancore.

Questo metodo richiede però che tutte le ancore siano sincronizzate tra di loro, risultando quindi più complesso.

Altri metodi che non si basano sul tempo sono:

AoA (Angle of Arrival), che richiede antenne direzionali per ottenere l'angolo di arrivo.

RSSI (Received Signal Strength Indicator), si basa sulla potenza del segnale, per calcolare la distanza conoscendo l'attenuazione.

Positioning:

Altre tecniche range based come la trilaterazione sono il Min-Max e la Multilaterazione, mentre la Triangolazione è angle based.

Min-Max:

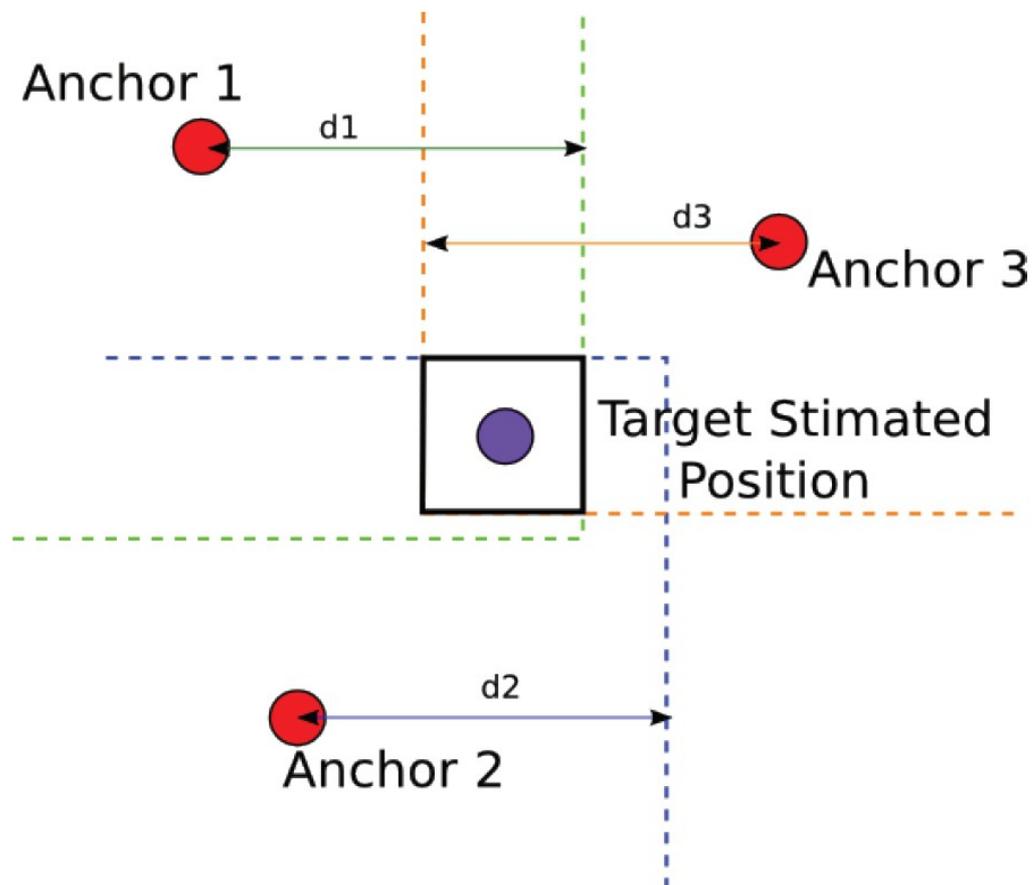


Immagine 1.5.1 : Algoritmo Min-Max.

L'algoritmo Min-Max utilizza l'intersezione di quadrati (o cubi in ambiente tridimensionale) per trovare l'area possibile in cui può essere il punto da localizzare.

I quadrati vengono costruiti in base al range misurato, r .

I lati sono a distanza r dall'ancora, e hanno lato pari a $2r$. L'ancora è al centro del quadrato.

Questo algoritmo è molto semplice, ma l'area risultante dall'intersezione può essere molto grande, ottenendo una localizzazione poco precisa.

Multilaterazione:

La tecnica di multilaterazione è simile a quella di trilaterazione, utilizza però il TDOA per calcolare il ranging, e un numero n di ancore o stazioni.

Calcolando la differenza di arrivo del segnale a due ancore è possibile ottenere delle iperboli, che sono definite appunto come luoghi geometrici i cui punti hanno come costante la differenza delle distanze tra i fuochi (che corrisponderebbero alle ancore).

Utilizza quindi l'intersezione di più iperboli, ognuna ottenuta da una coppia delle n ancore, per localizzare il punto.

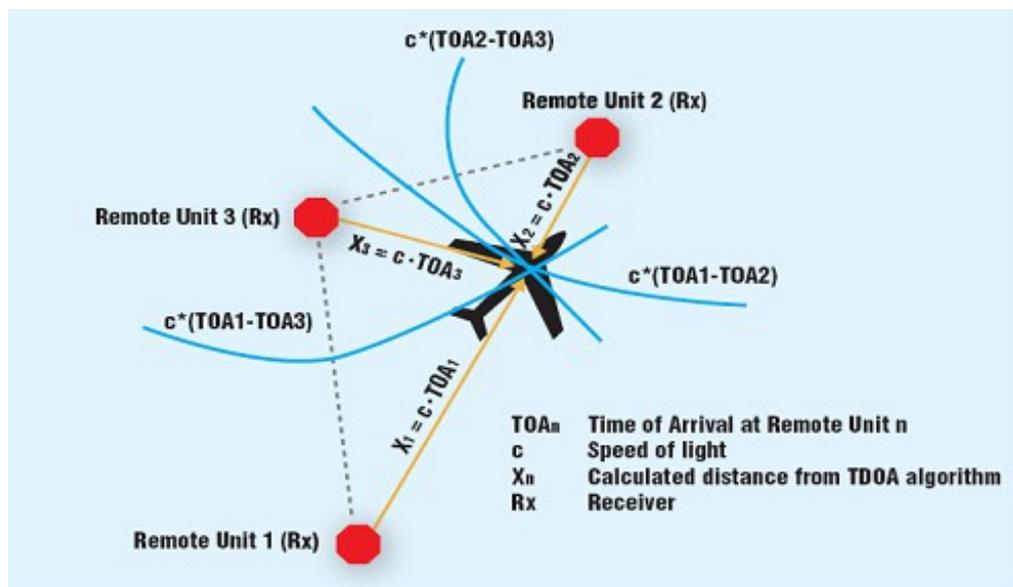


Immagine 1.5.2 : Algoritmo Multilaterazione.

Triangolazione:

La triangolazione è una tecnica angle based, e utilizza formule trigonometriche per trovare le coordinate del punto, partendo dagli angoli che forma con le ancore.

Conoscendo la distanza tra due ancore A e B, e calcolando tramite trigonometria i lati AC e BC, dove C è il punto da localizzare, si ottiene un triangolo, divisibile in due triangoli retti. La base del triangolo ABC è divisa in due parti nel punto X, che corrisponde alla coordinata X del punto da trovare, mentre la coordinata Y è l'altezza dei triangoli.

I cateti AX e BX si possono trovare tramite trigonometria, moltiplicando l'angolo per il coseno del lato, mentre l'altezza è poi possibile ottenerla anche tramite pitagora avendo già 2 lati, oppure sempre tramite trigonometria moltiplicando l'angolo per il seno dell'ipotenusa.

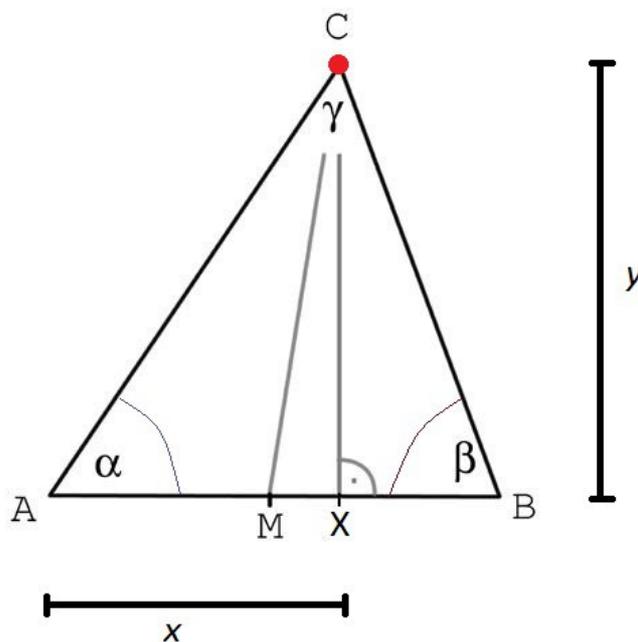


Immagine 1.5.3 : Triangolazione.

2 Analisi dell' algoritmo

La fase di ranging viene effettuata dalle ancore, che periodicamente misurano tramite two-way ranging la propria distanza dal tag.

Le informazioni del ranging vengono codificate in un *report*, sotto forma di stringa, che viene passato alla libreria C.

```
"ma00 t00 00000FC4 00000FC4 0D30 30 0005867C 4034 4034 a0"  
"ma01 t00 00000A8C 00000A8C 0D30 30 0005867C 4034 4034 a0"  
"ma02 t00 0000012B 0000012B 0D30 30 0005867C 4034 4034 a0"  
"ma03 t00 00000C68 00000C68 0D30 30 0005867C 4034 4034 a0"
```

Immagine 2.0.1 : Esempio di report.

Il formato dei report verrà analizzato più avanti nella tesi. (Capitolo 3.3)

I report arrivano al CLE che ne ricava i dati necessari alla trilaterazione, memorizzandoli in variabili e passandoli a un metodo nativo, che chiama la libreria.

Per utilizzare il codice della libreria, scritto in C, nel programma Java del server viene usata l'interfaccia nativa JNI.

2.1 Java Native Interface

La Java Native Interface o JNI, è un framework del linguaggio Java che consente al codice Java di richiamare (o essere richiamato da) codice cosiddetto "nativo", ovvero specifico di un determinato sistema operativo o, più in generale, scritto in altri linguaggi di programmazione, in particolare C, C++ e assembly^[6].

L'interfacciamento è basato sulla definizione di un insieme di classi di raccordo fra i due contesti, che presentano una interfaccia Java, ma che delegano al codice nativo l'implementazione dei loro metodi.

```

#include <jni.h>
/* Header for class com_engynya_horus_businesslogics_LocalizationManager */

#ifndef _Included_com_engynya_horus_businesslogics_LocalizationManager
#define _Included_com_engynya_horus_businesslogics_LocalizationManager
#ifdef __cplusplus
extern "C" {
#endif
#undef com_engynya_horus_businesslogics_LocalizationManager_serialVersionUID
#define com_engynya_horus_businesslogics_LocalizationManager_serialVersionUID 1309107848106833141i64
/*
 * Class:      com_engynya_horus_businesslogics_LocalizationManager
 * Method:     calculatePos
 * Signature:  (Ljava/lang/String;II[I[D[D[DI)D
 */
JNIEXPORT jdoubleArray JNICALL Java_com_engynya_horus_businesslogics_LocalizationManager_calculatePos
    (JNIEnv *, jobject, jstring, jint, jint, jintArray, jdoubleArray, jdoubleArray, jdoubleArray, jdoubleArray, jint);

#ifdef __cplusplus
}
#endif
#endif

```

Immagine 2.1.1 : Dichiarazione del metodo nativo per eseguire la trilaterazione.

Per utilizzare le JNI servono tre elementi.

1. Classe Java in cui viene chiamato e utilizzato il metodo.
2. Codice C contenente l'implementazione del metodo.
3. Header condiviso, presente sia nel progetto Java che nel progetto in C, contenente la dichiarazione del metodo.

Di seguito alcuni dei casi in cui è utile utilizzare la JNI^[7]:

- Quando le librerie standard di Java non supportano caratteristiche dipendenti dalla piattaforma, necessarie all'applicazione.
- Quando si ha già una libreria scritta in un altro linguaggio e la si vuole riutilizzare.
- Quando si vuole implementare una porzione di codice in linguaggio a basso livello per ottenere migliori prestazioni.

2.2 Algoritmo

I dati iniziali sono:

- Coordinate delle 4 ancore, memorizzate in struct con valori double per x,y,z.p1, p2, p3, p4.
- Dati di ranging, in millimetri, sempre memorizzati come double.
r1, r2, r3, r4.

Prima di eseguire l'algoritmo viene eseguito un controllo sulle coordinate delle ancore, per verificare che non ve ne siano di sovrapposte.

Il primo passaggio dell'algoritmo è creare un sistema di coordinate relativo attraverso delle operazioni su vettori, ottenendo 3 vettori unitari, quindi con modulo = 1, ex , ey , ez , che formeranno il nuovo sistema di coordinate relativo.

Il vettore differenza ($p1 - p2$) dà la direzione per il nuovo asse X.

Il vettore ex quindi è un vettore unitario colineare a ($p1 - p2$).

La direzione per il vettore ey invece viene ottenuta dal vettore differenza ($t1 - t2$), dove $t1$ è il vettore ($p3 - p1$) e $t2$ è la sua proiezione sul nuovo asse X, quindi su ex .

La proiezione di $t1$ su ex si ottiene tramite prodotto scalare, infatti il prodotto scalare di due vettori \mathbf{a} e \mathbf{b} , applicati sullo stesso punto, è definito come:

$$a \circ b = |a| \cdot |b| \cdot \cos(\theta)$$

dove θ è l'angolo compreso tra i due vettori.

Avendo due vettori unitari, come in questo caso, il cui modulo è 1, il prodotto dà come risultato il coseno di θ , corrispondente alla proiezione del primo vettore sul secondo.

Infine il vettore ez si ottiene dal prodotto vettoriale (*cross product*) di ex ed ey , che fornisce un vettore perpendicolare ai due.

$ez = \text{cross}(ex, ey)$;

```

vec3d cross(const vec3d vector1, const vec3d vector2)
{
    vec3d v;
    v.x = vector1.y * vector2.z - vector1.z * vector2.y;
    v.y = vector1.z * vector2.x - vector1.x * vector2.z;
    v.z = vector1.x * vector2.y - vector1.y * vector2.x;
    return v;
}

```

Immagine 2.2.1 : funzione che esegue il prodotto vettoriale.

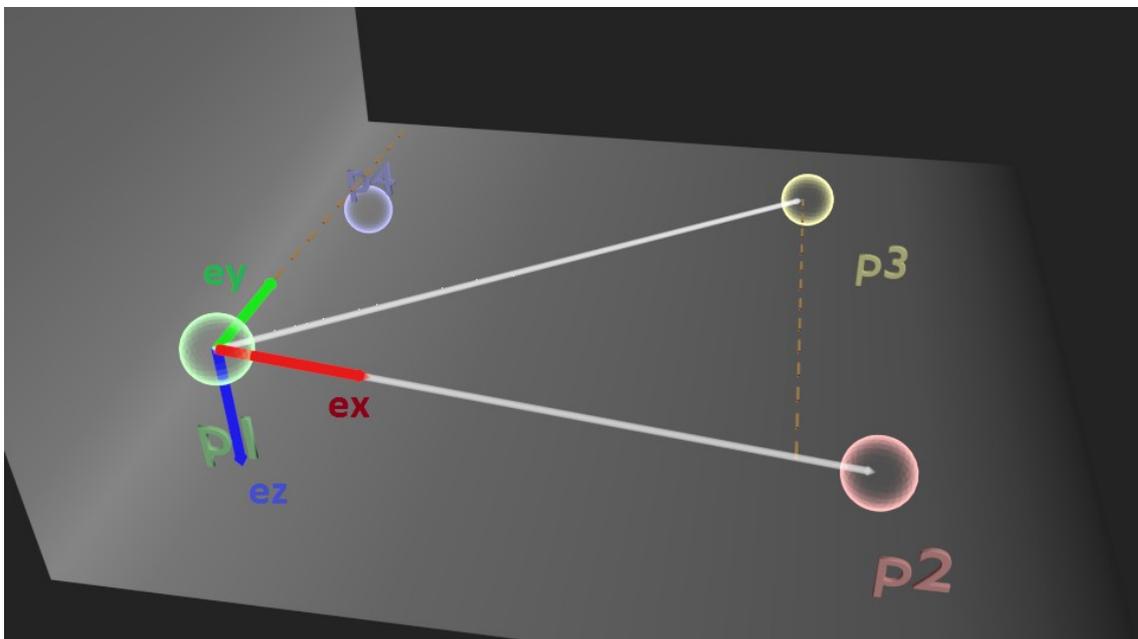


Immagine 2.2.2 : Rappresentazione geometrica del calcolo di ex, ey ed ez.

A questo punto l'algoritmo trova le coordinate relative del punto da localizzare, tramite le proprietà dei triangoli.

Si prende il triangolo formato da r_1 , r_2 , e il segmento $(p_2 - p_1)$.

Dove p_1 e p_2 sono le prime 2 ancore, r_1 ed r_2 sono i range, che puntano al tag, formando quindi un triangolo.

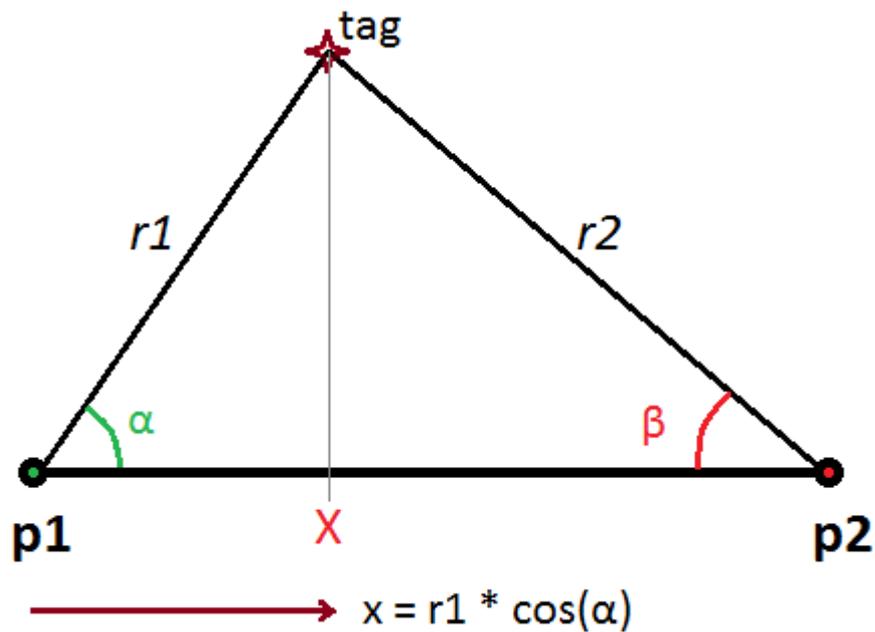
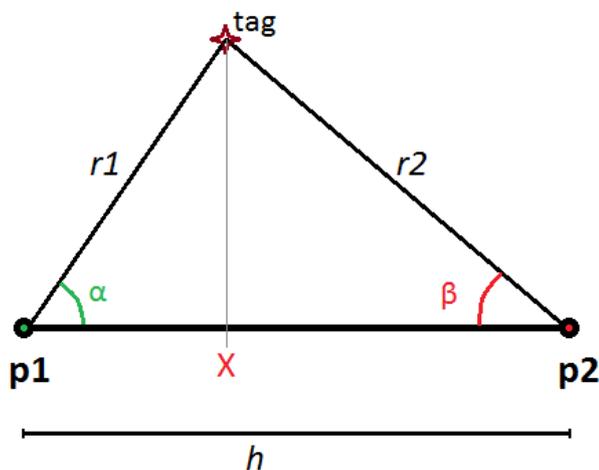


Immagine 2.2.3 : Calcolo della coordinata relativa X tramite trigonometria.

Utilizzando la trigonometria possiamo trovare la x (relativa alle coordinate di p1) moltiplicando il raggio r1 per il coseno dell'angolo che forma con la base (p2 – p1).

Non avendo gli angoli tra i dati disponibili, per trovare il valore di $\cos(\alpha)$ si procede con formula inversa partendo dal teorema di Carnot^[8].

Il teorema di Carnot, o teorema del coseno dice che in un triangolo qualsiasi il quadrato di un lato è uguale alla somma dei quadrati degli altri due diminuita del doppio prodotto di questi due lati per il coseno dell'angolo fra essi compreso.



$$h^2 = r1^2 + r2^2 - 2*r1*r2*\cos(\gamma)$$

$$r1^2 = h^2 + r2^2 - 2*h*r2*\cos(\beta)$$

$$r2^2 = h^2 + r1^2 - 2*h*r1*\cos(\alpha)$$

$$\cos(\alpha) = \frac{r1^2 - r2^2 + h^2}{2 * h * r1}$$

Immagine 2.2.4 : Formula inversa per ottenere il coseno.

Sostituendo quindi alla formula di prima, possiamo calcolare il valore di X come:

$$x = \frac{(r1^2 - r2^2)}{2 \cdot h} + \frac{h}{2}$$

Allo stesso modo, utilizzando il triangolo formato da r2, r3 e il segmento (p3 - p2), si può trovare la y.

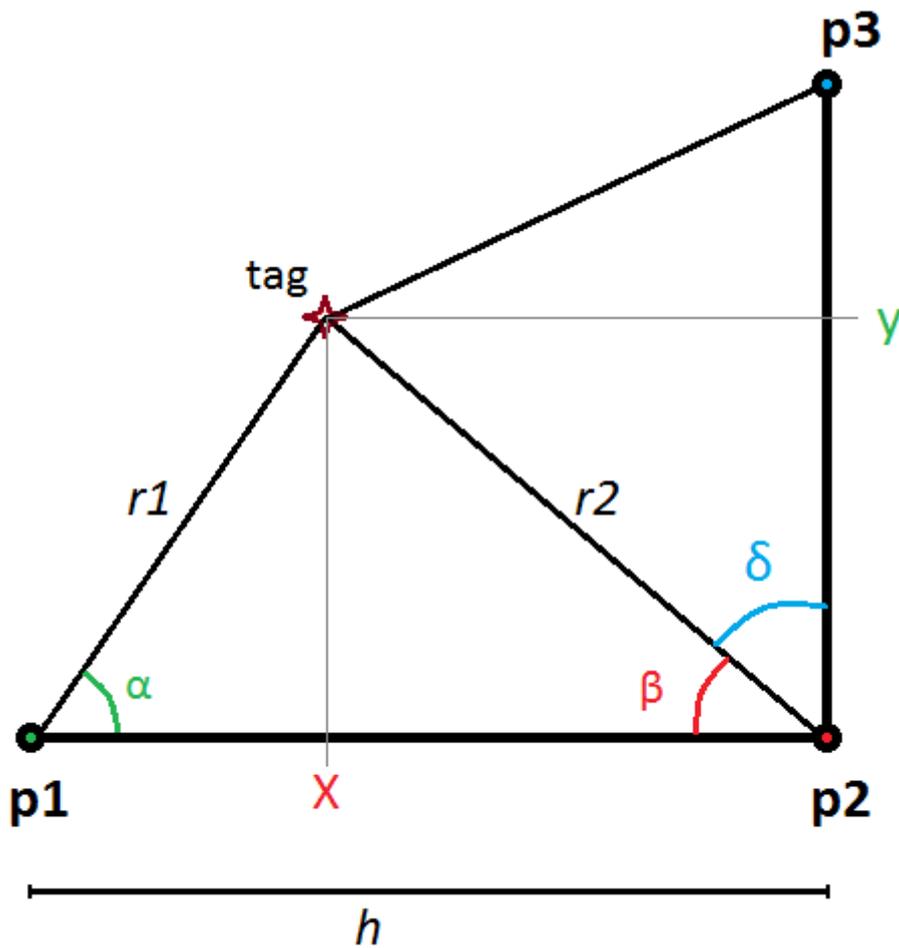


Immagine 2.2.5 : Secondo triangolo per calcolare la coordinata Y.

Per ottenere la Z invece l'algoritmo utilizza la formula del modulo di un vettore. Considerando il raggio r1 come un vettore, avendo le sue componenti X e Y appena calcolate, ottiene la Z per formula inversa:

$$\text{modulo: } r1^2 = x^2 + y^2 + z^2$$
$$z^2 = r1^2 - x^2 - y^2$$

Ottenuto il valore di Z al quadrato, viene eseguito un controllo che questo valore non sia negativo.

Nel caso il valore sia positivo, ne viene eseguita la radice quadrata, ottenendo il valore della coordinata Z.

Ottenuti X, Y e Z, relativi alle coordinate di p1, vengono moltiplicati per i vettori ex, ey ed ez, in modo da ottenere i valori relativi al nuovo sistema di coordinate ottenuto all'inizio.

Dopodichè vengono sommati alla posizione dell'ancora p1, ottenendo la posizione del tag.

Nel caso si ottenga un valore negativo l'algoritmo si interrompe.

In questo caso significa che non vi è intersezione tra le sfere, a causa di errori di ranging.

I raggi delle sfere vengono quindi incrementati, e si ricalcola la posizione con i nuovi valori, in caso di ulteriore fallimento si ripete il procedimento, per un numero fissato di volte, dopodiché se non si è ancora trovato un risultato viene restituito NaN (Not a Number), con un codice di errore, e la trilaterazione fallisce.

Il valore di z, essendo risultato di una radice quadrata, può essere sia positivo che negativo.

Si ottengono quindi due risultati, sommando e sottraendo la z nella direzione ez. Viene quindi utilizzata la quarta ancora come discriminante. Si calcola la distanza di ognuno dei due risultati da p4 e si confronta questo valore con r4 (il range della quarta ancora).

Il vettore risultato con modulo più simile a r4 viene preso come risultato migliore. Nel caso in cui manchi il segnale della quarta ancora, l'algoritmo può funzionare comunque anche con tre, ma verrà scelto arbitrariamente uno dei due risultati.

Esistono casi in cui i due risultati sono a eguale distanza da r_4 . Questo caso si verifica quando le quattro ancore sono disposte su uno stesso piano. In questo caso, un punto posizionato sopra il piano, e uno simmetrico posizionato sotto, sono a eguale distanza da tutte le ancore.

Infatti quattro sfere, adagiate sullo stesso piano, hanno sempre due punti di intersezione comuni a tutte le sfere.

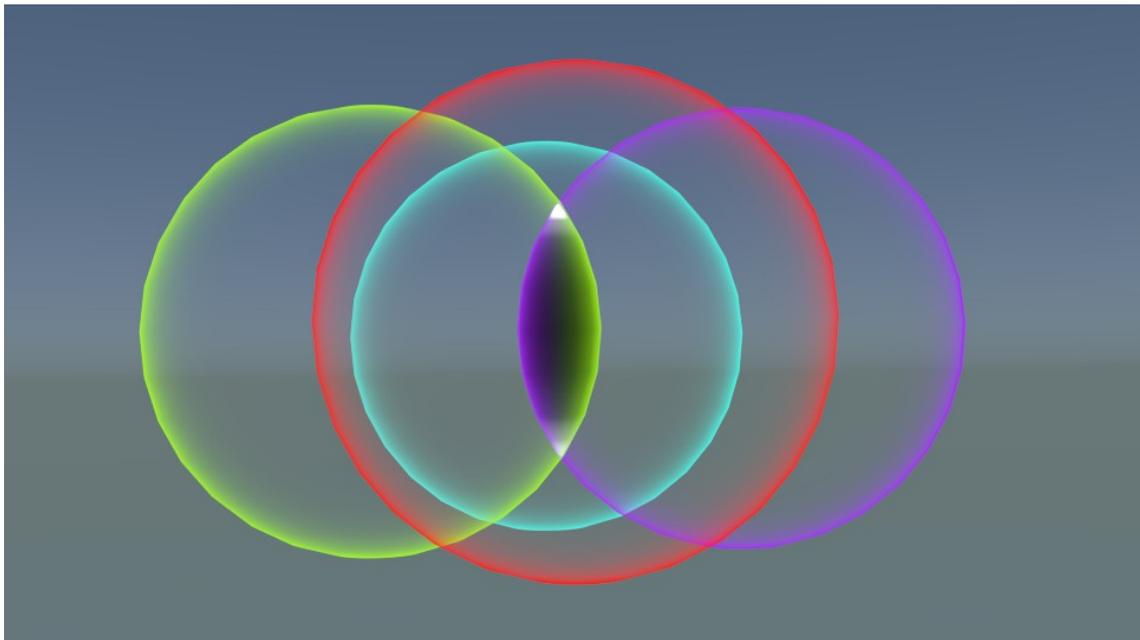


Immagine 2.2.6 : Intersezione di 4 sfere su uno stesso piano.

In queste situazioni viene deciso arbitrariamente, nel codice, quale soluzione prendere, solitamente quella sotto.

Infatti tendenzialmente, nell'ambiente di utilizzo, le ancore vengono montate in alto.

Per evitare questo problema a prescindere si possono montare una o più ancore ad altezza diversa in modo che non siano più disposte sullo stesso piano. In questa situazione infatti si avrà una sola soluzione valida.

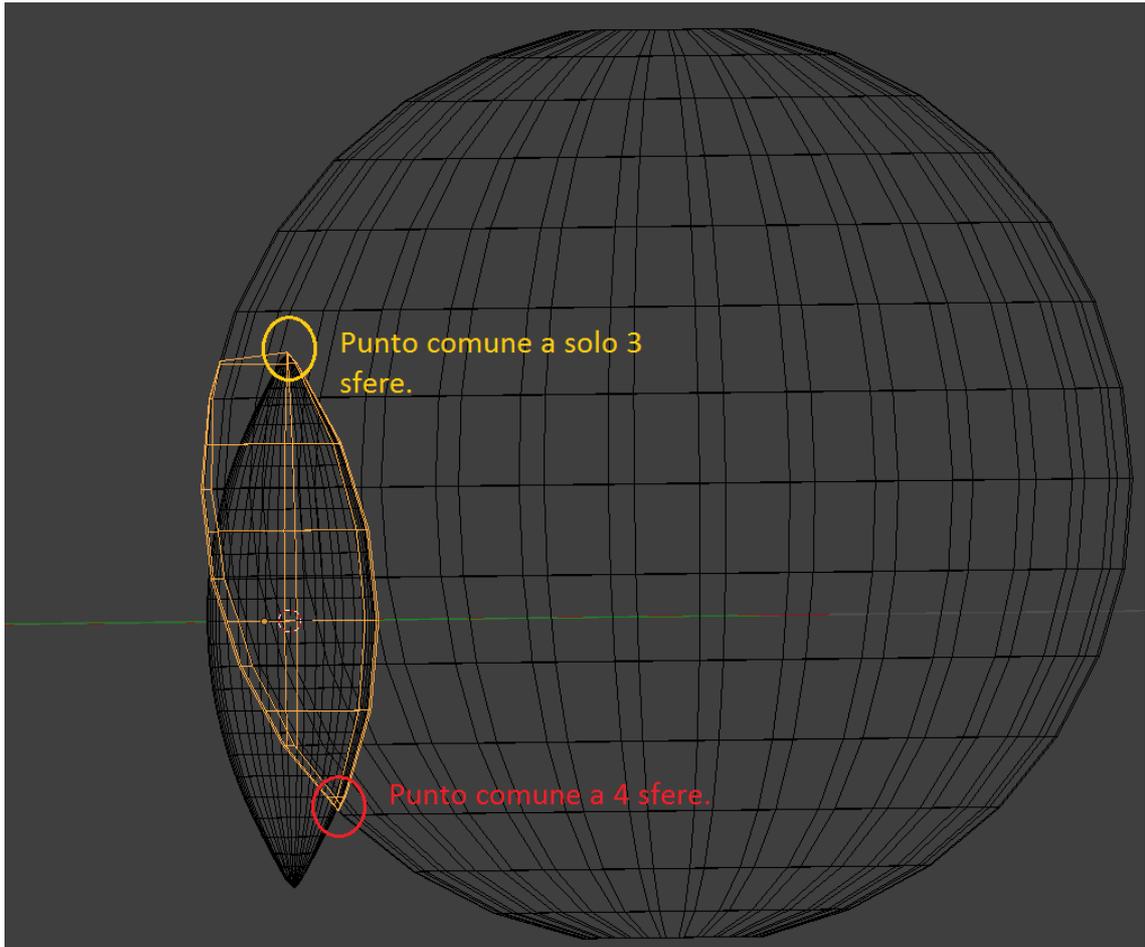


Immagine 2.2.7 : Intersezione con una sfera su piano differente.

Dimostrazione:

Si può dimostrare intuitivamente:

Intersecando due sfere si ottiene un cerchio; intersecandolo con una terza sfera distinta si ottengono due punti.

Per fare in modo che la quarta sfera intersechi entrambi i punti è necessario che questa sia adagiata sullo stesso piano.

Punti equidistanti

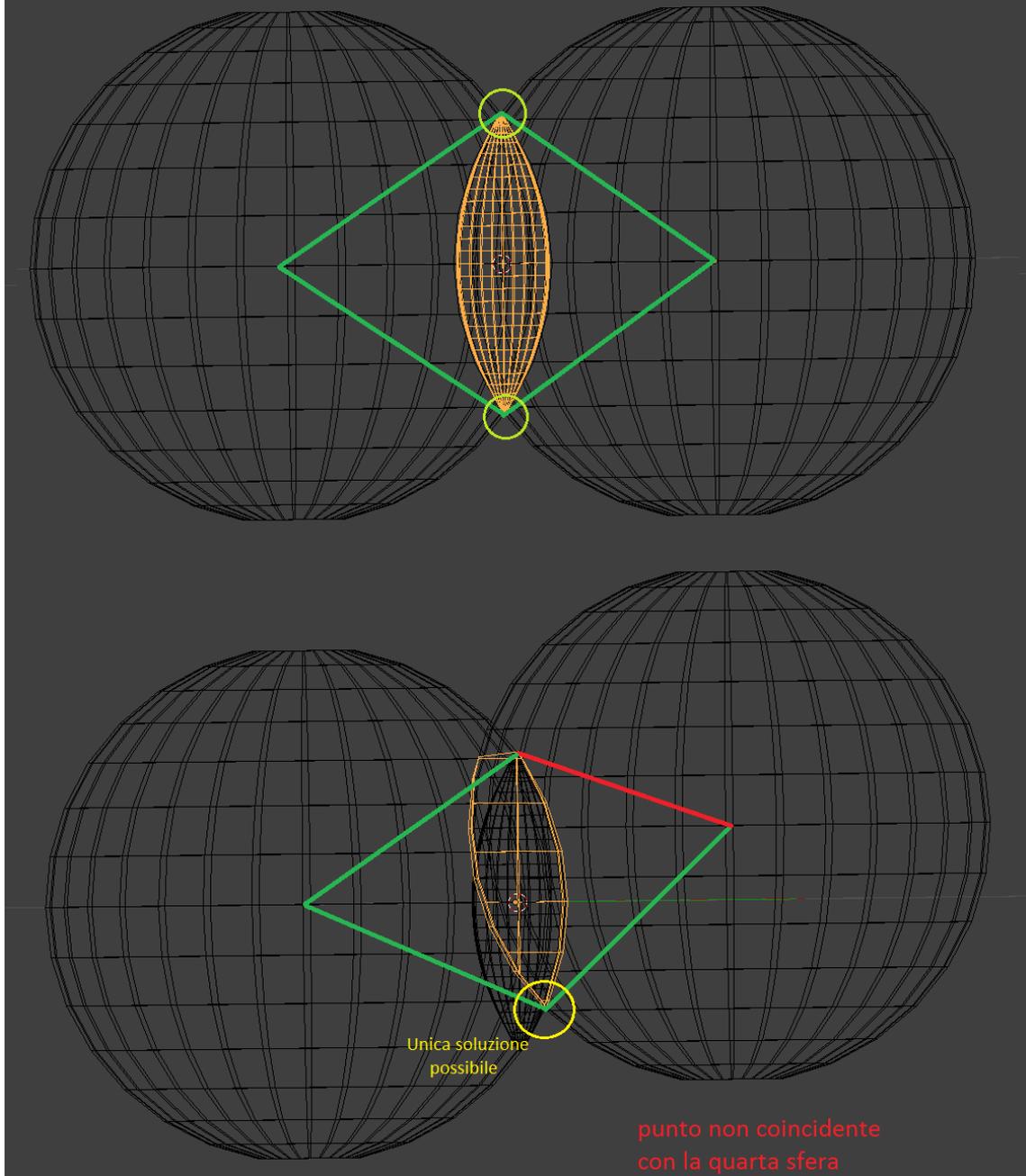


Immagine 2.2.8 : Distanza della quarta sfera dai due punti.

Infatti calcolando la distanza di questi due punti dal centro, con la formula del modulo, abbiamo:

$$\text{Soluzione} = \sqrt{(xa - xt)^2 + (ya - yt)^2 + (za - zt)^2}$$

Dove xt, yt, zt sono le coordinate del tag, mentre xa, ya, za dell'ancora.

Considerando due soluzioni chiamiamo x_{t_1} , y_{t_1} , z_{t_1} le coordinate del primo punto, mentre x_{t_2} , y_{t_2} , z_{t_2} quelle del secondo.

Considerando per semplicità un piano orizzontale posto a $z = 0$, i valori x_{t_1} e x_{t_2} , così come y_{t_1} e y_{t_2} sono sempre uguali.

I valori z_{t_1} e z_{t_2} sono invece opposti, per cui l'unico valore di z_a che permette di ottenere $(z_a - z_{t_1})$ opposto $(z_a - z_{t_2})$ (per cui saranno uguali, elevati al quadrato) è $z_a = 0$.

Nel momento in cui sposto la quarta ancora su una z diversa i valori $|z_a - z_t|$ saranno sempre diversi.

2.3 Geometric Dilution of Precision

La fase principale di *positioning* dell'algoritmo è conclusa.

Dopo questa fase si cerca di migliorare la soluzione ottenuta, la cosiddetta fase di *refinement*.

In questo algoritmo questa operazione viene eseguita tramite calcolo della gdop, ovvero Geometric Dilution of Precision^[9].

La diluizione della precisione, Geometric Dilution of Precision, è un metodo per indicare in che modo gli errori modificano i risultati finali.

L'incidenza dell'errore è legata alla posizione del punto da localizzare rispetto alle antenne.

Se consideriamo un errore sul ranging, le intersezioni delle sfere generano un'area, che dipende dall'angolo compreso tra i raggi, applicato al punto di intersezione.

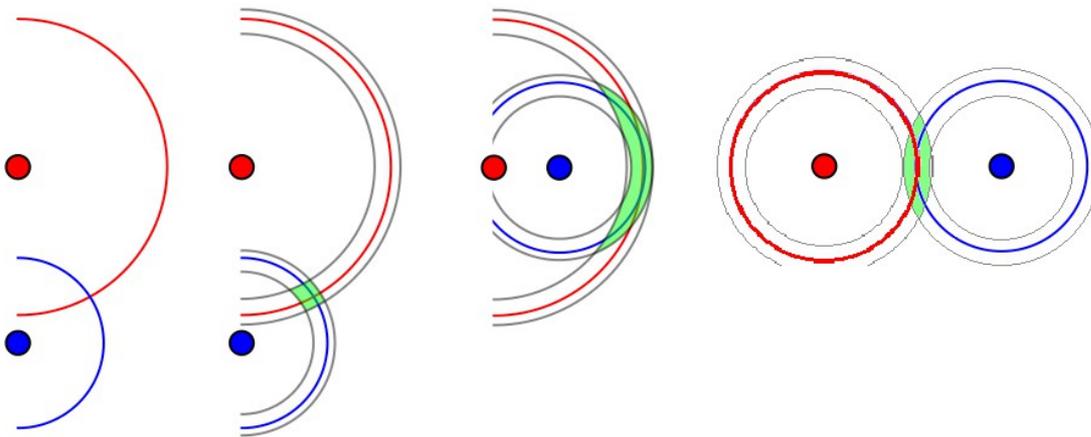


Immagine 2.3.1 : Aree di stima della posizione causate dall'errore.

L'algoritmo quindi itera la trilaterazione, cambiando l'ordine delle ancore ,tramite uno scorrimento a sinistra.

```
ptemp = p1; p1 = p2; p2 = p3; p3 = p4; p4 = ptemp;
rtemp = r1; r1 = r2; r2 = r3; r3 = r4; r4 = rtemp;
```

Immagine 2.3.2 : Shifting dei range.

Il primo risultato quindi viene trovato con p1, p2, p3 e viene scelto usando p4. Il secondo invece viene trovato con p2, p3, p4 e viene scelto tramite p1. E così via. Dopo aver trovato ogni risultato si esegue il calcolo del GDOP, trovando il valore per ogni coppia di ancore, basandosi sull'angolo che i loro range formano. Per trovare il valore di gdop si utilizza il coseno dell'angolo.

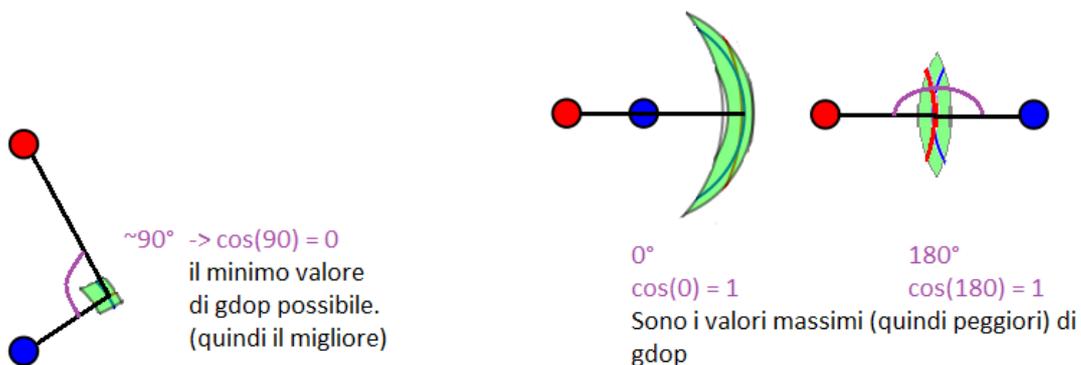


Immagine 2.3.3 : Calcolo della GDOP.

Il coseno fornisce il valore minimo, quindi migliore, a 90° . Per trovare questo valore vengono calcolate le distanze tra ancora e tag (ovvero il risultato appena trovato), facendo la differenza dei vettori posizione delle ancore e facendone il modulo, dopodichè vengono divisi i vettori per il proprio modulo in modo da ottenere vettori unitari.

In questo modo, il modulo di un vettore unitario corrisponde a 1, per cui il prodotto scalare tra i due risulta uguale al coseno dell'angolo tra essi compreso.

Esattamente il valore che si stava cercando.

Si ottengono 3 valori di $gdop$, uno per ogni coppia di ancore, nel gruppo di tre. A seconda dell'implementazione si può prendere in considerazione il massimo fra i tre, il minimo, o una media.

Questo valore viene poi confrontato e si mantiene il risultato di localizzazione del gruppo di ancore con un valore migliore, che quindi si trovano disposte in maniera migliore nello spazio, nei confronti del tag.

2.4 Aggiunte e modifiche

Concentricità:

Nel caso in cui delle sfere siano concentriche, non si potrà mai ottenere una intersezione.

Nell'algorithmo iniziale veniva controllato solamente in caso in cui due ancore fossero coincidenti, ritornando un errore nel caso la distanza tra le due fosse prossima a 0.

Ho quindi aggiunto un controllo della concentricità, che viene eseguito prima di chiamare l'algorithmo di trilaterazione.

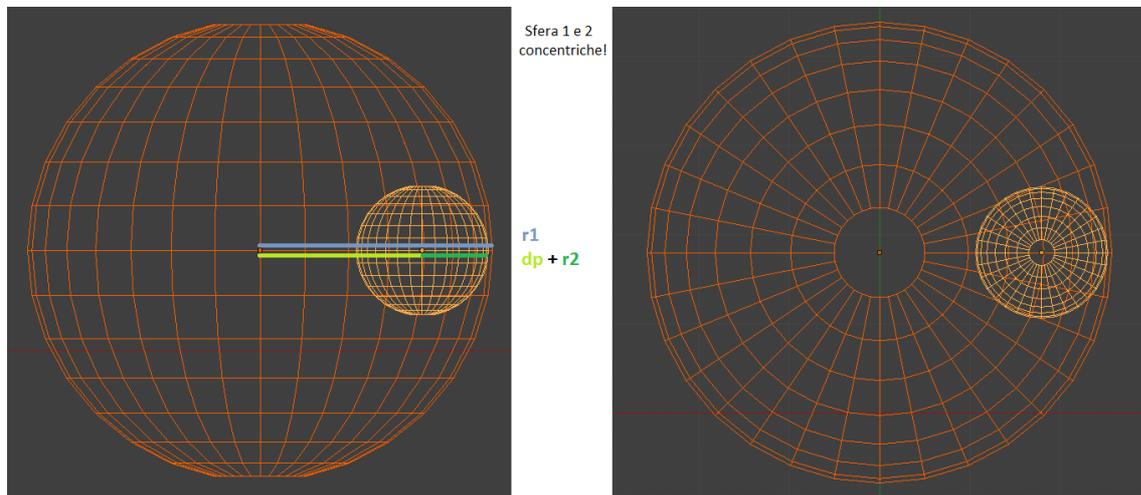


Immagine 2.4.1 : Sfere concentriche.

Per ogni coppia di ancore si calcola la distanza dp tra le due.

Se il raggio di una sfera è maggiore di $dp +$ il raggio della seconda sfera, le sfere sono concentriche, quindi sicuramente c'è stato un errore di ranging.

A questo punto viene ridimensionato il raggio della sfera più grande (nell'esempio $r1$) in modo che:

$$r1 = dp + r2;$$

Questo è il caso limite possibile. Il raggio può essere uguale alla somma della distanza più il secondo raggio, nel caso in cui il tag sia esterno alle ancore e sulla stessa linea.

Incremento e decremento:

Quando viene calcolata la coordinata z , si è visto nel capitolo 2.2 che potrebbe risultare un valore di z al quadrato negativo.

In questo caso i range vengono incrementati e si riesegue la trilaterazione.

Può però capitare che questo valore diventi a ogni iterazione più negativo, rendendo impossibile trovare la soluzione.

Ho quindi aggiunto una variabile che mantiene il valore precedente di z^2 ; in questo modo si può eseguire un controllo per accorgersi quando il valore diventa più negativo.

A questo punto l'algoritmo inizia a decrementare i range invece che incrementarli.

In questo modo l'algoritmo riesce a trovare una soluzione, anche se molto imprecisa, anche in casi di errori di ranging di enorme entità.

Questi casi sono però molto rari e inoltre il risultato trovato è comunque impreciso; un'altra possibile implementazione poteva quindi essere quella di ritornare un errore che interrompesse subito l'iterazione dell'algoritmo qualora si verificasse questo caso.

Una seconda modifica inerente a questa meccanica è la scelta del valore di incremento.

Nella versione iniziale venivano incrementati i range di 0,1.

I range sono in metri, quindi venivano incrementati di 10 centimetri.

default:

```
ovr_r1 += 0.10;  
ovr_r2 += 0.10;  
ovr_r3 += 0.10;  
ovr_r4 += 0.10;  
overlook_count++;  
break;
```

Immagine 2.4.2 : Codice iniziale dell'incremento.

Questo causava dei risultati poco precisi, quando si doveva ricorrere all'incremento, infatti quando l'errore non era così elevato venivano incrementate troppo.

Semplicemente diminuendo questo valore si ottengono dei risultati migliori.

```
default: // any other return value goes here
printf("default\n");
ovr_r1 += 0.10;
ovr_r2 += 0.10;
ovr_r3 += 0.10;
ovr_r4 += 0.10;
overlook_count++;
break;
```

Tag position: [1.500000, 1.500000, 2.657907]

```
default: // any other return value goes here
printf("default\n");
ovr_r1 += 0.010;
ovr_r2 += 0.010;
ovr_r3 += 0.010;
ovr_r4 += 0.010;
overlook_count++;
break;
```

Tag position: [1.500000, 1.500000, 2.202882]

```
default: // any other return value goes here
printf("default\n");
ovr_r1 += 0.001;
ovr_r2 += 0.001;
ovr_r3 += 0.001;
ovr_r4 += 0.001;
overlook_count++;
break;
```

Tag position: [1.500000, 1.500000, 2.053703]

Posizione reale : [1.5, 1.5, 2.0]

Immagine 2.4.3 : Comparazione tra diversi valori di incremento.

Si è quindi impostata una costante a un millimetro (0.001), inoltre, avendo diminuito l'incremento, si è aumentato il numero di iterazioni, per i casi di errori molto grandi che richiedono tanti incrementi.

```

#define      INC 0.001
#define      ITERATIONS 500

case ERR_DIMINISHING:
    ovr_r1 -= INC;
    ovr_r2 -= INC;
    ovr_r3 -= INC;
    ovr_r4 -= INC;
    overlook_count++;
    break;
default:
    ovr_r1 += INC;
    ovr_r2 += INC;
    ovr_r3 += INC;
    ovr_r4 += INC;
    overlook_count++;

```

Immagine 2.4.4 : Codice finale dell' incremento.

Media:

Nella fase di refinement si calcola la gdop di varie combinazioni di ancore per scegliere quella migliore.

Una diversa soluzione è quella di tenere tutti i risultati ottenuti dalle varie combinazioni di ancore ed eseguirne una media.

Viene eseguita una media pesata sul valore di gdop.

Il valore di gdop va da 0 (ottimo) e 1 (pessimo).

Per dare maggiore importanza ai risultati con un valore migliore, si moltiplica la soluzione per $(1 - \text{gdop})$.

Dopodiché si sommano le soluzioni.

Alla fine viene eseguita la divisione per il numero di soluzioni moltiplicato per $(1 - \text{best_gdop})$, dove `best_gdop` è una variabile contenente il valore migliore di gdop ottenuto.

Per aumentare il peso della soluzione con gdop migliore, si può eseguire una seconda media, tra questa soluzione e la soluzione media ottenuta.

Entrambe queste medie possono essere abilitate o disabilitate cambiando il valore di una costante da codice.

Infatti nei diversi test la media migliora il risultato solo poche volte, altre volte la soluzione pura, con gdop migliore, risulta più precisa della media.

3 Applicazione di testing

Al fine di valutare e confrontare gli algoritmi e di effettuare vari tipi di test, si è sviluppata una applicazione che simulasse il funzionamento del sistema ma con dati ideali.

Il programma è composto da due principali blocchi di funzionalità:

- Inserimento di tag con coordinate specifiche in un campo di ancore definito dall'utente, con possibilità di eseguire due diversi algoritmi che possono quindi essere confrontati.
- Lettura di file di log di test reali effettuati sul campo a scopo di studio o di verifica degli errori.

3.1 Componenti dell'interfaccia

Ci sono tre modalità di visualizzazione:

- Mappa
- Tabella
- Grafico

Ognuna fornisce le informazioni sulla posizione del tag in modo diverso.

Per inizializzare e modificare i tag e le ancore, più altri parametri per le diverse funzioni del programma, esiste una finestra di *Settings*, mentre per far partire e fermare il programma c'è una console di comando.



Immagine 3.1.1 : Pulsanti.

Mappa:

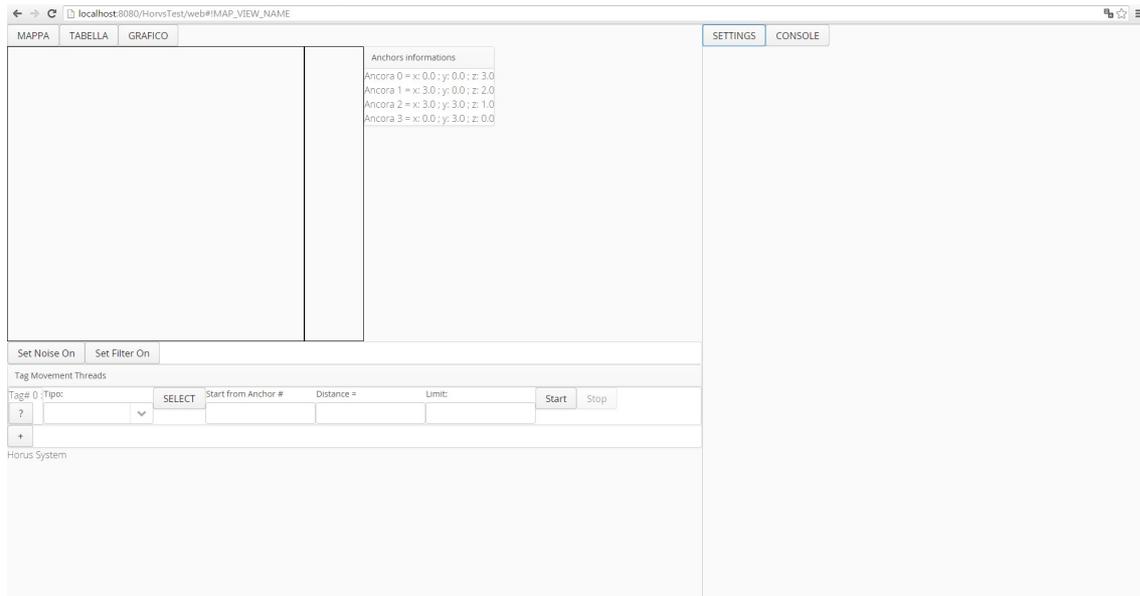


Immagine 3.1.2 : Modalità di visualizzazione mappa, applicazione ferma.

La modalità di visualizzazione mappa comprende:

- Un canvas su cui compare il quadrante di quattro ancore e i tag inizializzati, che compaiono come quadratini colorati, mentre i tag di riferimento, ovvero quelli con coordinate ideali, con cui confrontare le posizioni calcolate, sono quadratini neri. In questo riquadro vengono visualizzate le coordinate x e y.
- Un secondo canvas rettangolare in cui visualizzare la coordinata z. L'altezza, quindi la coordinata z, delle ancore è visualizzata con una linea grigia, mentre quella dei tag come una linea colorata.
- Un pannello informativo con le coordinate delle ancore.

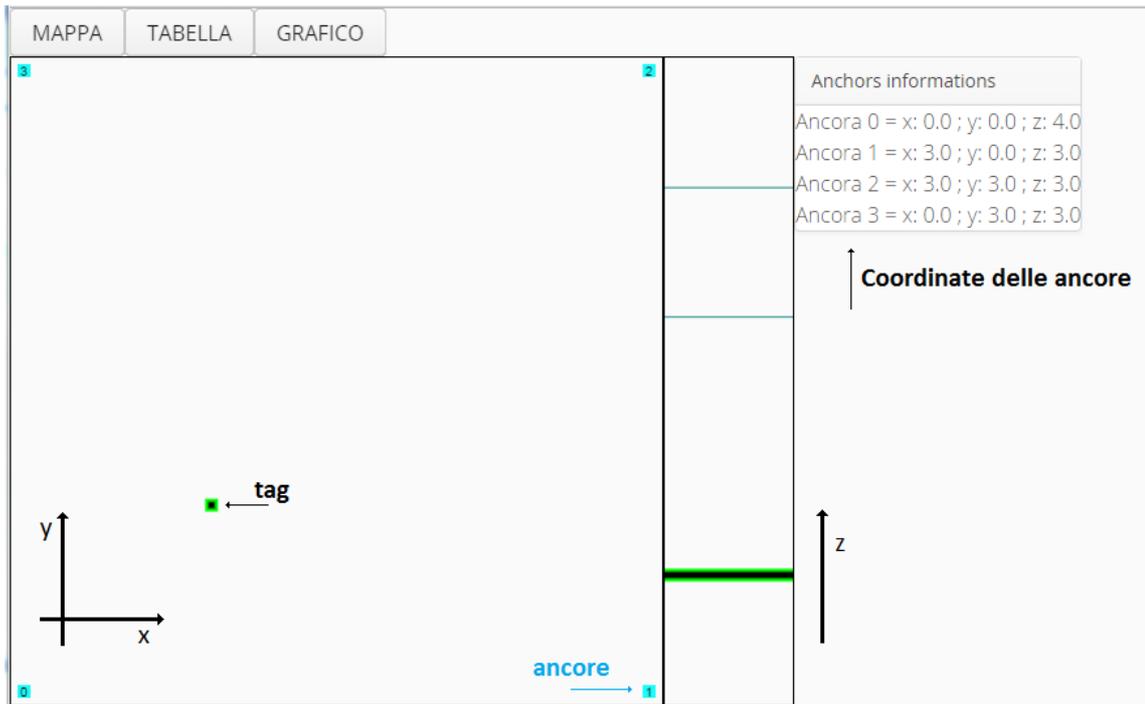


Immagine 3.1.3 : Visualizzazione mappa con applicazione attiva.

Tabella:

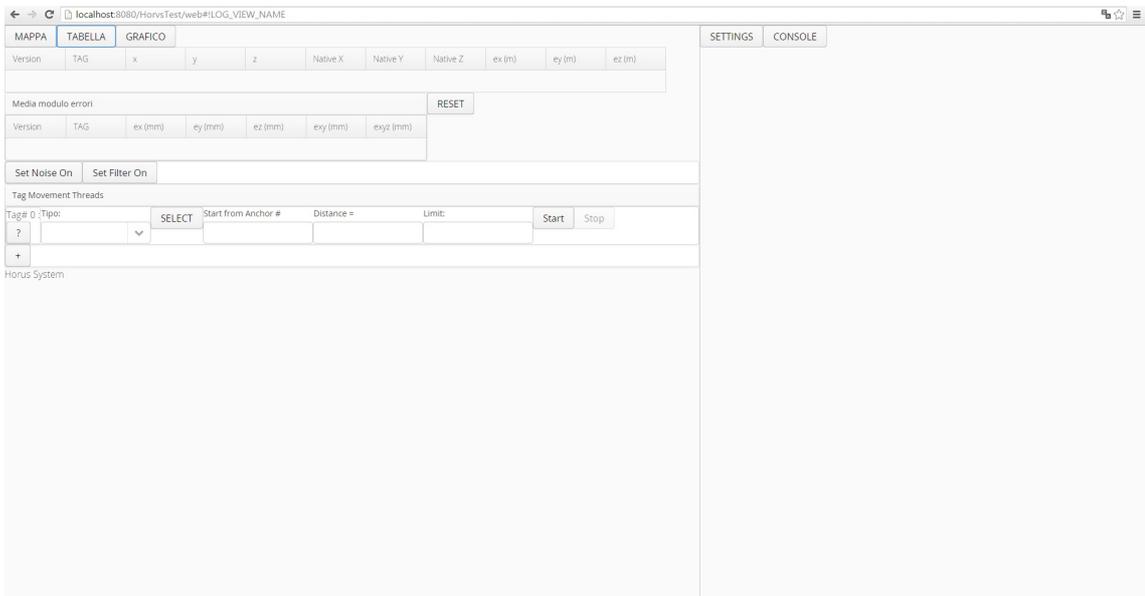


Immagine 3.1.4 : Modalità di visualizzazione tabella, applicazione ferma.

La modalità di visualizzazione tabella comprende:

- Una tabella in cui sono visualizzate le informazioni sui tag. Coordinate calcolate, coordinate ideali, errori di posizione in x,y e z, ovvero la differenza tra la coordinata calcolata e quella ideale; il codice identificativo del tag e la versione dell'algoritmo utilizzata.
- Una seconda tabella in cui è mostrato l'errore medio cumulativo, ovvero la media di tutti i moduli degli errori che vengono calcolati ad ogni localizzazione. Sono divisi nelle componenti x,y e z; poi c'è il modulo dell'errore bidimensionale, calcolato solo su x e y, e quello tridimensionale calcolato su x, y e z.
- Un tasto reset per azzerare le medie degli errori presenti nella seconda tabella, per ricominciare a calcolarle da capo.

MAPPA	TABELLA	GRAFICO								
Version	TAG	x	y	z	Native X	Native Y	Native Z	ex (m)	ey (m)	ez (m)
1	0	0,9	0,9	1,001	0,9	0,9	1	-0	-0	0,001
Media modulo errori							<input type="button" value="RESET"/>			
Version	TAG	ex (mm)	ey (mm)	ez (mm)	exy (mm)	exyz (mm)				
1	0	0	0	0	0	0				

Immagine 3.1.5 : Visualizzazione tabella, applicazione attiva.

MAPPA	TABELLA	GRAFICO								
Version	TAG	x	y	z	Native X	Native Y	Native Z	ex (m)	ey (m)	ez (m)
0	0	0,509	1,369	2	0,509	1,369	2	0	-0,001	0
1	0	0,508	1,369	2,001	0,509	1,369	2	-0	-0	0,001
Media modulo errori							<input type="button" value="RESET"/>			
Version	TAG	ex (mm)	ey (mm)	ez (mm)	exy (mm)	exyz (mm)				
0	0	1,613	0,279	4,744	1,748	5,125				
1	0	0,35	0,278	1,185	0,506	1,336				

Immagine 3.1.6 : Visualizzazione tabella, confronto dei risultati di due algoritmi.

Grafico:

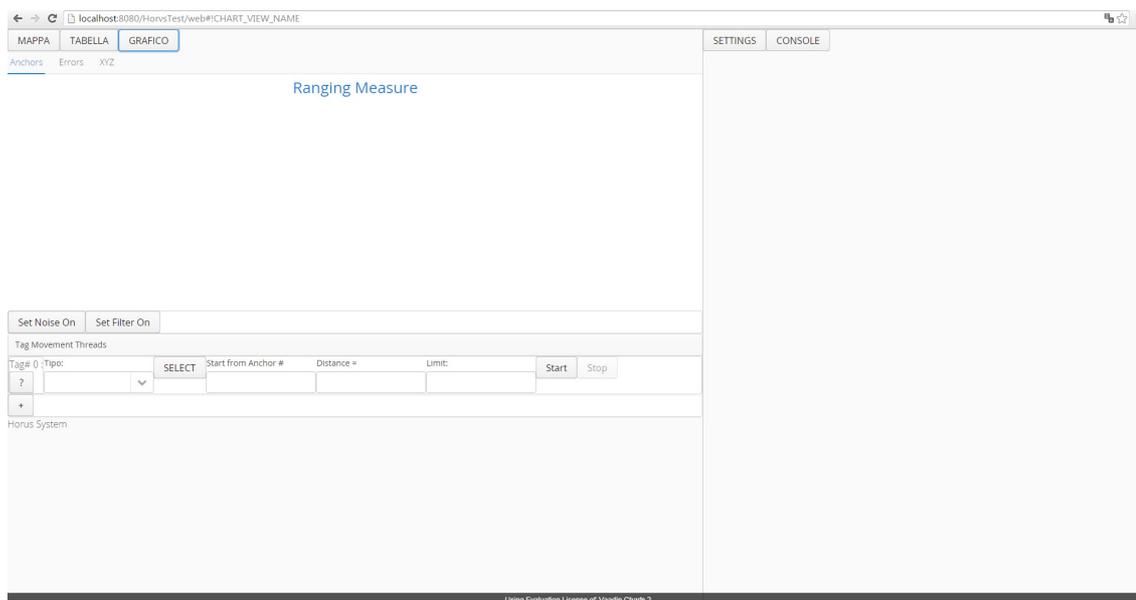


Immagine 3.1.7 : Modalità di visualizzazione grafico, applicazione ferma.

La modalità di visualizzazione grafico comprende:

- Un grafico a barre che rappresenta la distanza di ogni tag dalle ancore, in millimetri.
- Un grafico a linee, che mostra l'errore di ranging, ovvero la differenza tra il range effettivo, calcolato sui dati ideali, e quello misurato. Nell'applicazione questi valori coincideranno sempre, in quanto il sistema è simulato, quindi ideale, ma è possibile applicare un rumore artificiale ai ranging per poterlo osservare nel grafico. Inoltre può essere utilizzato nella lettura di log di misurazioni reali.
- Un grafico a punti, tridimensionale, che mostra lo spostamento del tag nello spazio.

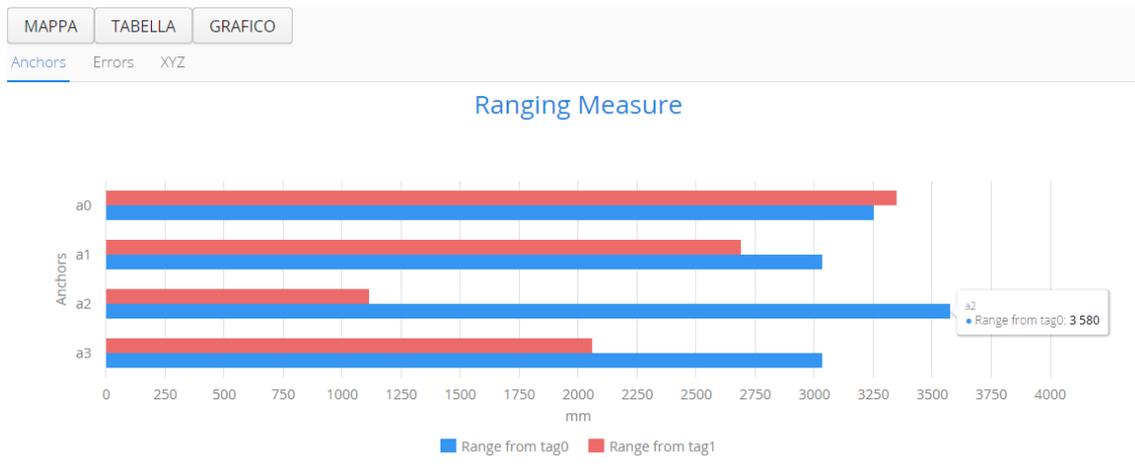


Immagine 3.1.8 : Grafico a barre.

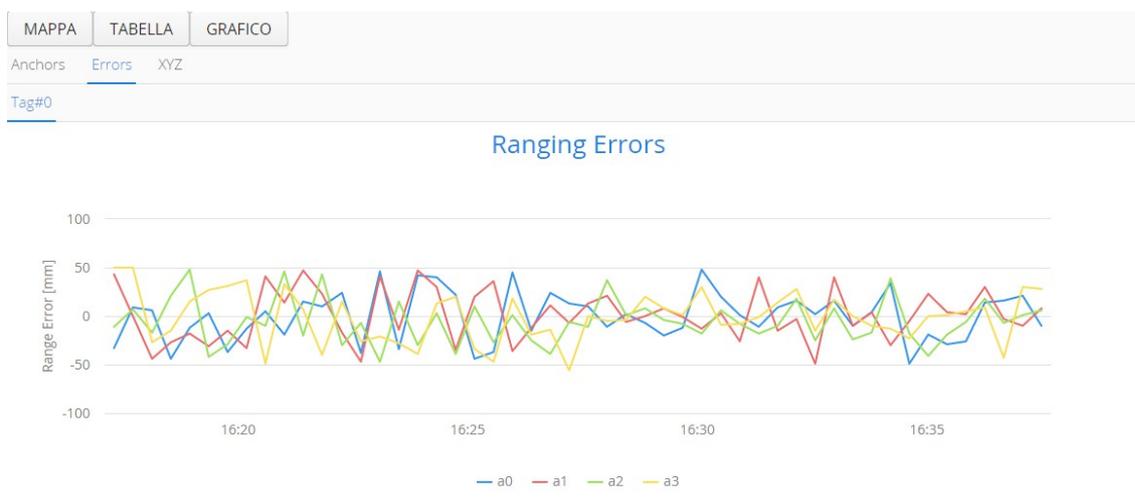


Immagine 3.1.9 : Grafico del rumore.

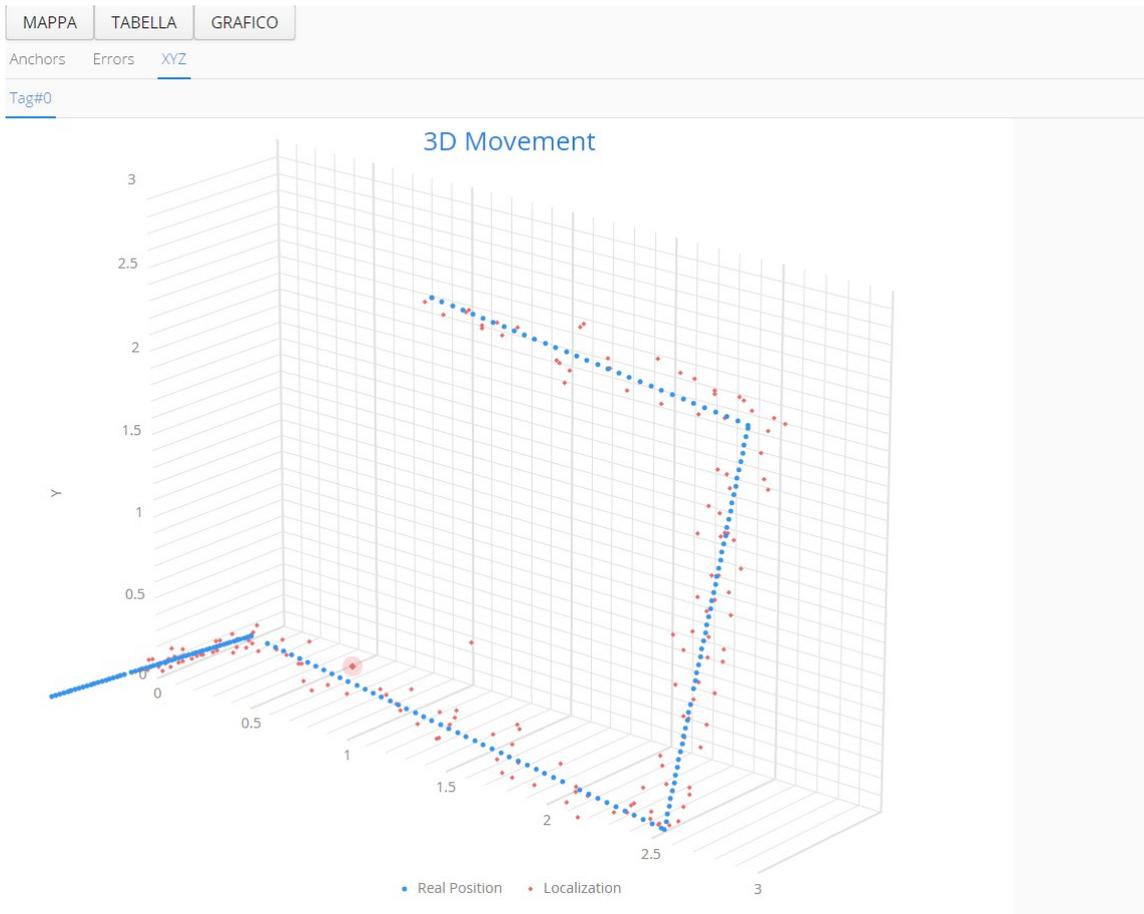


Immagine 3.1.10 : Grafico dello spostamento, con rumore applicato.

Settings:

SETTINGS	CONSOLE		
Tags settings			
Tag attivi: 2			
Tag #	x =	y =	z =
1	2	2.5	3
Init Tag	Show Reports	Tag #	
Anchors settings			
Anchor #0	x =	y =	z =
Anchor #1	x =	y =	z =
Anchor #2	x =	y =	z =
Anchor #3	x =	y =	z =
Init anchors			
Output logs			
Enable log prints	Confirm	Current Mode: Enable all logs	
Disable all logs			
Log Output Path	Confirm		
Current Directory: C:\Users\Daniele\Documents\workspace mars\HorvsTest\log\			
Input logs			
Scegli file	Nessun file selezionato	Upload	
Transporter step			
Current Value: 0.05			
Virtual noise settings			
Positive Noise (m): 0.05 Negative Noise (m): -0.05 Average value (m): 0.0			
UNIFORM			
Filtering parameters			

Immagine 3.1.11 : Schermata delle impostazioni

Questo menu presenta le varie impostazioni riferite alle diverse funzioni del programma.

- Impostazioni dei tag:

Tags settings			
Tag attivi: 2			
Tag #	x =	y =	z =
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Init Tag"/>	<input type="button" value="Show Reports"/>	Tag # <input type="text"/>	

Immagine 3.1.12 : Pannello di impostazioni per i tag.

Questo pannello contiene una etichetta che mostra il numero di tag inizializzati, una serie di campi di testo con cui si può posizionare un tag esistente (inserendo nel primo campo il suo identificativo) o un tag nuovo (inserendo un id che non esiste) nelle coordinate x,y,z specificate. Il tasto “Init Tag” applica l’inizializzazione specificata nei campi di testo. Il tasto “Show Reports” mostra i report (le stringhe formattate coi valori esadecimali dei range) relativi al tag specificato nel campo di test a fianco.

```
ma00 t00 000007AF 000007AF 00EE EE 0007CDFA 4034 4034 a0
ma01 t00 00000922 00000922 00EE EE 0007CDF9 4034 4034 a0
ma02 t00 00000BC3 00000BC3 00EE EE 0007CDF9 4034 4034 a0
ma03 t00 00000AAD 00000AAD 00EE EE 0007CDF9 4034 4034 a0
```

Immagine 3.1.13 : Notifica contenente i report.

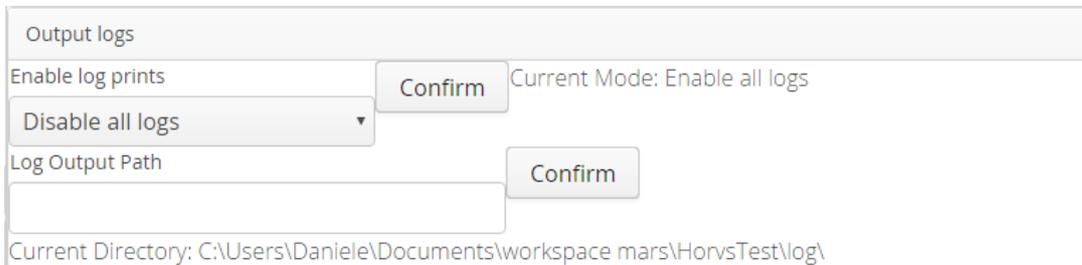
- Impostazioni delle ancore:

Anchors settings			
Anchor #0	x =	y =	z =
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Anchor #1	x =	y =	z =
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Anchor #2	x =	y =	z =
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Anchor #3	x =	y =	z =
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Init anchors"/>			

Immagine 3.1.14 : Pannello di impostazione per le ancore.

Questo pannello permette di impostare le coordinate delle ancore, in modo da poterne cambiare la disposizione durante il funzionamento dell'applicazione.

- Impostazioni per la stampa dei log:



Output logs

Enable log prints Current Mode: Enable all logs

Disable all logs ▼

Log Output Path

Current Directory: C:\Users\Daniele\Documents\workspace mars\HorvsTest\log\

Immagine 3.1.15 : Pannello di impostazioni per i log.

L'applicazione durante il funzionamento stampa dei file di log; tramite questo pannello si può scegliere quali log stampare e cambiare il percorso in cui salvarli.

Le scelte disponibili sono:

Non stampare nessun log; stampa tutto; stampa solo log posizione; solo report; solo errori; solo errori di range; solo errori di posizione.

- Lettura di log:



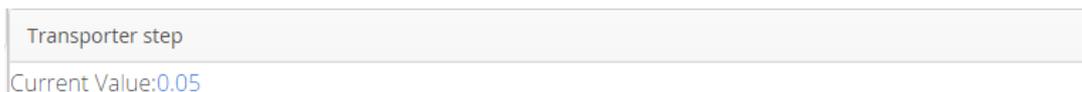
Input logs

Scegli file Nessun file selezionato

Immagine 3.1.16 : Pannello per caricare un file di test.

Tramite questo pannello si può caricare da disco un file di test, ovvero dei log ottenuti durante il funzionamento reale del sistema, per poterne eseguire una simulazione. (Maggiori informazioni nel paragrafo 3.3).

- Impostazioni delle traslazioni:



Transporter step

Current Value:[0.05](#)

Immagine 3.1.17 : Impostazione dello step.

In questo pannello si imposta il valore dello step, ovvero la distanza (in metri) che il tag percorre a ogni passo, durante uno spostamento.

Gli spostamenti verranno analizzati più avanti.

- Impostazioni del rumore:



Virtual noise settings

Positive Noise (m): 0.05 Negative Noise (m): -0.05 Average value (m): 0.0

UNIFORM

Immagine 3.1.18 : Impostazioni del rumore.

Durante le simulazioni è possibile applicare un rumore artificiale sulle misure di ranging, per cercare di produrre un comportamento più simile alla realtà.

In questo pannello si possono configurare i valori del rumore.

Il tasto permette di cambiare da un rumore distribuito uniformemente a una distribuzione gaussiana.

I campi “Positive Noise (m)” e “Negative Noise (m)” permettono di modificare il valore massimo e quello minimo che può avere il rumore. L'etichetta “Average Value” di fianco mostra il valor medio del rumore. Questi campi sono solo indicativi quando è attivo il rumore gaussiano, che non è limitato, quindi rientrerà in quei valori la maggiorparte delle volte ma non sempre.

- Impostazioni dei filtri:



Filtering parameters

Validity period (ms) 120000

Buffer Length 10

Alpha

APPLY

Immagine 3.1.19 : Pannello delle impostazioni per il filtraggio.

In questo pannello di possono cambiare i parametri del filtro.

Al momento è implementato un unico tipo di filtro che non utilizza tutti questi valori, ad esempio il valore “Buffer Length” per valori superiori a 1 non produce cambiamenti.

Il tasto “Apply” permette di resettare i filtri e reimpostarli con i nuovi valori.

Maggiori informazioni sui filtri verranno fornite nel capitolo 5.

Console:

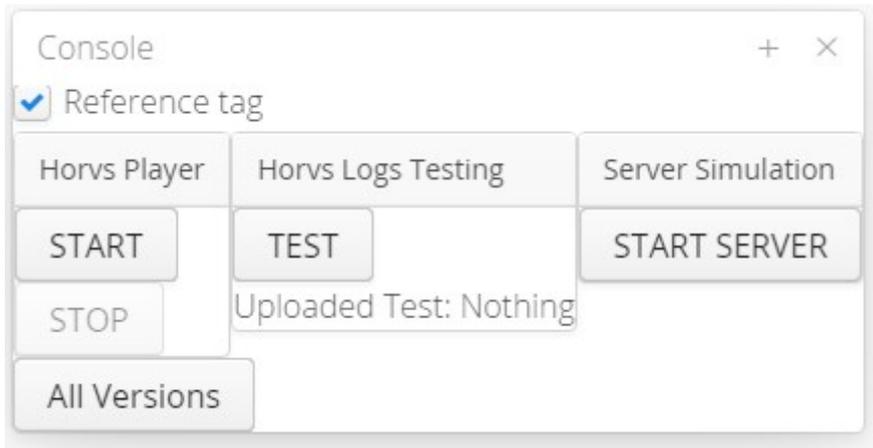


Immagine 3.1.20 : Console di comando dell'applicazione.

L'attivazione e interruzione del programma, l'esecuzione di test e altre funzioni, sono accessibili dalla console, che si può aprire e chiudere con l'apposito tasto.

- Spunta “Reference tag” : permette di visualizzare o meno il tag di riferimento, ovvero quello con coordinate ideali, con cui confrontare i risultati della localizzazione.

Disattivarla rende non più disponibili le funzioni a esso collegate, come il calcolo degli errori.

- Tasto “START” : Questo pulsante fa partire l'applicazione. È disabilitato quando il programma è già attivo.
- Tasto “STOP” : Questo tasto invece blocca l'esecuzione del programma, è disabilitato quando il programma è già fermo.
- Tasto “All Versions” : La libreria che esegue la trilaterazione può contenere due algoritmi diversi, con questo tasto si possono mostrare i risultati di entrambi per poterli confrontare.
- Tasto “TEST” : permette di eseguire un file di test.
- Etichetta “Uploaded test” : indica il nome del file di test caricato tramite il pannello nella finestra Settings (mostrato precedentemente).
- Tasto “START/STOP SERVER” : Attiva o disattiva la funzione di server dell'applicazione (maggiori informazioni nel capitolo 6).

Rumore e Filtri:



Immagine 3.1.21 : Tasti per attivare e disattivare rumore e filtri.

Da questi tasti si può applicare del rumore al sistema simulato, e impostare dei filtri per attenuare questo rumore.

Le funzioni di applicazione del rumore e di filtraggio verranno esposte nel capitolo 5.

3.2 Funzioni di traslazione dei tag

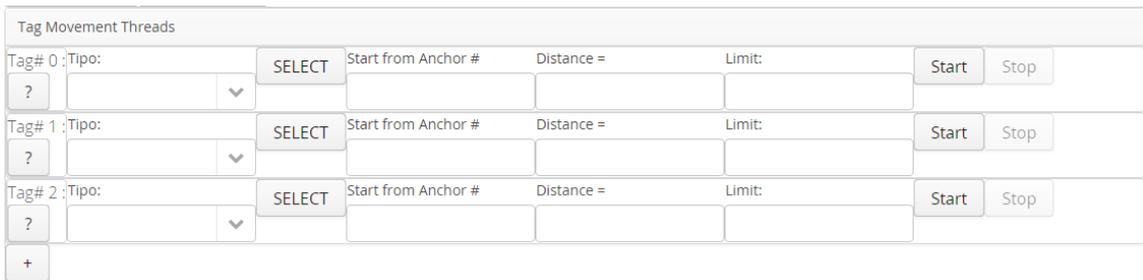
The screenshot shows a window titled 'Tag Movement Threads'. It contains three rows for configuring tag movements, labeled 'Tag# 0', 'Tag# 1', and 'Tag# 2'. Each row has a 'Tipo:' dropdown menu with a question mark icon, a 'SELECT' button, a 'Start from Anchor #' text input field, a 'Distance =' text input field, a 'Limit:' text input field, and 'Start' and 'Stop' buttons. Below the third row is a '+' button to add more tags.

Immagine 3.2.1 : Finestra di impostazione dei movimenti dei tag.

Per eseguire delle simulazioni significative è stato necessario creare delle funzioni di spostamento che permettessero di muovere i tag in maniera simile alla realtà.

In questa finestra ci sono dei pannelli, uno per ogni tag, che permettono di impostare un tipo di movimento e applicarlo al tag.

Quando un nuovo tag verrà inserito nel sistema si potrà attivare un nuovo pannello a esso collegato, cliccando il tasto “+”.

Per applicare uno spostamento si sceglie un tipo di movimento dal combobox, si preme il tasto “SELECT” e quel tipo di spostamento viene inizializzato.

A questo punto si inseriscono i dati richiesti nei campi e si preme il tasto “START” per farlo partire.

Si può fermare in qualsiasi momento premendo “STOP”.

Il tasto “?” fa comparire una notifica con informazioni sul tipo di spostamento selezionato.

Una volta fatto partire lo spostamento, se si vuole farlo terminare dopo un certo numero di passi, si può impostare questo numero nel campo Limit, presente nella maggiorparte dei movimenti. Impostando 0 o lasciando vuoto, lo spostamento continuerà finchè non si preme STOP.

Il tag effettuerà un passo a ogni localizzazione, e la distanza percorsa dal tag in un passo può essere modificata dal pannello delle impostazioni (*Immagine 3.1.17*).

Anchors Tour:

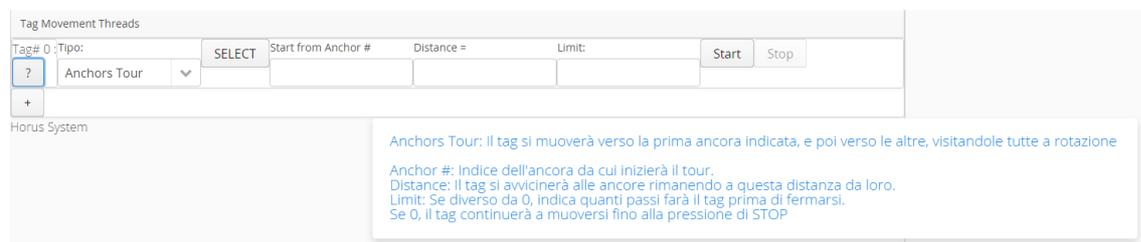


Immagine 3.2.2 : Schermata di aiuto per Anchors Tour.

Questo tipo di spostamento fa visitare al tag tutte le ancore.

Il tag si sposterà inizialmente verso la prima ancora da cui iniziare, ovvero quella specificata nel campo di testo “Start from Anchor#”, e si fermerà alla distanza specificata (nel campo “distance”) dall'ancora, dopodiché inizierà a spostarsi verso quella successiva, fermandosi sempre alla distanza impostata.

Se viene specificata distanza 0, il tag si sovrapporrà alle ancore.

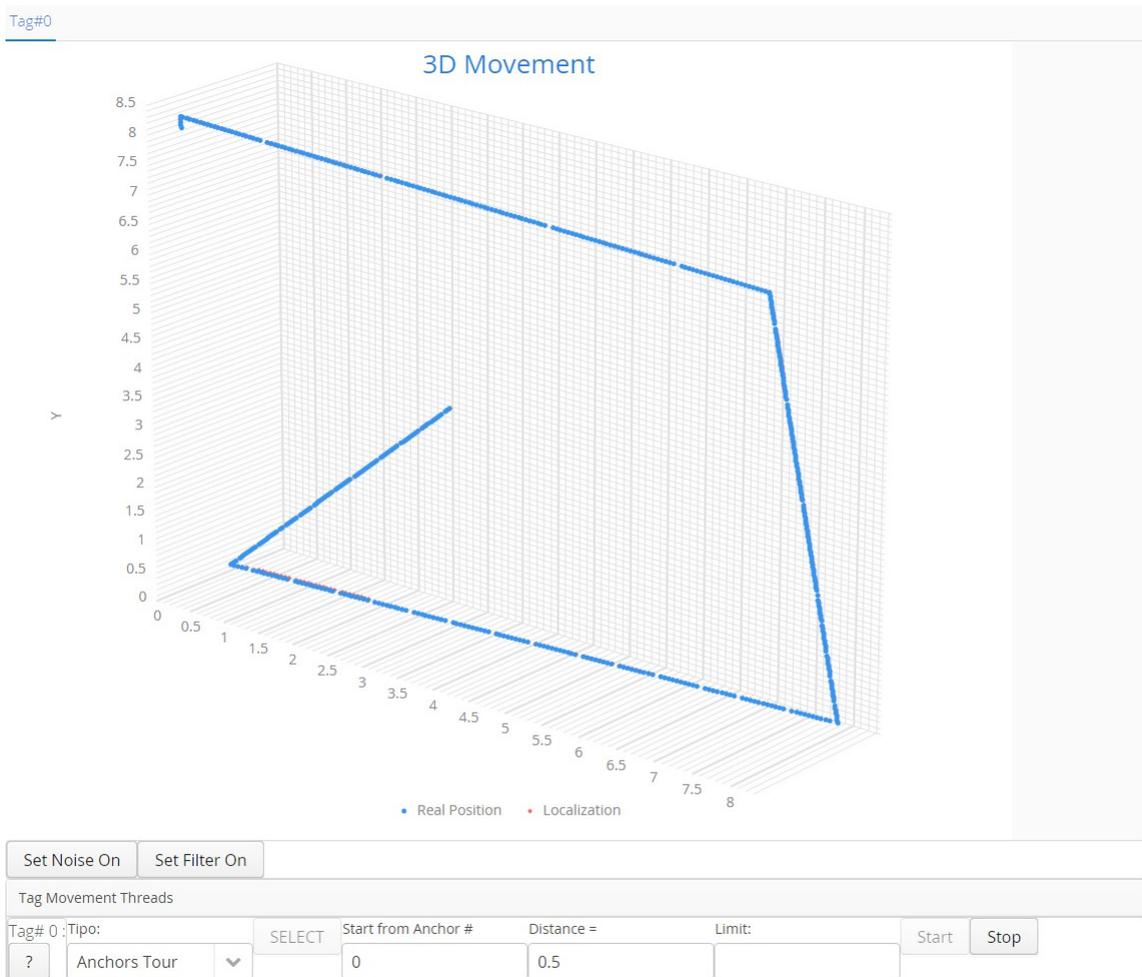


Immagine 3.2.3 : Esempio di Anchors Tour su grafico XYZ.

Fixed Z Tour:

Immagine 3.2.4 : Schermata di aiuto per Fixed Z Tour.

Esegue lo stesso spostamento di “Anchors Tour”, ma senza mai spostarsi sull'asse Z, rimanendo quindi sullo stesso piano.

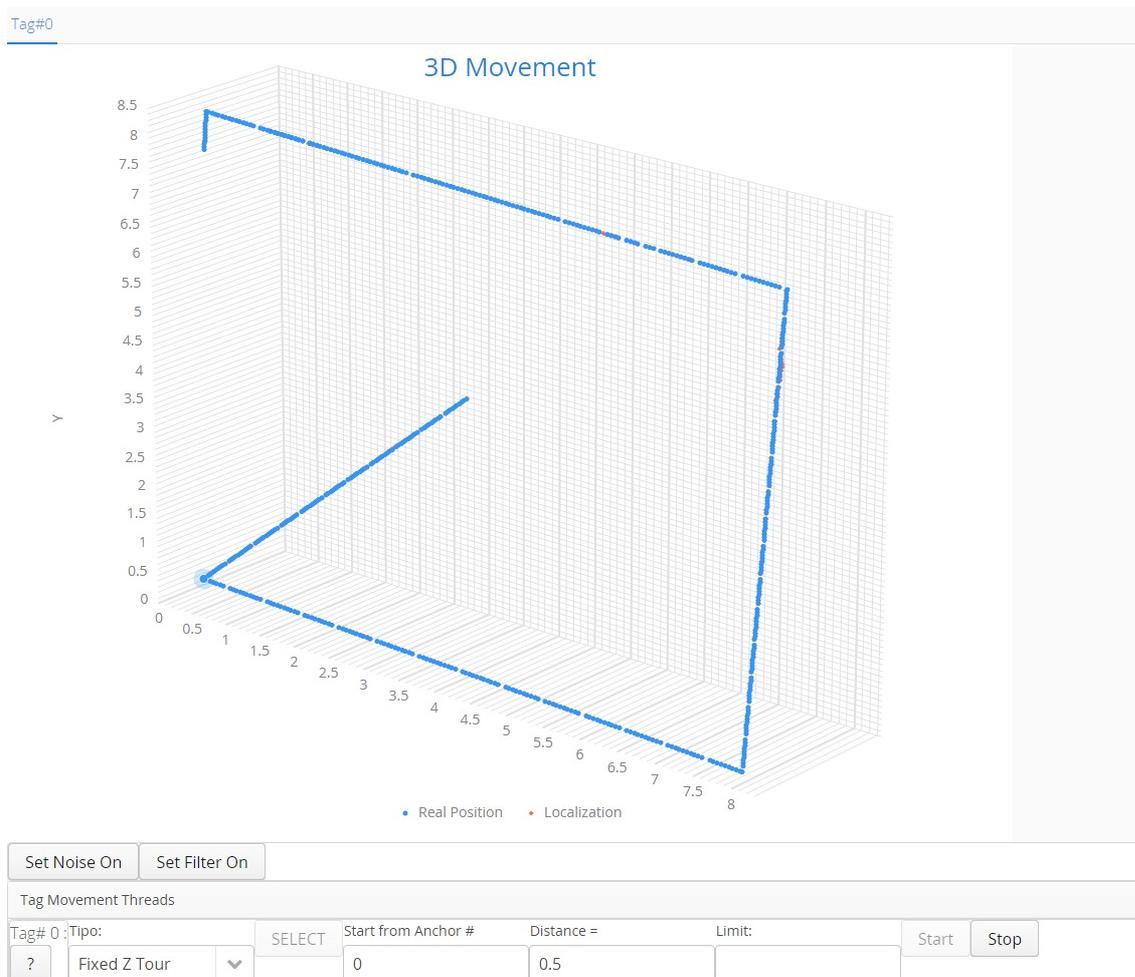


Immagine 3.2.5 : Esempio di Fixed Z Tour su grafico XYZ.

Circle:



Immagine 3.2.6 : Schermata di aiuto per Circle.

Il tag si sposta verso destra di una distanza pari al raggio del cerchio, inserito nel campo “Radius”, dopodiché seguirà una traiettoria circolare in senso antiorario.

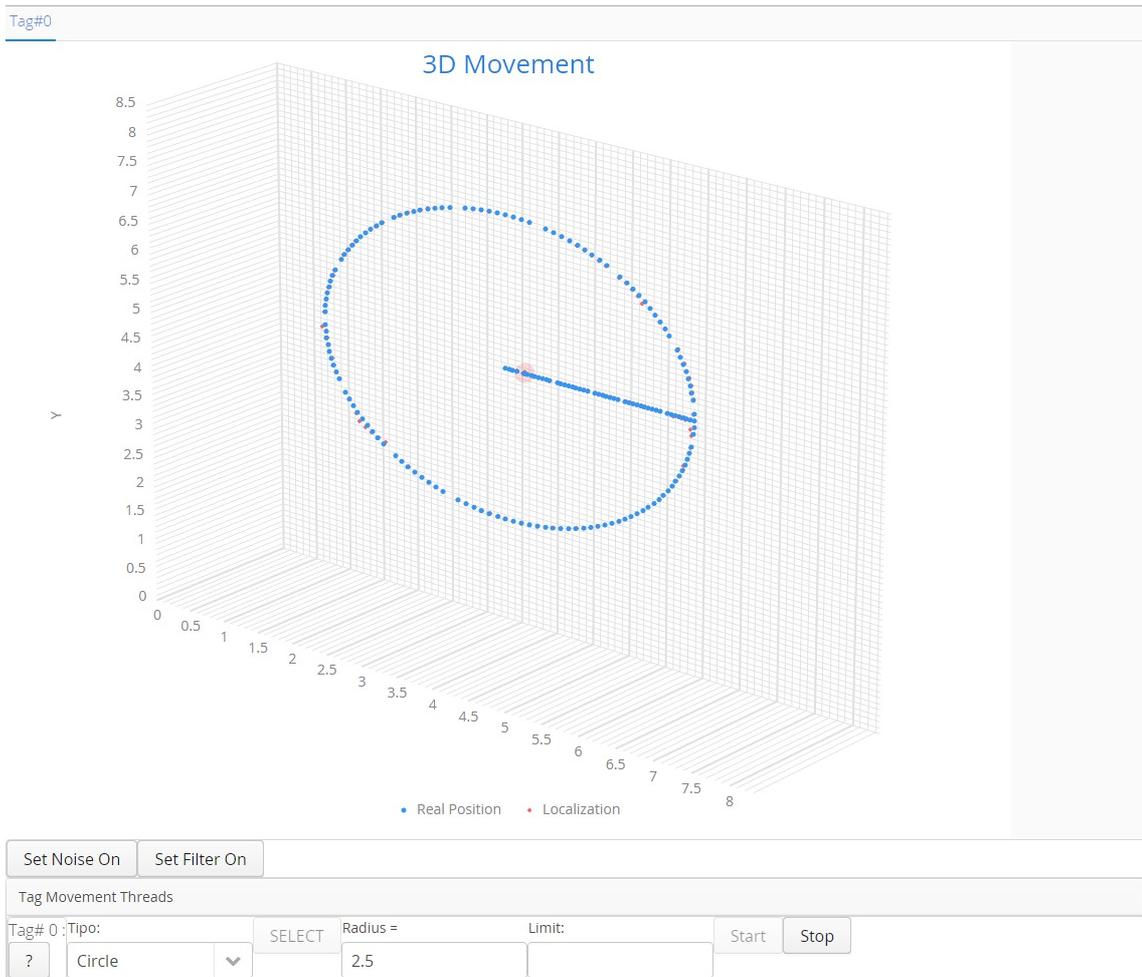


Immagine 3.2.7 : Esempio di Circle su grafico XYZ.

Move to:

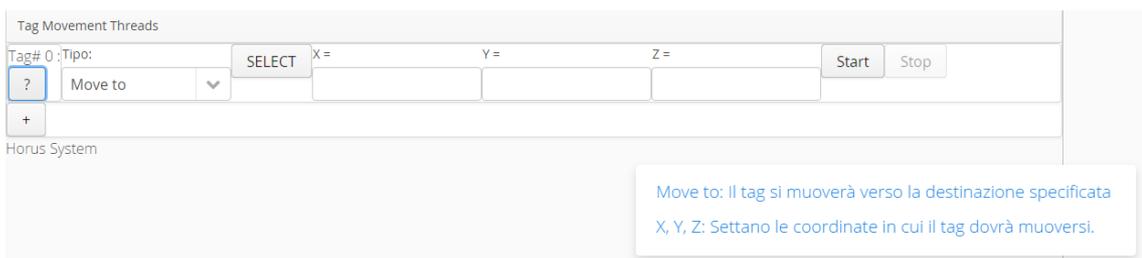


Immagine 3.2.8 : Schermata di aiuto per Move To.

Semplice spostamento dalla posizione attuale del tag verso quella specificata.

Il movimento si fermerà da solo una volta raggiunta la destinazione, non è presente il campo "Limit".

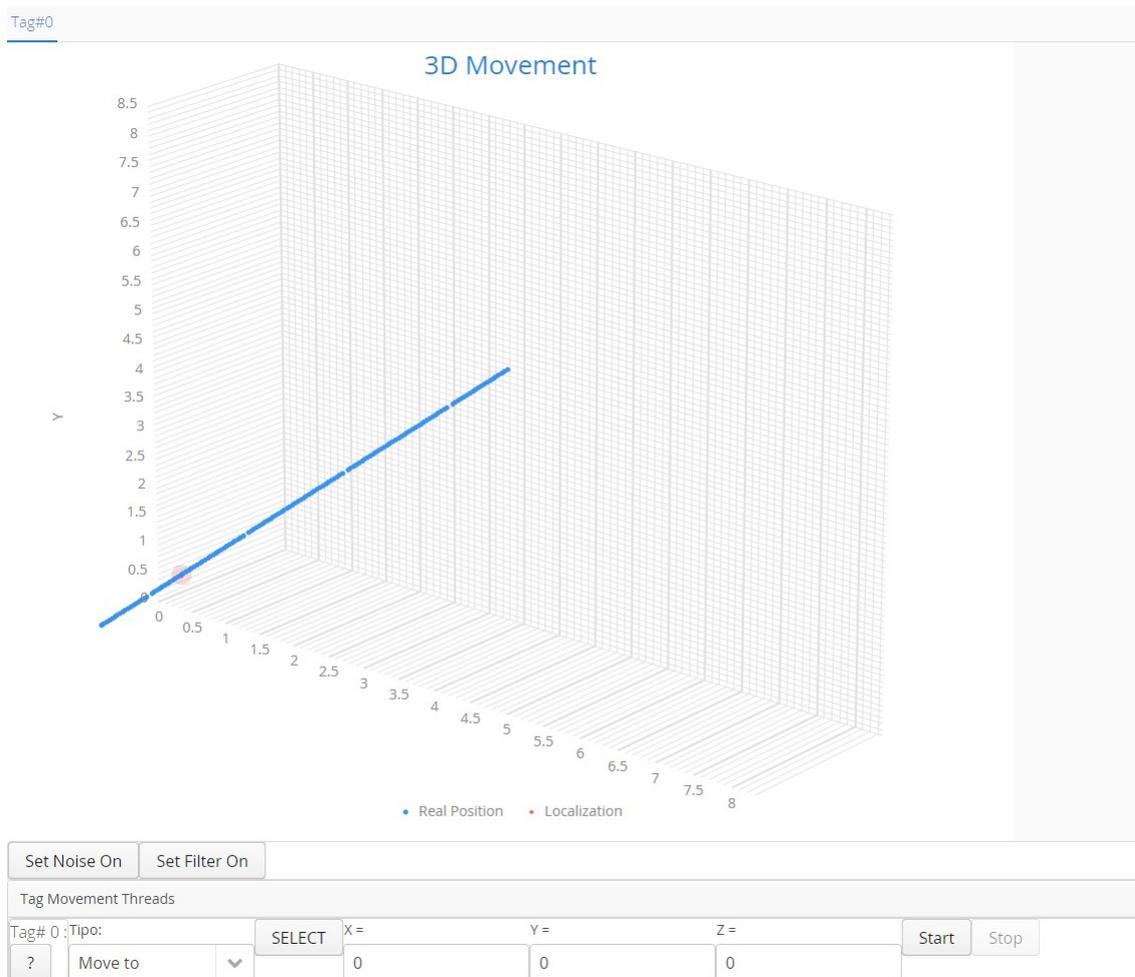


Immagine 3.2.9 : Esempio di Move To in grafico XYZ.

Z Sine:

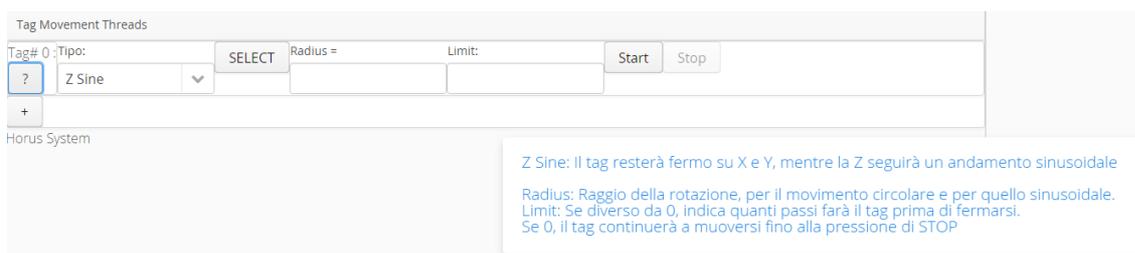


Immagine 3.2.10 : Schermata di aiuto per Z Sine.

Il tag resta fermo sulle coordinate X e Y, mentre salirà e scenderà sulla coordinata Z seguendo un andamento sinusoidale, ovvero seguendo la formula:

$$z = z + \left(radius * \sin \left(degree * \left(\frac{\pi}{180} \right) \right) \right)$$

dove *degree* è l'ampiezza in gradi dell'angolo, che viene incrementato a ogni ciclo, quindi a ogni passo, con modulo 360.

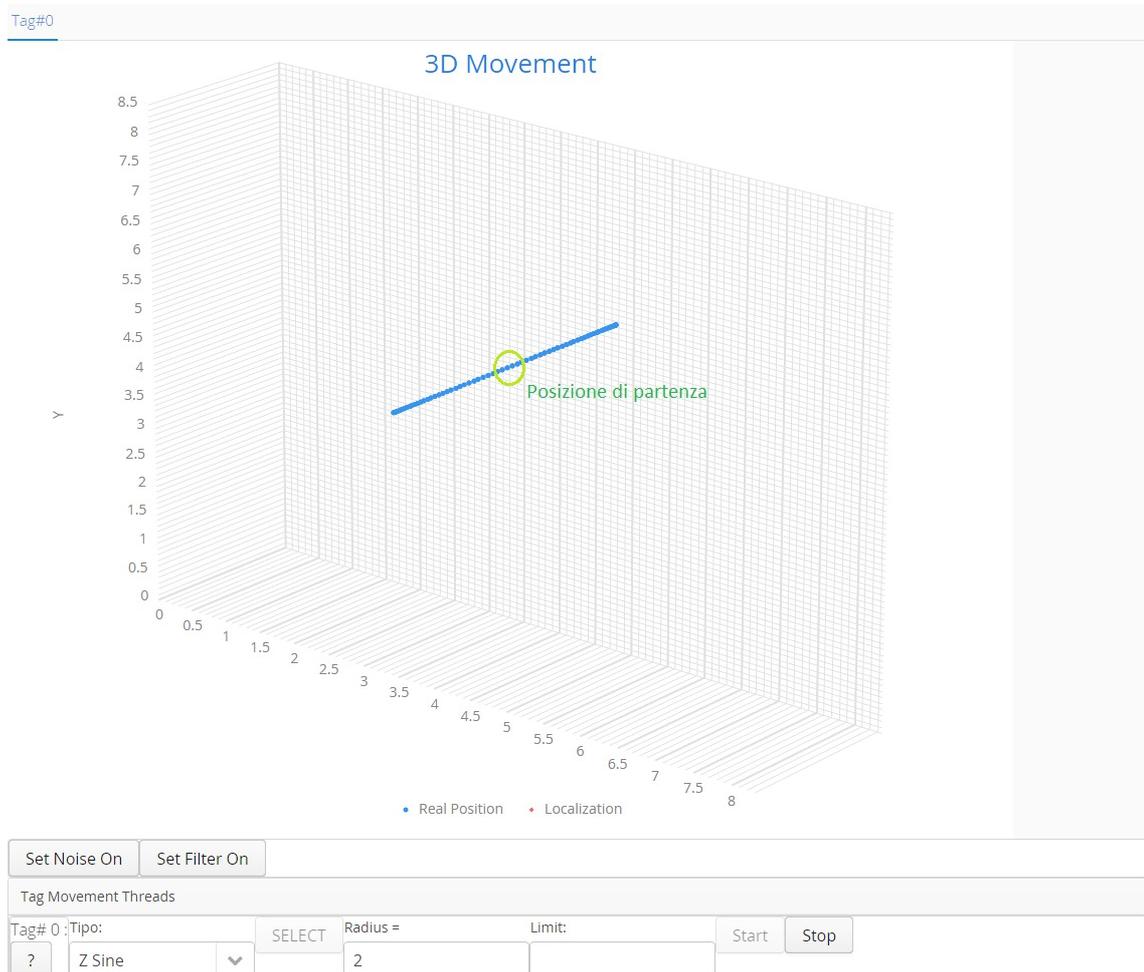


Immagine 3.2.11 : Esempio di Z Sine su grafico XYZ.

Full coverage:

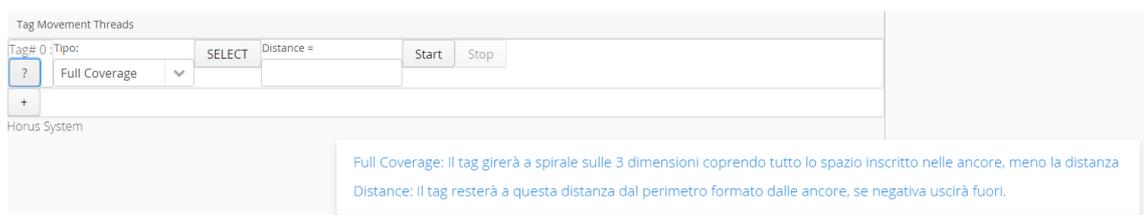


Immagine 3.2.12 : Schermata di aiuto per Full Coverage.

Questo tipo di spostamento copre tutto lo spazio compreso tra le ancore, spostandosi a turno su ognuna delle coordinate per una distanza che incrementa a ogni turno. Il valore di distanza indica quanto lo spazio da coprire dista da quello effettivamente

cirscritto dalle ancore; nel caso sia negativo, significa che coprirà uno spazio più grande di quello delle ancore.

Questo tipo di spostamento è utile perchè coprendo tutto lo spazio si può cercare di capire quali zone sono più sensibili a errori di posizione rispetto ad altre.

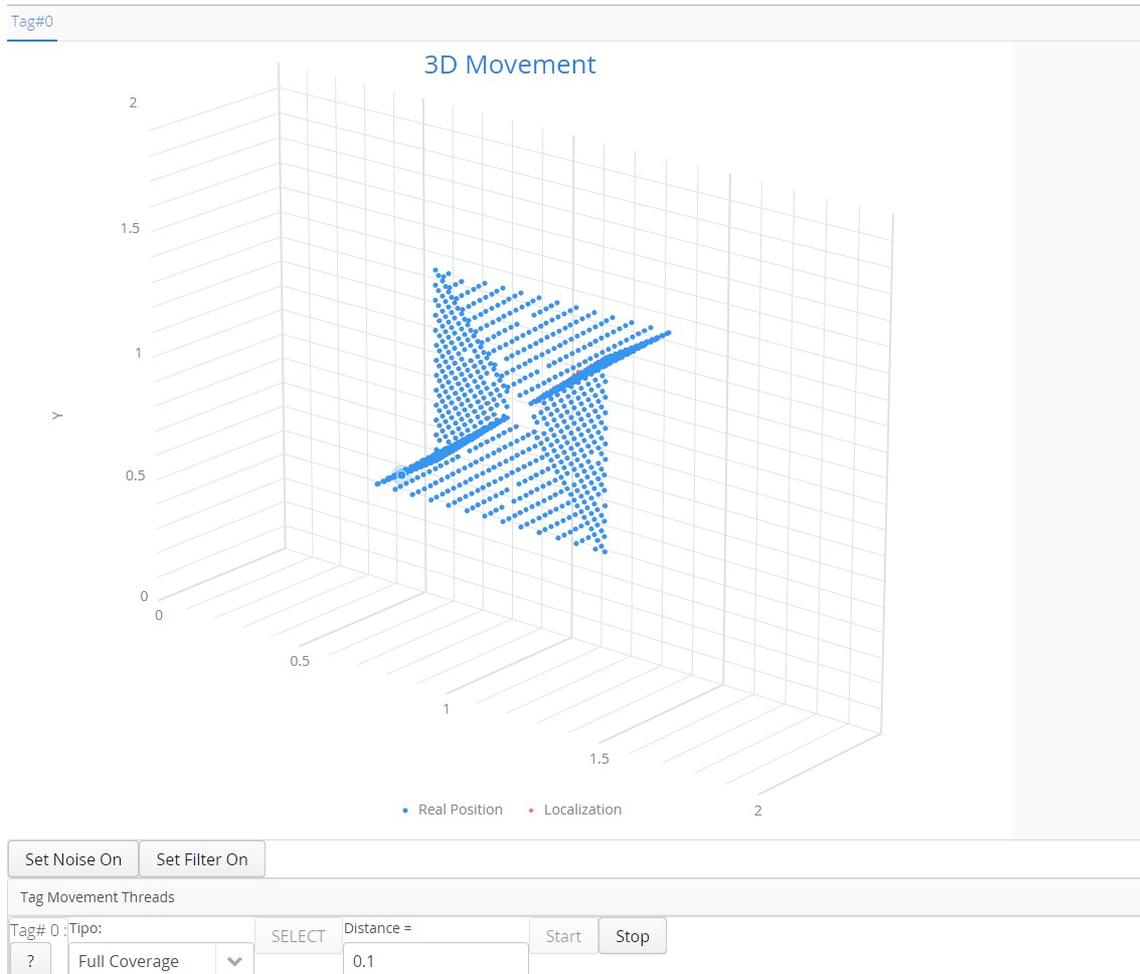


Immagine 3.2.13 : Esempio di Full Coverage su grafico XYZ.

3.3 Vaadin

Il programma è una applicazione web che gira su un server locale gestito da Apache Tomcat, un software open source per l'implementazione di *Java Servlet*, *JavaServer Pages*, *Java Expression Language* e *Java WebSocket*^[10].

Il software è stato progettato come applicazione web per utilizzare, per l'interfaccia grafica, il framework vaadin.

Il framework vaadin è un progetto open source sotto licenza Apache che fornisce componenti grafici lato server (a volte anche comprendenti alcuni elementi client-side) in java, per la costruzione di web application^[11].

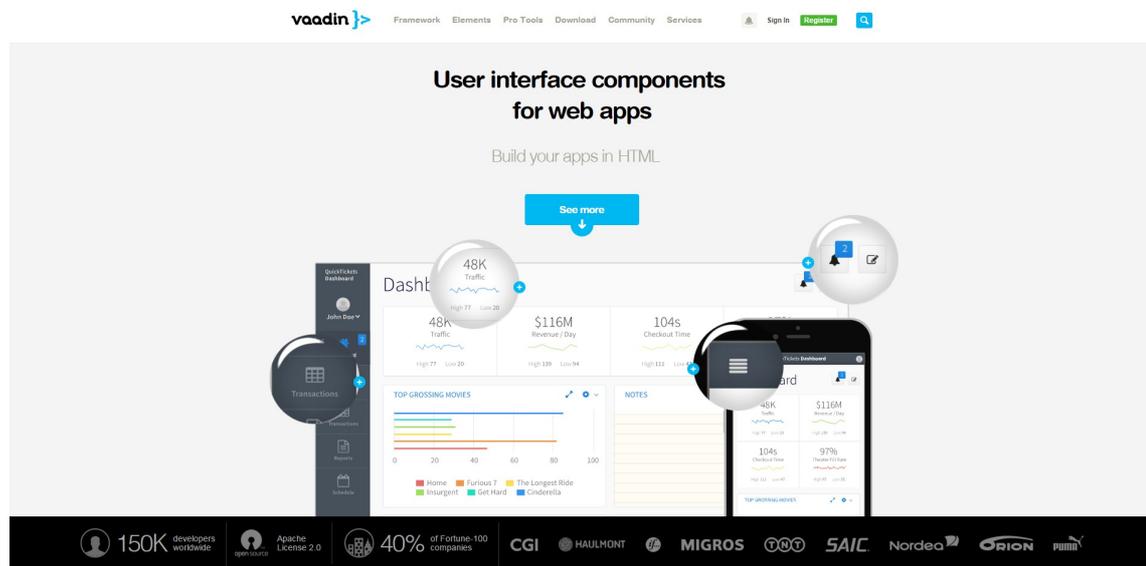


Immagine 3.3.1 : Home page di Vaadin.

A differenza delle tecnologie JavaScript, l'architettura Vaadin è a lato server, ovvero la maggior parte della logica viene eseguita su server; per i componenti lato client si basa sul Google Web Toolkit (GWT).

Per la visualizzazione lato browser utilizza la tecnologia Ajax^[12], questo permette di avere componenti grafici reattivi a prescindere dalla comunicazione col server.

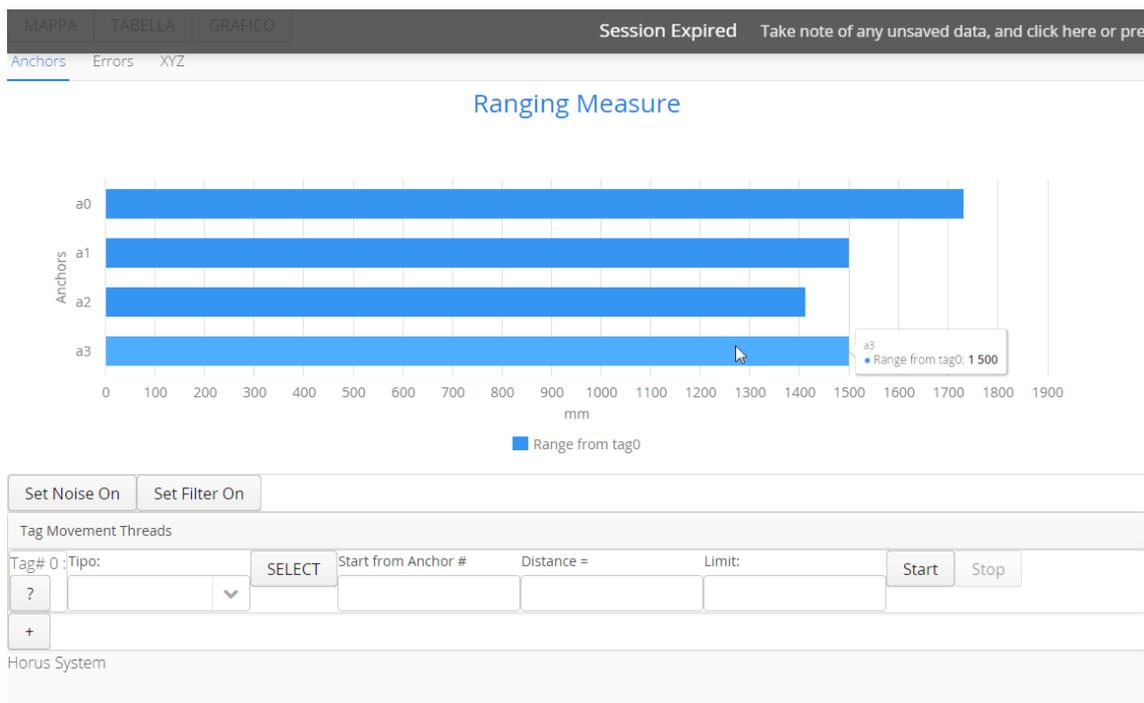


Immagine 3.3.2 : Possibilità di selezionare i dati del grafico a server spento.

Vaadin nasce come estensione del Web framework open source Millstone, con il nome di IT Mill Toolkit.

Nel 2007 Michael Widenius, principale autore di MySQL, investe in questa tecnologia che nel 2008 cambia nome in Vaadin Framework.

Successivamente viene creata la piattaforma web di distribuzione e il framework viene espanso con vari add-on, gratuiti e a pagamento.

L'Add-On Vaadin Charts, utilizzato in questa applicazione per disegnare i grafici è sotto licenza a pagamento.

Principali caratteristiche^[13]:

- Vasta gamma di componenti per l'interfaccia utente.
- Widget interattivi.
- Supporto a touch e drag&drop.
- Layout strutturati in java o in html o entrambi.
- Supporto al pattern Model View Controller per la gestione dei dati.
- Creazione di nuovi componenti tramite composizione e ereditarietà.
- Numerosi add-on disponibili dal sito internet.

- Compatibilità con CSS per cambiare lo stile grafico.
- Temi grafici integrati.
- Gestione lato server dell'interfaccia utente.
- Supporto a SSL
- Plugin per i principali ambienti di sviluppo come Eclipse e NetBeans.
- Supporto ai principali Browser.
- Lato client basato su GWT.

L'ultima release stabile di Vaadin è la 7.6.1 del 13 Gennaio 2016.

4 Principali funzionalità

Come anticipato nel capitolo precedente, le due principali funzionalità del programma sono quelle di poter confrontare due algoritmi eseguendo due trilaterazioni diverse a ogni localizzazione e la possibilità di riprodurre test effettuati nella realtà leggendo file di log appositi.

L'applicazione è stata utilizzata per studiare gli errori che rendono meno precisa la localizzazione.

Ci sono principalmente due tipologie di errore:

1. Errori di posizione:

Sono imprecisioni nel calcolo della posizione.

Sono dovuti principalmente ad approssimazioni dei dati.

Generalmente sono errori molto piccoli.

2. Errori di ranging:

Sono errori nella misura della distanza tra ancora e tag.

Possono dipendere da vari fattori, anche esterni e possono essere anche molto grandi.

```
193 /*1.5, 1.7, 2.0 */
194 char str1[] = "ma00 t00 00000A9E 00000A9E 01E4 E4 0006B1AC 4034 4034 a0";
195 char str2[] = "ma01 t00 00000BF8 00000BF8 01E4 E4 0006B1AC 4034 4034 a0";
196 char str3[] = "ma02 t00 00000D69 00000D69 01E4 E4 0006B1AB 4034 4034 a0";
197 char str4[] = "ma03 t00 00000AE7 00000AE7 01E4 E4 0006B1AB 4034 4034 a0";
198 /* */
```

Ranging reports

```
<terminated> (exit value: 0) HorvsServerLite.exe [C/C++ Application] C:\Users\Daniele\Documents\workspace C\HorvsServerLite\Debug\HorvsServerLite.exe (19/02/16, 16:10)
```

0 Tag position: [1.500303, 1.700326, 2.001499]

1 Tag position: [1.500169, 1.699971, 2.000961]

Real position: 1.5, 1.7, 2.0

```
198 /*_with_noise_*/
199 char str1[] = "ma00 t00 00000A8F 00000A8F 13FD FD 000AC230 4034 4034 a0";
200 char str2[] = "ma01 t00 00000C16 00000C16 13FD FD 000AC230 4034 4034 a0";
201 char str3[] = "ma02 t00 00000D6E 00000D6E 13FD FD 000AC22F 4034 4034 a0";
202 char str4[] = "ma03 t00 00000AB6 00000AB6 13FD FD 000AC22F 4034 4034 a0";
```

Ranging with noise

```
<terminated> (exit value: 0) HorvsServerLite.exe [C/C++ Application] C:\Users\Daniele\Documents\workspace C\Horv
```

0 Tag position: [1.472279, 1.719124, 2.022429]

1 Tag position: [1.472279, 1.719124, 2.022429]

Position result with input noise

Immagine 4.0.1 : Confronto di risultati con e senza errori di ranging.

4.1 Confronto di due algoritmi

Utilizzando valori ideali, senza alcun tipo di rumore o inesattezza, l'applicazione permette di vedere soltanto gli errori di posizione.

Questo permette di verificare la precisione di un algoritmo in condizioni ideali e confrontarlo con altri.

Scopo dell'applicazione:

Si è reso necessario sviluppare questa applicazione per poter constatare se le modifiche apportate alla funzione di trilaterazione portassero delle migliorie.

Infatti avviare la trilaterazione con un input fisso non dava risultati significativi, in quanto il risultato può dipendere da diversi fattori.

Una versione dell'algoritmo può dare un risultato migliore in certe posizioni, ma risultare peggiore nella media, oppure avere imprecisioni su alcune coordinate più che su altre.

Tramite questa applicazione è possibile generare un flusso di dati in input all'algoritmo per verificarne il comportamento in diverse situazioni e poter anche tracciare dei grafici.

Per confrontare due algoritmi, o due versioni dello stesso, è necessario compilare la libreria (.dll in sistemi Windows) con il codice dei due algoritmi e inserirla nella directory da cui l'applicazione la caricherà.

Avviata il programma si può inizializzare il tag in coordinate specifiche; nel caso in cui sia inserito nel file di configurazione verrà inizializzato un tag con le coordinate inserite in tale file.

Per avere un confronto tra i due algoritmi si deve abilitare la visualizzazione di entrambi dall'apposito tasto; inoltre si può abilitare anche la visualizzazione del tag di riferimento, ovvero quello nella posizione esatta, che viene disegnato sulla mappa nelle posizioni ideali, inserite dall'utente; in questo modo si può osservare anche quanto gli algoritmi variano rispetto alla realtà, ovvero quanto è l'errore di posizione.

Durante l'esecuzione l'applicazione interrogherà la libreria contenente l'algoritmo per due volte a ogni localizzazione, specificando, con un parametro passato alla funzione nativa, quale versione dell'algoritmo utilizzare.

Al momento l'applicazione è strutturata per considerare due versioni.

Il numero di versione indica quale algoritmo utilizzare tra i due presenti nella libreria, che possono essere algoritmi diversi o due versioni dello stesso, come detto in precedenza.

L'applicazione è composta da un thread principale, specificato in una classe Java di tipo Runnable (che implementa quindi l'interfaccia Runnable).

Il corpo di questo thread consiste nella chiamata alla funzione nativa e, dopo aver ricevuto i risultati di localizzazione da essa, nell'aggiornamento della grafica con i nuovi dati ottenuti.

Queste operazioni sono contenute in un ciclo *while* e vengono ripetute ogni 100 millisecondi, tramite la funzione *sleep* dei thread java, che mette in pausa l'esecuzione per 100 millisecondi. Questo tempo è comunque modificabile dal file di configurazione. L'esecuzione del ciclo viene interrotta premendo il tasto "STOP", che setta la variabile che forma il predicato del *while*.

Tramite le visualizzazioni mappa e tabella, l'utente può quindi visualizzare sull'interfaccia i risultati della trilaterazione.

A questo punto è necessario muovere il tag e farlo spostare lungo la mappa, per ottenere dei dati che testino al meglio l'algoritmo.

Per farlo vengono utilizzati gli spostamenti preimpostati descritti nel paragrafo 3.2.

Un esempio di test che si può effettuare è impostare un certo tipo di movimento e lasciarlo eseguire per un determinato periodo di tempo, osservando l'errore medio nell'apposita tabella (*Immagine 3.1.6*).

Un altro test possibile è la verifica della resistenza al rumore.

Alcune funzioni dell'algoritmo, come quella relativa all'incremento dei range in caso di fallita trilaterazione, servono esclusivamente ad aumentare la resistenza agli errori.

Per verificarla si può applicare un rumore artificiale, tramite gli appositi tasti (*Immagine 3.1.21*), di tipo uniforme o gaussiano.

In questo caso il tag può anche rimanere fermo visto che il rumore cambierà continuamente la posizione localizzata, e si potrà notare sempre l'errore medio nella tabella, per verificare quale dei due abbia il valore minore.


```
Report.txt - Blocco note
File Modifica Formato Visualizza ?
ma00 t00 00000A8F 00000A8F 13FD FD 000AC230 4034 4034 a0
ma01 t00 00000C16 00000C16 13FD FD 000AC230 4034 4034 a0
ma02 t00 00000D6E 00000D6E 13FD FD 000AC22F 4034 4034 a0
ma03 t00 00000AB6 00000AB6 13FD FD 000AC22F 4034 4034 a0

Without noise:
ma00 t00 00000A9E 00000A9E 13FE FE 000AC230 4034 4034 a0
ma01 t00 00000BF8 00000BF8 13FE FE 000AC230 4034 4034 a0
ma02 t00 00000D69 00000D69 13FE FE 000AC22F 4034 4034 a0
ma03 t00 00000AE7 00000AE7 13FE FE 000AC22F 4034 4034 a0

Range Error:
Clear: 00000A9E
Noisy: 00000A8F
Difference: -15
Clear: 00000BF8
Noisy: 00000C16
Difference: 30
Clear: 00000D69
Noisy: 00000D6E
Difference: 5
Clear: 00000AE7
Noisy: 00000AB6
Difference: -49
```

Immagine 4.1.2 : Esempio di log di report con rumore.

- **Log degli errori di ranging:**

Questo log viene stampato durante la funzione di test di log reali (paragrafo 4.2) e contiene, per ogni posizione, l'errore di ranging di una determinata ancora.

Vengono stampati quattro file ogni volta, uno per ogni ancora.

I dati sono rappresentati nel formato:

id ancora; range effettivo; range misurato; differenza.

rangeError_a0.txt - Blocco note	rangeError_a1.txt - Blocco note	rangeError_a2.txt - Blocco note	rangeError_a3.txt - Blocco note
File Modifica Formato Visualizza ?	File Modifica Formato Visualizza ?	File Modifica Formato Visualizza ?	File Modifica Formato Visualizza ?
0;0000069D;000006fc;95 0;0000069D;00000709;108 0;0000069D;00000711;116 0;0000069D;000006f1;84 0;0000069D;000006e5;72 0;0000069D;000006e3;70 0;0000069D;000006cd;48 0;0000069D;000006df;66 0;0000069D;000006d1;52 0;0000069D;000006db;62 0;0000069D;000006e5;72 0;0000069D;000006d8;59 0;0000069D;000006d3;54 0;0000069D;000006da;61 0;0000069D;000006cf;50 0;0000069D;000006da;61 0;0000069D;000006d1;52 0;0000069D;000006d2;53 0;0000069D;000006cc;47 0;0000069D;000006c6;41 0;0000069D;000006d2;53 0;0000069D;000006dc;63 0;0000069D;000006c7;42 0;0000069D;000006f7;90	1;0000069D;000006ec;79 1;0000069D;000006df;66 1;0000069D;000006da;61 1;0000069D;000006f4;87 1;0000069D;000006f5;88 1;0000069D;000006d2;53 1;0000069D;000006db;62 1;0000069D;000006f7;90 1;0000069D;000006d6;57 1;0000069D;000006d3;54 1;0000069D;000006ee;81 1;0000069D;000006e9;76 1;0000069D;000006d3;54 1;0000069D;000006e3;70 1;0000069D;00000701;100 1;0000069D;000006db;62 1;0000069D;000006d8;59 1;0000069D;000006e5;72 1;0000069D;000006fb;94 1;0000069D;000006da;61 1;0000069D;000006ed;80 1;0000069D;00000705;104 1;0000069D;000006d8;59 1;0000069D;000006ca;45	2;0000069D;000006b0;19 2;0000069D;00000662;-59 2;0000069D;000006ef;82 2;0000069D;000006fe;97 2;0000069D;000006bc;31 2;0000069D;00000694;-9 2;0000069D;00000669;-52 2;0000069D;000006ad;16 2;0000069D;000006cd;48 2;0000069D;0000067c;-33 2;0000069D;0000069d;0 2;0000069D;000006d1;52 2;0000069D;000006ab;14 2;0000069D;000006a9;12 2;0000069D;00000672;-43 2;0000069D;0000069e;1 2;0000069D;000006be;33 2;0000069D;0000065e;-63 2;0000069D;00000716;121 2;0000069D;00000672;-43 2;0000069D;000006a6;9 2;0000069D;00000698;-5 2;0000069D;00000657;-70 2;0000069D;000006ed;80	3;0000069D;0000068d;-16 3;0000069D;000006b2;21 3;0000069D;0000069e;1 3;0000069D;00000678;-37 3;0000069D;0000064e;-79 3;0000069D;000006b1;20 3;0000069D;00000695;-8 3;0000069D;0000065e;-63 3;0000069D;000006b8;27 3;0000069D;00000691;-12 3;0000069D;00000682;-27 3;0000069D;0000069d;0 3;0000069D;0000069a;-3 3;0000069D;0000066f;-46 3;0000069D;0000068a;-19 3;0000069D;0000068e;-15 3;0000069D;0000068e;-15 3;0000069D;00000692;-11 3;0000069D;00000690;-13 3;0000069D;0000067c;-33 3;0000069D;000006b6;25 3;0000069D;00000697;-6 3;0000069D;0000066d;-48 3;0000069D;00000660;-61

Immagine 4.1.3 : Esempio di log degli errori di ranging.

Quali log stampare e quali no, si può decidere dal pannello settings (Immagine 3.1.15).

Questi file di testo possono essere utilizzati ad esempio per ottenere dei grafici, tramite diversi programmi come Matlab o Excel.

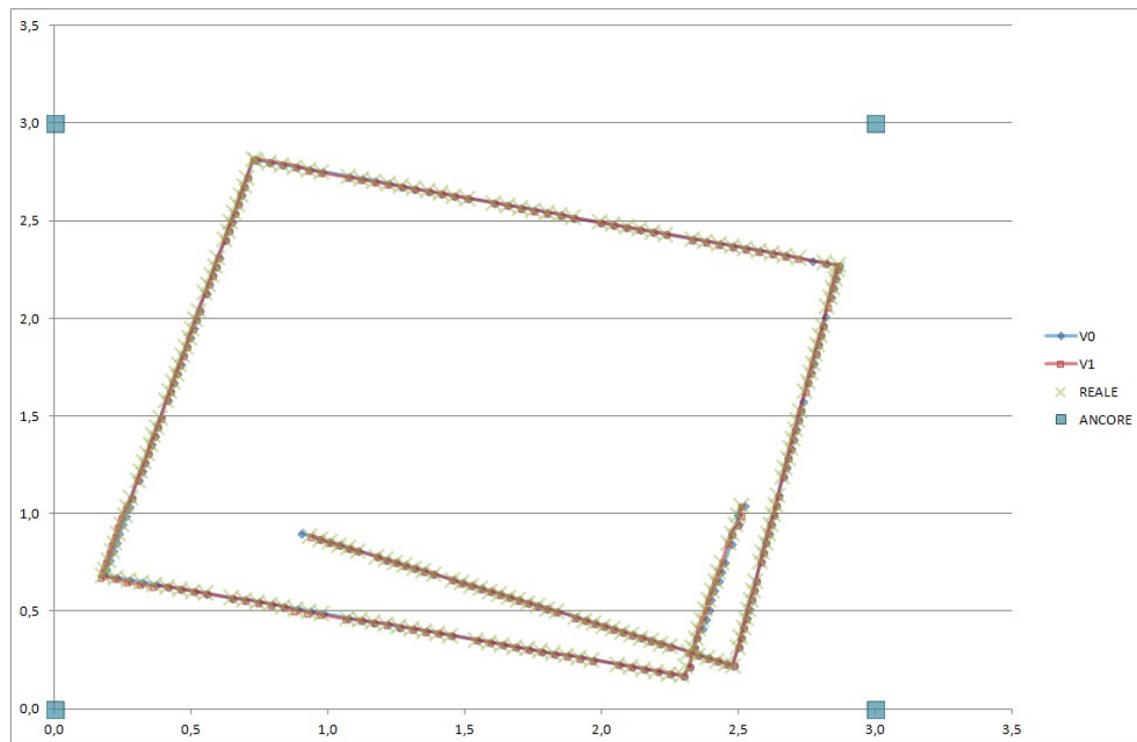


Immagine 4.1.4 : Esempio di grafico XY fatto con Excel.

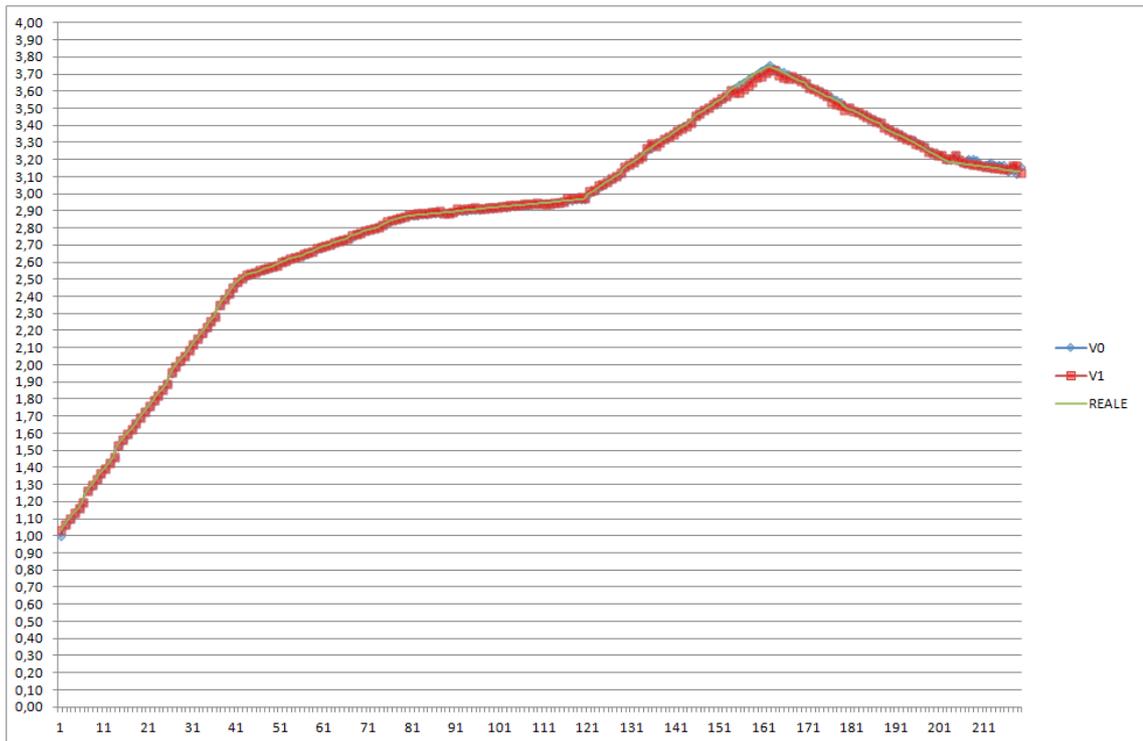


Immagine 4.1.5 : Esempio di grafico della coordinata Z, fatto con Excel.

I grafici di figura 4.1.4 e 4.1.5 sono stati creati tramite Excel, caricando i log stampati dall'applicazione.

In questo caso è stato effettuato un confronto tra due versioni dell'algorithmo:

1. **versione 0:** libreria originale della DecaWave. Si limita al calcolo della trilaterazione senza eseguire il controllo della gdop. L'unica modifica apportata è stata cambiare il valore di incremento dei raggi nel caso di errore della trilaterazione (spiegato in paragrafo 2.4).
2. **versione 1:** versione con tutte le modifiche discusse nel paragrafo 2.4, con fase di refinement composta dal calcolo della gdop.

Il test da cui sono stati ricavati i grafici è stato fatto in assenza di rumore, quindi con solo gli errori di posizione, che sono molto bassi, per cui dal grafico non si nota una grande differenza tra i due algoritmi che forniscono entrambi dei risultati molto fedeli alle posizioni reali.

I grafici sul confronto diventano più significativi nei test di resistenza al rumore, in cui si applica un errore sui range.

Infatti vista l'ottima precisione dell'algorithm in condizioni ideali (con errori di pochi millimetri) risulta più significativo concentrarsi sul comportamento in presenza di errori, che sarà poi la situazione nella realtà.

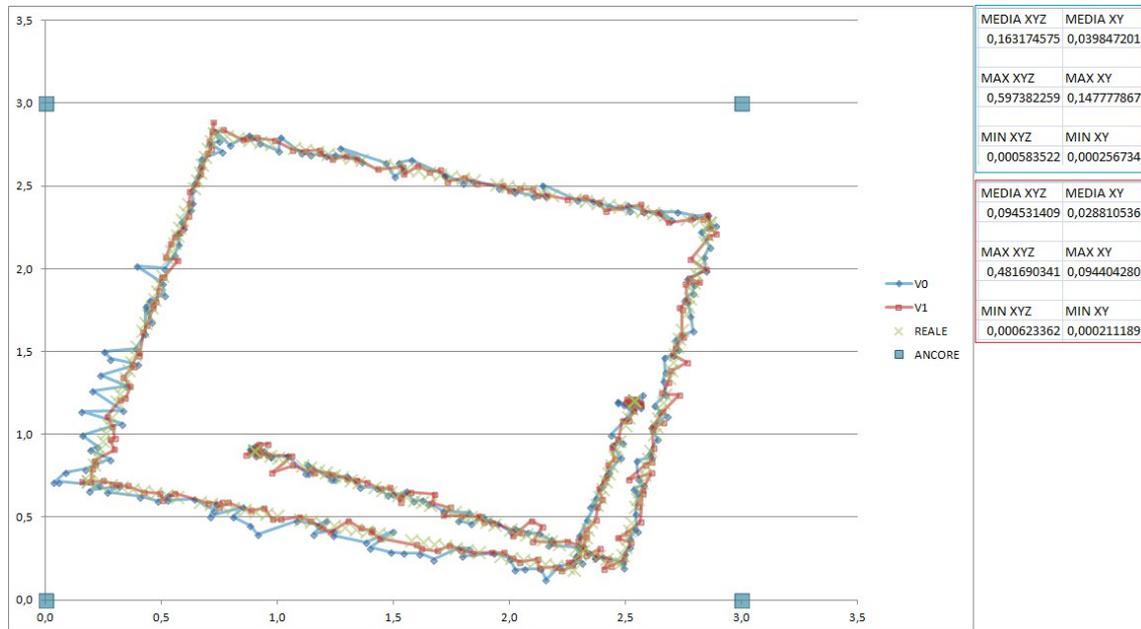


Immagine 4.1.6 : Esempio di grafico XY con rumore.

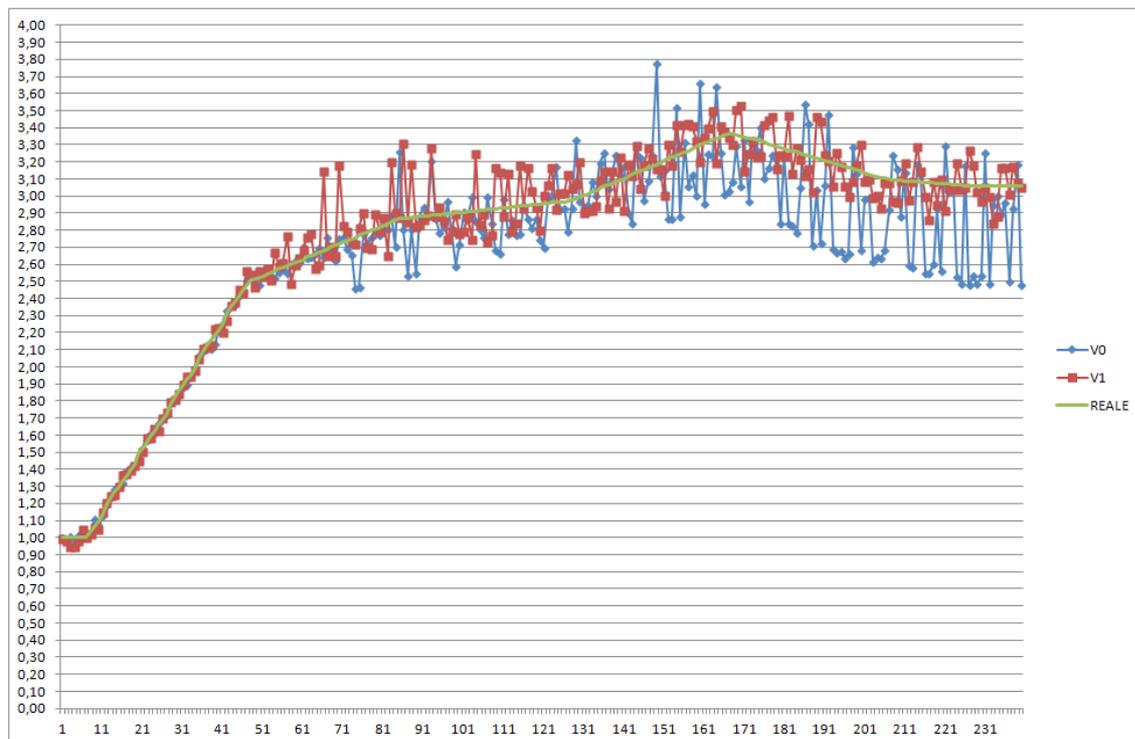


Immagine 4.1.7 : Esempio di grafico Z con rumore.

Nelle immagini 4.1.6 e 4.1.7 sono presenti i grafici dello stesso test ma con un rumore gaussiano applicato.

In immagine 4.1.6 sono presenti anche i valori numerici degli errori, minimi, massimi e della media; per entrambe le versioni, sempre ricavati tramite excel.

4.2 Analisi di log reali

La seconda funzione di questo programma è la possibilità di riprodurre test effettuati nella realtà tramite la lettura di file di log.

Avviando il sistema reale e facendolo funzionare si ottengono dei log contenenti le posizioni ricevute dalle ancore, che possono essere lette dal programma di simulazione per riprodurle.

```
Enter IP address of the master anchor or "b" to break
Switching to address 192.168.80.2
Attempt to read data from anchor 192.168.80.2
ma0000 t0000 00001cf0 00001cf0 62cd 88 0000edc1 4034 4034 a0000
ma0001 t0000 00001b7a 00001b70 62ce 88 0000edc3 4034 4034 a0000
ma0002 t0000 00001213 0000119b 62cf 88 0000edc5 4034 4034 a0000
ma0003 t0000 00001524 000014d4 62d0 88 0000edc6 4034 4034 a0000
ma0000 t0000 00001d01 00001d01 62d1 89 0000ee01 4034 4034 a0000
ma0001 t0000 00001b84 00001b7a 62d2 89 0000ee03 4034 4034 a0000
ma0002 t0000 00001232 000011ba 62d3 89 0000ee05 4034 4034 a0000
ma0003 t0000 0000153a 000014ea 62d4 89 0000ee06 4034 4034 a0000
ma0000 t0000 00001cd7 00001cd7 62d5 8a 0000ee41 4034 4034 a0000
ma0001 t0000 00001b89 00001b7f 62d6 8a 0000ee43 4034 4034 a0000
ma0002 t0000 0000123a 000011c2 62d7 8a 0000ee45 4034 4034 a0000
ma0003 t0000 00001519 000014c9 62d8 8a 0000ee46 4034 4034 a0000
ma0000 t0000 00001cd8 00001cd8 62d9 8b 0000ee81 4034 4034 a0000
ma0001 t0000 00001b9d 00001b93 62da 8b 0000ee83 4034 4034 a0000
ma0002 t0000 00001244 000011cc 62db 8b 0000ee85 4034 4034 a0000
ma0003 t0000 00001564 00001514 62dc 8b 0000ee86 4034 4034 a0000
ma0000 t0000 00001cd8 00001cd8 62dd 8c 0000eec1 4034 4034 a0000
ma0001 t0000 00001b38 00001b2e 62de 8c 0000eec3 4034 4034 a0000
ma0002 t0000 00001242 000011ca 62df 8c 0000eec5 4034 4034 a0000
ma0003 t0000 00001545 000014f5 62e0 8c 0000eec6 4034 4034 a0000
ma0000 t0000 00001cf5 00001cf5 62e1 8d 0000ef01 4034 4034 a0000
ma0001 t0000 00001b5c 00001b52 62e2 8d 0000ef03 4034 4034 a0000
ma0002 t0000 0000121d 000011a5 62e3 8d 0000ef05 4034 4034 a0000
ma0003 t0000 00001572 00001522 62e4 8d 0000ef06 4034 4034 a0000
ma0000 t0000 00001ceb 00001ceb 62e5 8e 0000ef41 4034 4034 a0000
ma0001 t0000 00001b13 00001b09 62e6 8e 0000ef43 4034 4034 a0000
```

Immagine 4.2.1 : Esempio di log.

I log contengono principalmente i report di ranging delle ancore, composti nel seguente formato^[14]:

m : messaggio.

aID : id dell'ancora, di quattro cifre esadecimali.

tID : id del tag, sempre di quattro cifre.

range corretto : valore del range in 32 bit, con correzione dell'inclinazione, valore memorizzato nel firmware dell'ancora.

range grezzo : valore non corretto del range.

numero di range : numero del range misurato, valore incrementale, di 16 bit.

numero di sequenza : numero di sequenza, di 8 bit, in modulo 256. La sequenza indica una localizzazione, i range di diverse ancore con lo stesso numero di sequenza si riferiscono alla stessa localizzazione.

tempo : tempo di fine della trasmissione tra ancora e tag per il calcolo del range, in 32 bit.

txad : Ritardo dell'antenna in trasmissione.

rxad : Ritardo dell'antenna in ricezione.

dID : identificativo del dispositivo (ancora o tag; solitamente ancora), collegato al CLE.

Le stringhe che non sono dei report vengono saltate durante la lettura del file.

Il file può essere caricato in 2 modi:

1. Directory predefinita: viene letto in automatico il test col nome specificato in un file di configurazione, all'interno della cartella nel percorso predefinito, sempre specificato nelle configurazioni.
2. Caricamento manuale del file: il file di log viene manualmente caricato nel programma tramite un componente di upload nella gui (*Immagine 3.1.16*).

Durante il test viene calcolata la posizione tramite i report presenti nel log, ricreando gli spostamenti effettuati nella realtà; ovvero, viene prelevato il valore del range (viene utilizzato il range corretto) di almeno tre ancore con la stessa sequenza, nel caso in cui non ci siano almeno tre report da utilizzare la sequenza viene saltata.

Una volta ottenuti i valori dei range questi vengono passati alla libreria che ne calcola la trilaterazione.

In casi particolari, ad esempio log registrati con il tag fermo in una posizione conosciuta, è possibile inserire il tag di riferimento in quella posizione per osservare gli errori di misurazione, calcolare gli errori di ranging e studiare le fluttuazioni dovute al rumore.

Nei test iniziali questo è stato il primo caso analizzato, utile per avere un'idea della precisione del sistema.

Si predispose un sistema di coordinate in cui l'ancora master è solitamente a x e y uguali a 0. Il tag viene fissato in una posizione e si misura manualmente la distanza dai lati del quadrante formato dalle ancore, ricavando poi le sue coordinate.

A questo punto si lascia in esecuzione il sistema per un breve periodo di tempo, raccogliendo alcune centinaia di report.

Nell'applicazione si devono quindi riprodurre le condizioni, impostando le ancore e i tag nelle stesse coordinate.

Questa operazione può essere fatta sia da un file di configurazione prima dell'attivazione del programma, che dopo, tramite il pannello delle impostazioni (*Immagine 3.1.12 e Immagine 3.1.14*).

In questo modo il tag di riferimento sarà fisso nella posizione indicata, mentre si osserverà la posizione calcolata variare in una nuvola di punti, dovuta all'errore reale, presente nei range ricavati dal file di log.

Per far partire la simulazione si carica il file e si preme il tasto "TEST".

Il test può essere effettuato anche con entrambi gli algoritmi, come spiegato nel paragrafo precedente.

I risultati della simulazione, comprendenti gli errori di posizione e gli errori di ranging, si possono poi stampare in dei file che possono essere utilizzati per graficare il test.

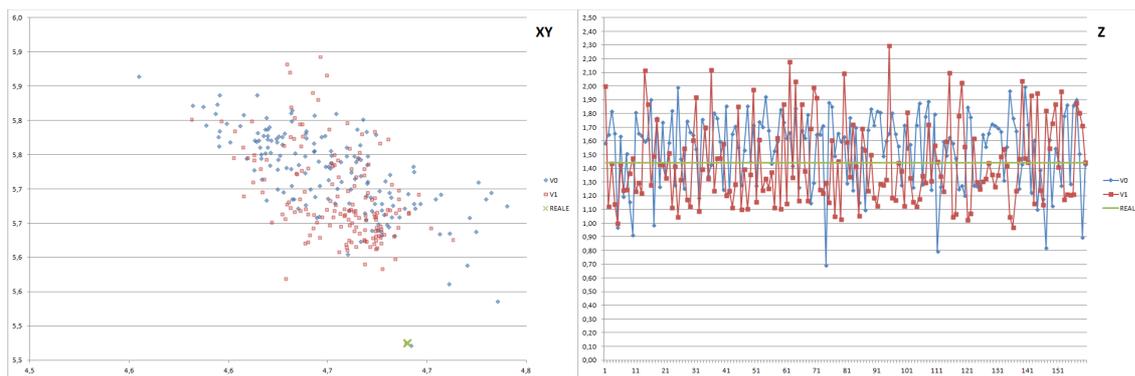


Immagine 4.2.2 : Esempio di grafici XY e Z, effettuati sui dati di un test reale.

Inoltre si può utilizzare anche la modalità di visualizzazione “Grafico”, per osservare in tempo reale il formarsi della nuvola di punti.

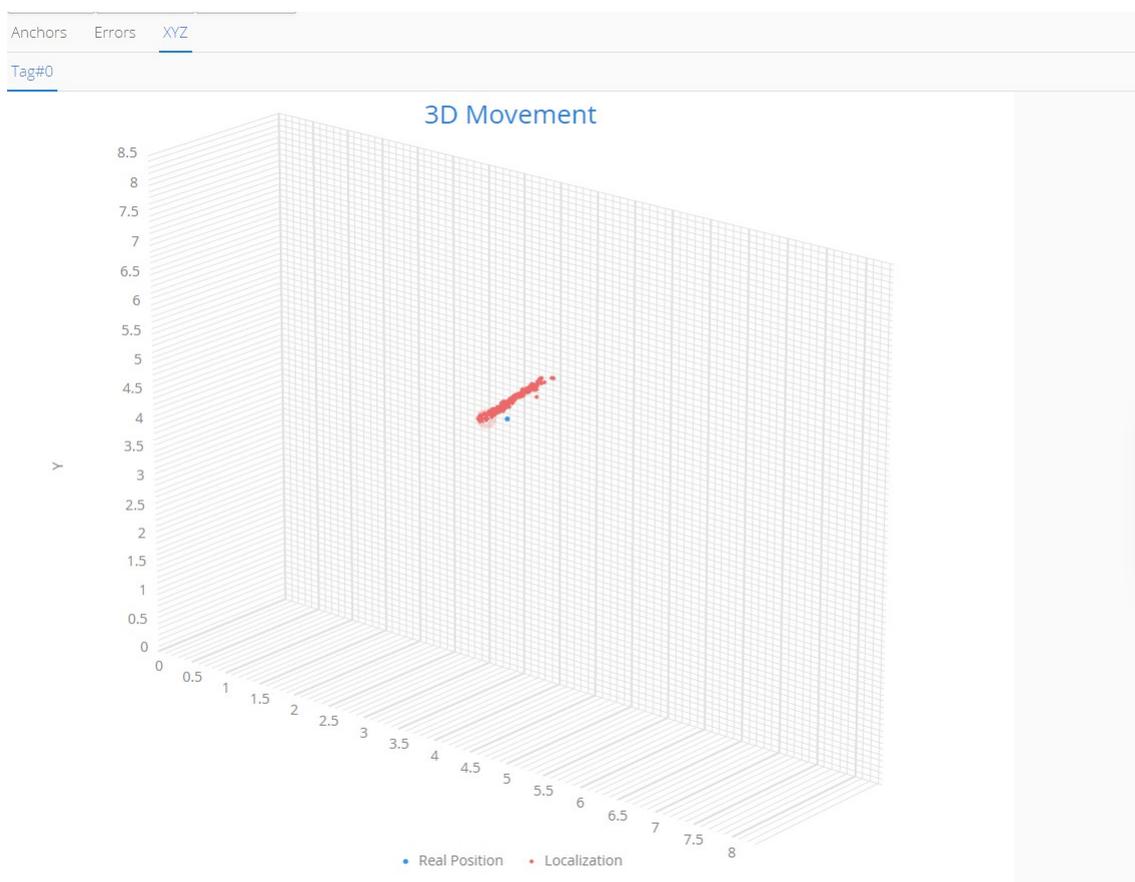


Immagine 4.2.3 : Visualizzazione del grafico XYZ nella simulazione di un test reale.

Nella simulazione di un test con tag fermo, impostando il tag di riferimento, si possono registrare anche gli errori di ranging.

Il range effettivo viene calcolato con la formula del modulo, sulle coordinate del tag di riferimento, in relazione alle quattro ancore; a questo viene sottratto il range ottenuto dal file, quindi quello reale, ottenendo l'errore.

Anche questi errori possono essere stampati in un file di log (*Immagine 4.1.3*) e quindi graficato.

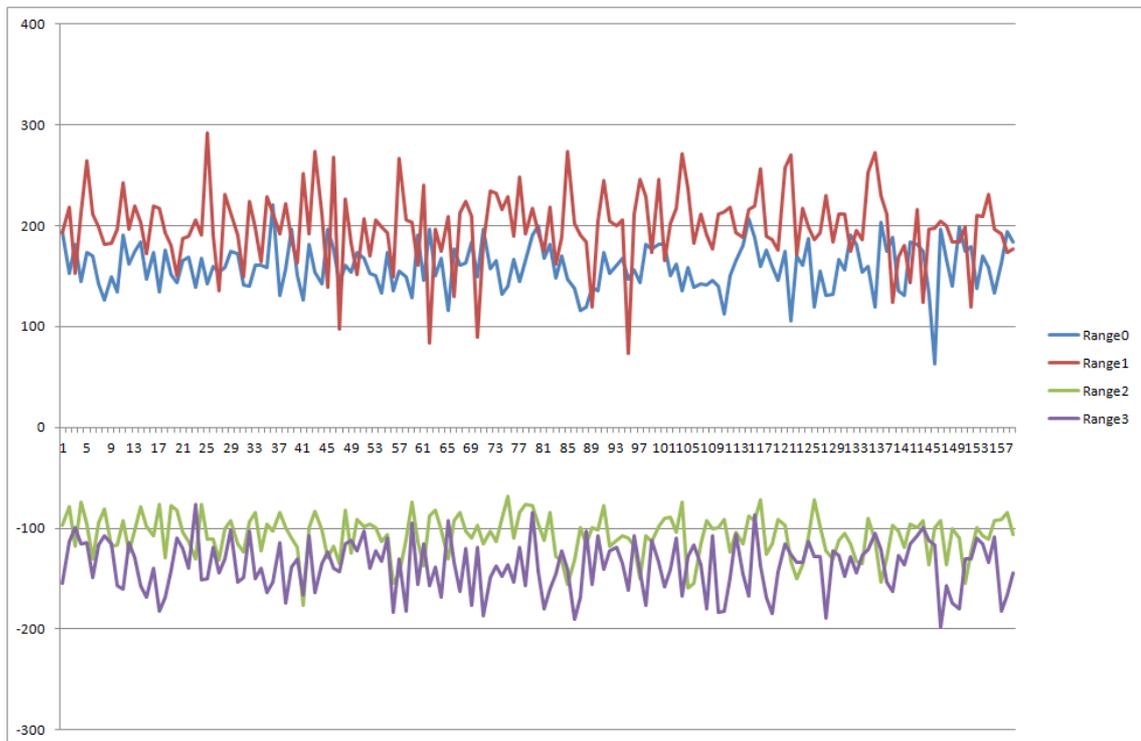


Immagine 4.2.4 : Grafico degli errori di ranging (in mm).

Lo stesso grafico si può osservare sempre in tempo reale, dall'applicazione.

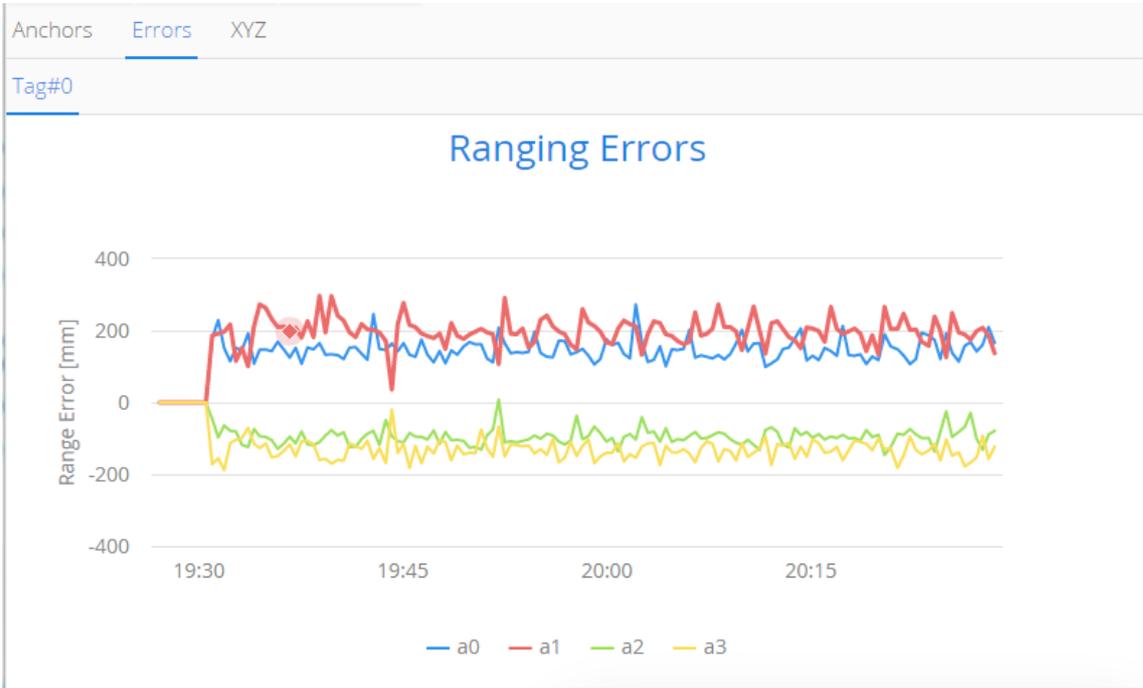


Immagine 4.2.5 : Grafico degli errori nell'applicazione in funzione.

5 Rumore e filtri

Effettuando i test come visto nel paragrafo 4.2 si può osservare il rumore presente nella realtà, durante il funzionamento del sistema.

Risulta quindi importante poter testare la resistenza al rumore della trilaterazione, motivo per cui è stata implementata la funzione di applicazione di rumore artificiale e di filtraggio, applicabili dai tasti mostrati nel paragrafo 3.1 (*Immagine 3.1.21*).

Il rumore che è possibile impostare è di due tipologie:

- **Uniforme:** Al valore del range viene sommato un numero pseudo randomico, distribuito uniformemente tra il valore massimo e minimo, impostabili dal pannello settings (*Immagine 3.1.18*).
- **Gaussiano:** Il valore del rumore, aggiunto al range è generato tramite distribuzione gaussiana, con media 0 e varianza 1, e poi ridimensionato per essere mediamente compreso tra il valore massimo e quello minimo. Ovvero, con varianza 1, la maggiorparte dei valori sarebbe stata compresa tra -1 e 1; il valore risultante viene quindi diviso per l'ampiezza massimo impostata in modo che la maggior parte dei valori sia compresa tra il massimo e minimo configurati. Viene ulteriormente ridotto il valore, per fare in modo che una percentuale ancora maggiore dei valori sia compresa nel limite, in modo da avere un rumore di ampiezza massimo non troppo differente da quello uniforme. In ogni caso, attivando il rumore gaussiano, i limiti imposti sono sempre indicativi, in quanto questo tipo di rumore non è limitato.

Il rumore Gaussiano è quello più significativo, in quanto nella realtà il disturbo non è uniforme.

Quando il rumore è applicato, dopo che il programma ha calcolato i range, vi somma un valore generato pseudo-randomicamente sul momento.

Questo valore viene poi adattato, ridimensionandolo, per avere come limiti i valori impostati.

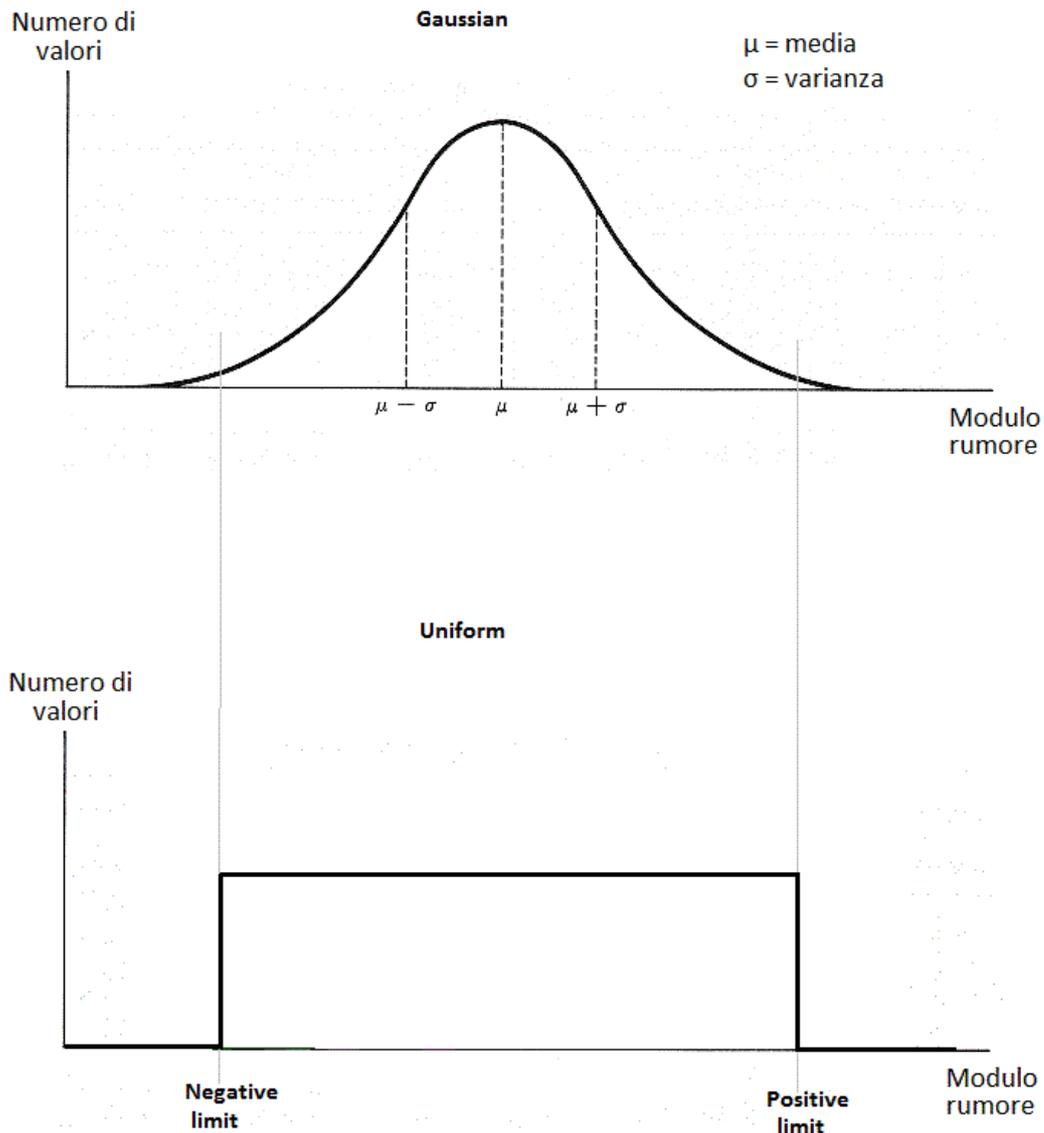


Immagine 5.0.1 : Istogramma delle distribuzioni gaussiana e uniforme.

Il rumore sui valori di range può essere visualizzato nel grafico “Ranging Errors” (Immagine 3.1.9).

Questo grafico indica la differenza, in millimetri, tra il valore con rumore e quello reale, del range.

L'asse X è un asse temporale, in cui sono segnati minuti e secondi; l'asse Y è in millimetri.

Quando la linea del grafico è sullo zero significa che l'errore è nullo.

Sul grafico sono presenti 4 linee sovrapposte, ognuna indica il rumore per i range misurati di una certa ancora.



Immagine 5.0.2 : Entrambi i tipi di rumore visualizzati sul grafico nell'applicazione.

Nella visualizzazione mappa invece, si può osservare la posizione del tag che cambia continuamente attorno al tag di riferimento, formando una nuvola di punti. Questi punti possono essere visualizzati nel grafico XYZ.

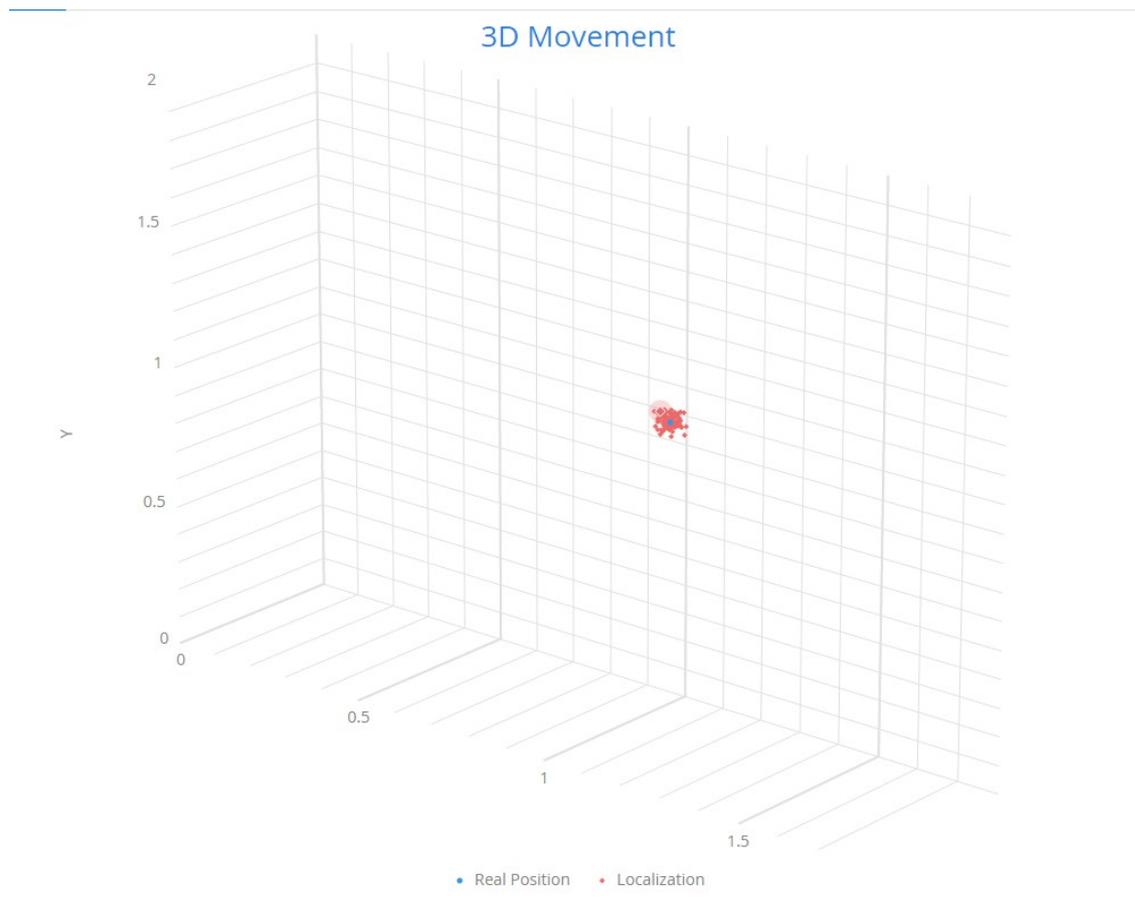


Immagine 5.0.3 : Nuvola di punti visualizzata sul grafico XYZ.

Cliccare sul tasto “Show Report” (*Immagine 3.1.12*) quando il rumore è applicato, mostrerà sia i report con rumore che quelli senza.

```

ma00 t00 00000CCC 00000CCC 14FD FD 00112732 4034 4034 a0
ma01 t00 00000BE9 00000BE9 14FD FD 00112732 4034 4034 a0
ma02 t00 00000DFC 00000DFC 14FD FD 00112732 4034 4034 a0
ma03 t00 00000BCA 00000BCA 14FD FD 00112732 4034 4034 a0

Without noise:
ma00 t00 00000CBA 00000CBA 14FE FE 00112733 4034 4034 a0
ma01 t00 00000BDC 00000BDC 14FE FE 00112733 4034 4034 a0
ma02 t00 00000DFC 00000DFC 14FE FE 00112733 4034 4034 a0
ma03 t00 00000BDC 00000BDC 14FE FE 00112733 4034 4034 a0

```

Immagine 5.0.4 : Report con rumore.

Il rumore applicato artificialmente è a valor medio nullo, inizialmente; è possibile modificare il valor medio agendo sui valori massimo e minimo nel pannello delle impostazioni.

Il rumore reale visto nel paragrafo 4.2 (*Immagine 4.2.4 e 4.2.5*), non era a valor medio nullo, inoltre ogni ancora aveva un valor medio differente, alcune positivo altre negativo.

L'errore in teoria dovrebbe risultare sempre con un valor medio nullo, ma nei test reali questo non si verifica a causa di errori di calibrazione.

Nell'hardware utilizzato, i dati di ranging subiscono una variazione in base alla potenza del segnale, quindi alla distanza, ottenendo un errore sempre negativo o sempre positivo a seconda che il tag sia lontano o vicino^[15].

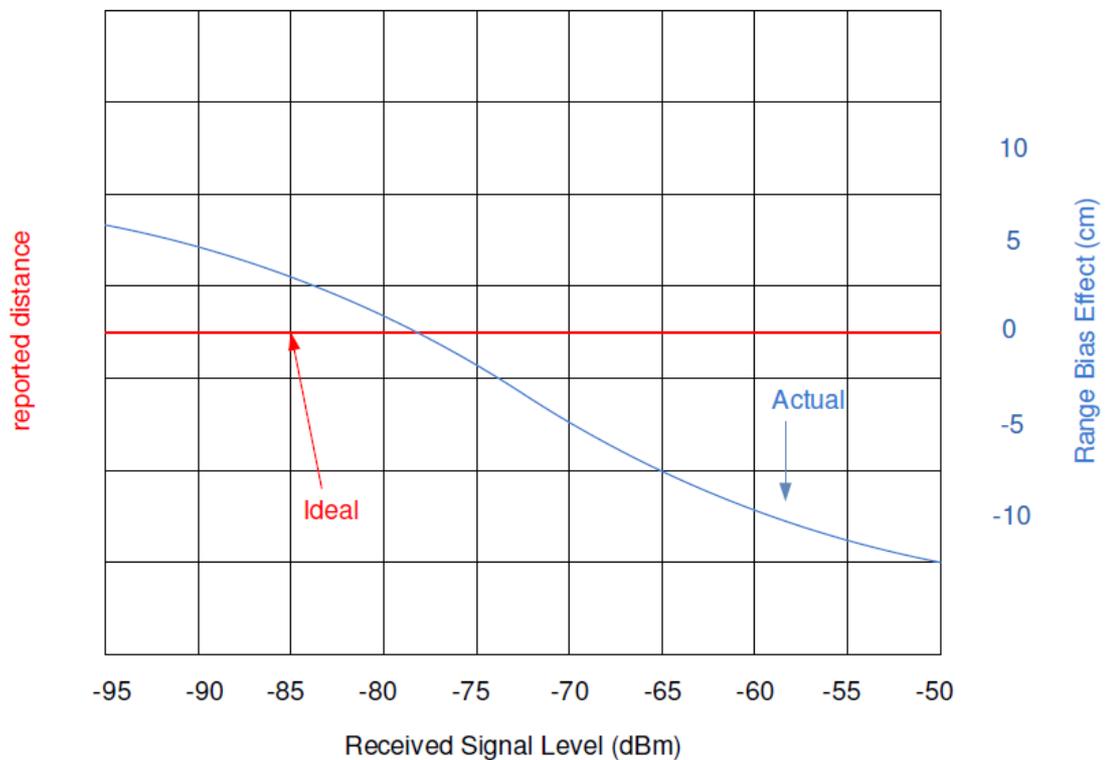


Immagine 5.0.5 : Diagramma sulla variazione di distanza riportata in base all'intensità del segnale.

Questo effetto viene compensato a livello di firmware, in cui i dati vengono corretti basandosi sul valore di Antenna Delay (presente nei report, come visto nel paragrafo 4.2), questo valore deve essere impostato in maniera corretta tramite una calibrazione dell'hardware.

5.1 Filtri

Applicare del rumore alla simulazione non è solo utile per osservare i risultati degli algoritmi di trilaterazione in presenza di errori; permette anche di poter sviluppare e testare dei filtri che permettano di smussare questo rumore.

I filtri possono avere due utilità:

- Fare una media dei valori per compensare il rumore.
- Utilizzare i valori di posizione passati per riuscire a prevedere la traiettoria del tag.

I filtri utilizzati infatti sono nominati come “Forecaster”; sono collegati a una serie temporale che contiene ogni valore di posizione associato a un valore di tempo.

Questo permette di pesare i valori in base a quanto sono vecchi, per poter fare una media esponenziale pesata che permette di ottenere risultati meno rumorosi.

Un filtro più avanzato può utilizzare la storia delle posizioni memorizzate, per ipotizzare la posizione futura del tag, stimando la traiettoria, che può quindi essere usata come posizione filtrata, senza rumore, o utilizzata in una media.

Al momento è stato implementato soltanto un tipo di filtro, che esegue semplicemente una media ponderata basata sul tempo, e non può calcolare la traiettoria.

Si possono filtrare i valori di range oppure di posizione, filtrando i valori delle singole coordinate.

La pesatura avviene tramite un valore *alpha*, compreso tra 0 e 1.

Il nuovo valore ottenuto, che viene aggiunto alla serie temporale, viene moltiplicato per *alpha*, mentre la storia passata viene moltiplicata per $1 - \alpha$.

Il valore di media ottenuto viene memorizzato diventando la nuova storia passata, che verrà usata per il prossimo calcolo della media.

Variando il valore di *alpha*, quindi, si può scegliere la potenza del filtraggio.

Mantenendo il valore a 1, il filtro sarà disattivo, ovvero il valore nuovo resterà invariato; con $\alpha = 0$ invece il valore nuovo sarà uguale a quello vecchio.

Maggiore è il valore di *alpha*, minore è l'influenza del valore passato nella media.

I valori da filtrare però non sono ottenuti a istanti di tempo fissi, l'intervallo di tempo tra l'acquisizione di un valore e un altro potrebbe variare di qualche millisecondo, oppure una trilaterazione potrebbe fallire, dovendo quindi aspettare un nuovo ciclo per avere valori validi.

Per questo motivo la pesatura nella media viene modificata con un ulteriore fattore, che è l'intervallo di tempo che intercorre tra il nuovo valore e quello vecchio; ovvero, più il valore memorizzato nella storia è vecchio, meno varrà il suo peso nella media.

Per avere un punto di riferimento per discriminare quanto vecchio è un risultato, viene scelto un valore *paramsMsReference*, che indica quanto è ampio l'intervallo di riferimento tra due risultati consecutivi.

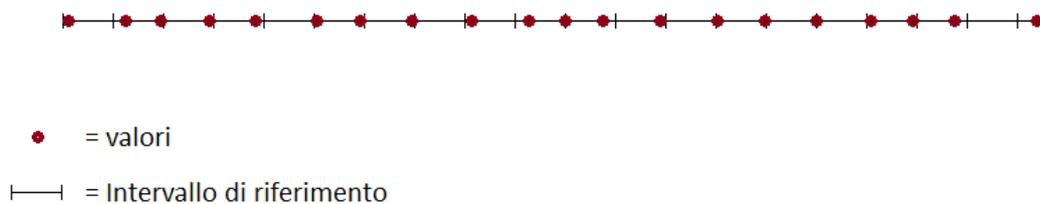


Immagine 5.1.1 : Serie temporale con riferimenti.

Un ulteriore parametro, *forecastingValidityPeriodMS*, indica quanti millisecondi devono passare dall'ultimo risultato acquisito perché non si possa più utilizzare il vecchio valore, a causa del suo “invecchiamento”.

Questo parametro e il parametro alpha possono entrambi essere modificati dall'utente tramite il pannello delle impostazioni sulla GUI (*Immagine 3.1.19*).

Il calcolo della media viene effettuato nel seguente modo:

- Ottenere dalla serie temporale il valore di tempo in cui è stato memorizzato l'ultimo risultato e calcolare la differenza con l'istante di tempo in cui si è ottenuto il nuovo risultato.
- Calcolare un valore di correzione in base a quanto la differenza di tempo calcolata differisce dall'intervallo di riferimento e dividere per il tempo di validità, in modo che più la differenza tra l'intervallo vero e quello di riferimento si avvicina al periodo di validità, più il valore di correzione si avvicina a 1.

- Calcolare il valore di alpha corretto.

Viene preso il valore *alphaCompare* come $1 - \alpha$, e viene moltiplicato per il valore di correzione, dopodichè viene sommato ad α ottenendo il valore di α corretto.

Se il fattore di correzione è uguale a 0 (l'intervallo vero è uguale a quello di riferimento) la media viene calcolata utilizzando il valore di α impostato; se invece è uguale a 1 (l'intervallo ha superato il limite di validità) α viene presa uguale a 1, per cui la storia non viene presa in considerazione.

Se il valore di α corretto risulta minore di 0 o maggiore di uno, viene comunque preso 0 o 1.

- Calcolo del valore filtrato tramite media pesata:

$$media = \alpha * risultato + (1 - \alpha) * media\ precedente$$

```
long timeGap = sampleDate.getTime() - previousDate.getTime();
double weightCorrection = (timeGap - paramsMsReference) / (double)(forecastingValidityPeriodMs);
double currentAlfa = Math.max(0, Math.min(alfa + (weightCorrection * alfaComp), 1.0));
filteredValue = (currentAlfa * sampleValue) + ((1.0 - currentAlfa) * filteredValue);
```

Immagine 5.1.2 : Porzione di codice in cui viene eseguita la media pesata.

La funzione di filtraggio viene attivata dal tasto di fianco a quello del rumore. Si possono osservare i risultati sempre dai grafici.

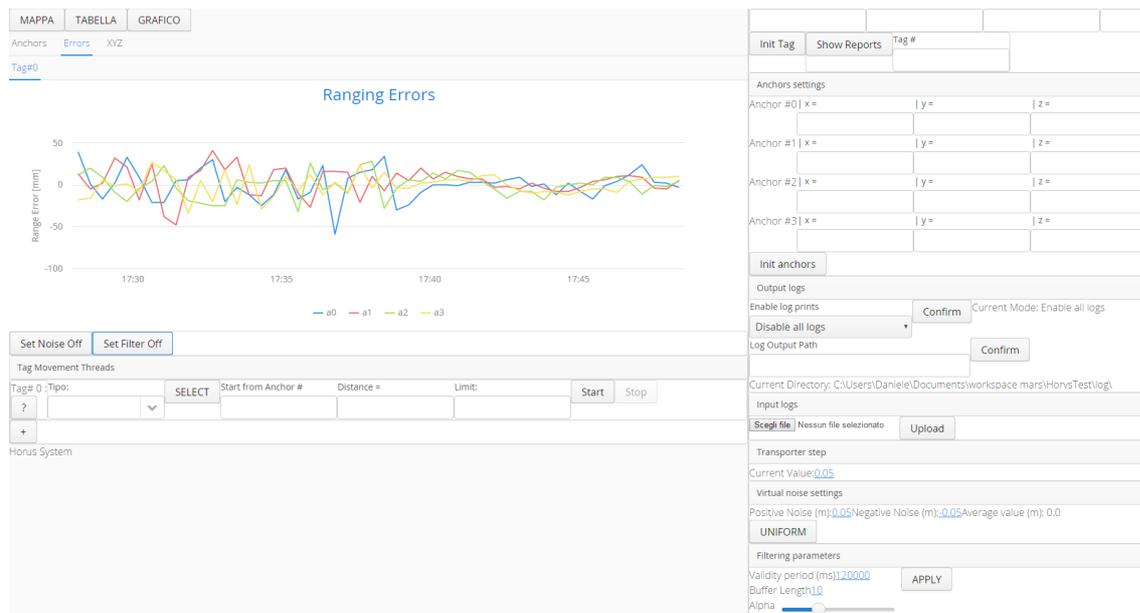


Immagine 5.1.3 : Grafico del rumore durante l'attivazione del filtro.

Il grafico in figura 5.1.3 è ottenuto con un valore di alpha di 0.33. Si può osservare il risultato anche nel grafico 3d XYZ, eseguendo due simulazioni con e senza filtro e notare la differenza nell'estensione della nuvola di punti dei risultati calcolati.

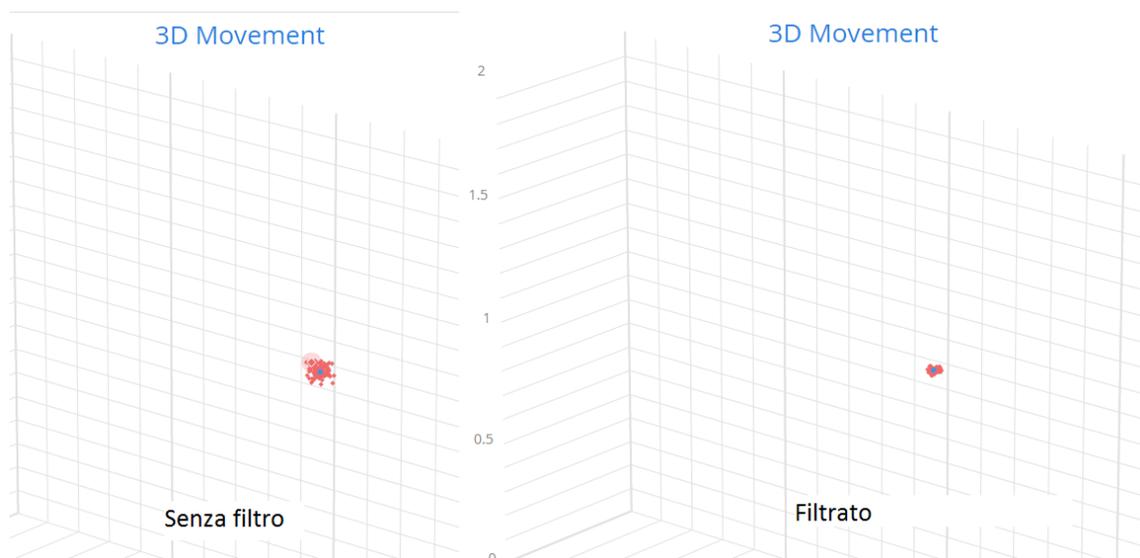


Immagine 5.1.4 : Osservazione del filtraggio su grafico XYZ.

L'effetto del filtraggio durante il movimento del tag è un lieve ritardo.

Impostando il tag di riferimento nella visualizzazione mappa, e attivando uno spostamento per il tag, con rumore e filtri attivi, si nota che la posizione calcolata si muove seguendo il pattern di movimento impostato, ma seguendo il tag di riferimento con un lieve ritardo, a causa del fatto che la posizione nuova viene avvicinata a quella precedente con la media.

Tramite un filtro più avanzato che calcola la traiettoria è possibile compensare questo ritardo, sommando alla posizione ottenuta un vettore nella direzione della traiettoria del tag; questa operazione crea comunque dei ritardi nei cambi repentini di direzione, che il filtro non può predire.

6 Server testing

Il programma, come accennato all'inizio del paragrafo 3.3, è una applicazione web che funziona su un server Tomcat.

Nel programma di gestione del sistema reale questo comportava una criticità, in quanto, nel caso il server avesse avuto dei malfunzionamenti o dei blocchi, l'intero sistema avrebbe smesso di funzionare.

Per risolvere questo problema è stato diviso il programma in due sezioni. Un programma java standalone che gestisce la logica e l'interfaccia grafica che gira su Tomcat.

Questo ha portato una nuova necessità; il dover testare la grafica senza necessariamente installare e configurare il sistema hardware e far partire il software java che lo gestisce. È stata quindi implementata una nuova funzionalità nell'applicazione di testing, in grado di instaurare una comunicazione, tramite socket, con l'applicazione di gestione della GUI e di inviare dati creati dall'utente.

Il programma java che gestisce la logica del programma e comunica con le ancore ha la parte di Server; contiene quindi un thread che apre una socket in ascolto su una porta specifica, attendendo che il client mandi una richiesta; nel frattempo il programma continua a girare in parallelo al thread di gestione del Server, senza rischiare di essere interrotto da possibili errori di comunicazione sulla socket.

L'interfaccia grafica, è il client e cerca di collegarsi al server per ottenere i dati necessari a visualizzare lo stato del sistema su schermo.

In caso di mancanza di connessione la grafica non può aggiornarsi.

La comunicazione tra Server e Client avviene tramite una interfaccia composta da diversi messaggi, il Server dovrà avere una interfaccia in cui ogni metodo verrà chiamato come risposta a un messaggio ricevuto dal client.

L'applicazione quindi dovrà avere la stessa interfaccia in modo da poter ricevere i messaggi del client e rispondergli con i giusti dati.

A questo scopo è stata progettata una classe simile al Server Manager del programma Java del sistema originale; con un thread che rimane in attesa aprendo una socket in ricezione aspettando i messaggi del client.

Le funzionalità principali, ovvero la localizzazione di tag e la visualizzazione delle posizioni su mappa o tabella, sono già implementate nell'applicazione, che può quindi rispondere subito alle richieste di base del client:

- Richiesta della dimensione della mappa.
- Richiesta della posizione delle ancore.
- Richiesta della posizione di eventuali tag.

Le ancore e i tag vengono inizializzati normalmente dall'utente nell'applicazione, a questo punto è sufficiente attivare la funzionalità di simulazione del server tramite il tasto “START SERVER” nella console dei comandi (*Immagine 3.1.20*).

Una volta attivato il server si può far partire il client che interrogherà l'applicazione sulla socket e aggiornerà la grafica visualizzando i tag e le ancore sulla mappa e sulla tabella.

La porta sulla quale la socket comunica è definita nei file di configurazione delle applicazioni e deve essere chiaramente uguale.

È così possibile utilizzare l'interfaccia grafica del sistema per visualizzare i tag, che si possono far muovere tramite le funzioni di traslazione della simulazione, e così testare la grafica per poterla mettere a punto e per verificarne la funzionalità.

Oltre alle funzionalità di base, l'applicazione si è dovuta aggiornare per simulare anche le funzionalità avanzate che sono stata inserite nel sistema, per poterle testare sulla GUI sempre tramite la simulazione del server.

La prima funzione avanzata implementata è l'anticollisione; la possibilità di avvertire un tag quando questo si avvicina a un altro, per impedire che collidano.

Il sistema è stato utilizzato in questo caso in un magazzino in cui i tag erano montati su carrelli trasportatori, e la funzione anticollisione è stata principalmente sviluppata per evitare gli scontri tra carrelli.

Inizialmente è stata implementata una sola strategia di anticollisione, detta *Bubble* ovvero Bolla.

Questa strategia semplicemente descrive una “bolla” circolare di un certo raggio intorno al tag, quando un altro tag entra in questa bolla scatta un allarme.

Sul campo, il tag può essere collegato a un sistema di frenata e può quindi mandare un segnale che ferma il carrello, o semplicemente può suonare o far vibrare il volante; i tipi di allarme possono essere personalizzati a seconda dell'hardware a disposizione.

Sulla GUI deve quindi essere visualizzata l'informazione delle collisioni.

Questa informazione viene visualizzata evidenziando i tag a rischio di collisione tramite un cerchio colorato e una circonferenza di colore più scuro che mostra l'estensione della bolla; inoltre viene mostrata una X nel punto in cui è prevista la collisione, che viene ottenuto tramite una media delle posizioni per ottenere il punto mediano tra i due tag.

Questa indicazione della croce sul punto di collisione previsto è stata poi rimossa, in quanto a causa del rumore nel sistema reale non si aveva un punto fisso, ma continuava a variare rendendo l'indicazione priva di informazione.

Per simulare questa funzione è stato inserito nell'applicazione il concetto di collisione, sfruttando le stesse classi presenti nel programma originale.

A ogni ciclo di localizzazione quindi, vengono controllate le posizioni dei tag, e nel caso in cui alcuni di questi siano vicini tra loro, a una distanza inferiore alla soglia descritta dal raggio della bolla, viene creato un oggetto contenente l'informazione di collisione, che contiene l'identificativo dei due tag.

In caso di collisioni multiple tra più di due tag, vengono creati più oggetti, uno per ogni coppia di tag in collisione.

La collisione non viene mostrata nella simulazione, ma viene comunque registrata e spedita al client che la richiede a ogni aggiornamento.

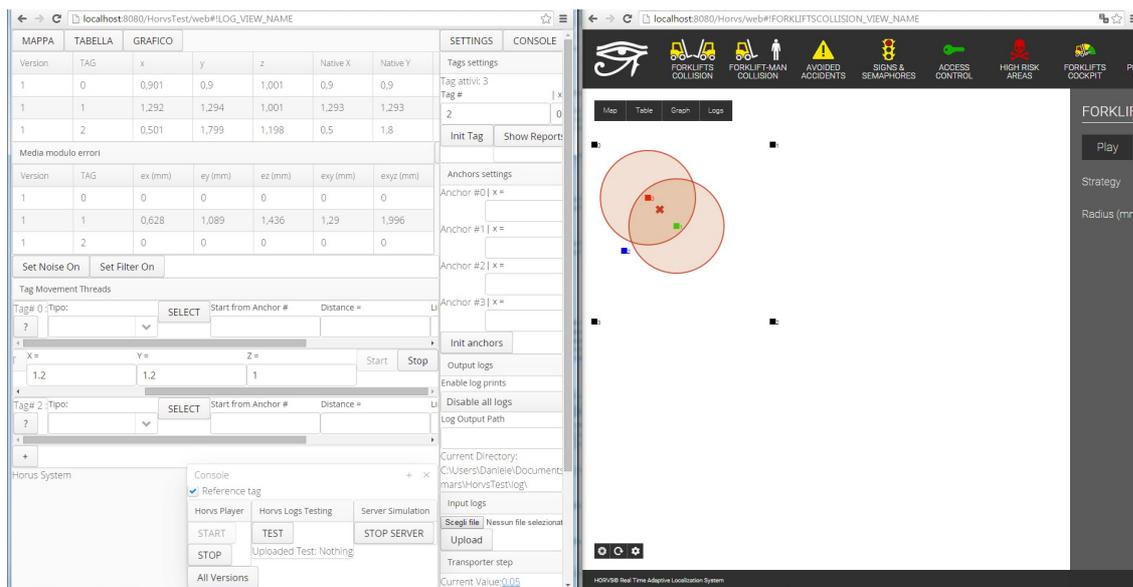


Immagine 6.0.1 : Simulazione dell'anticollisione.

Dall'interfaccia grafica è possibile inserire un valore in millimetri per definire il raggio della bolla, e quindi ridimensionarla.

Questo comando viene spedito come messaggio al server comunicando su una seconda porta, diversa dalla precedente, che è predisposta alla comunicazione di comandi, mentre la prima era per le richieste.

Infatti dall'interfaccia è possibile configurare alcuni parametri che il programma deve poi impostare.

Anche questa meccanica è stata simulata, accettando i messaggi relativi ai comandi, in questo caso per il ridimensionamento della bolla.

L'applicazione di simulazione riceve quindi il nuovo valore impostato dall'utente e sovrascrive quello predefinito.

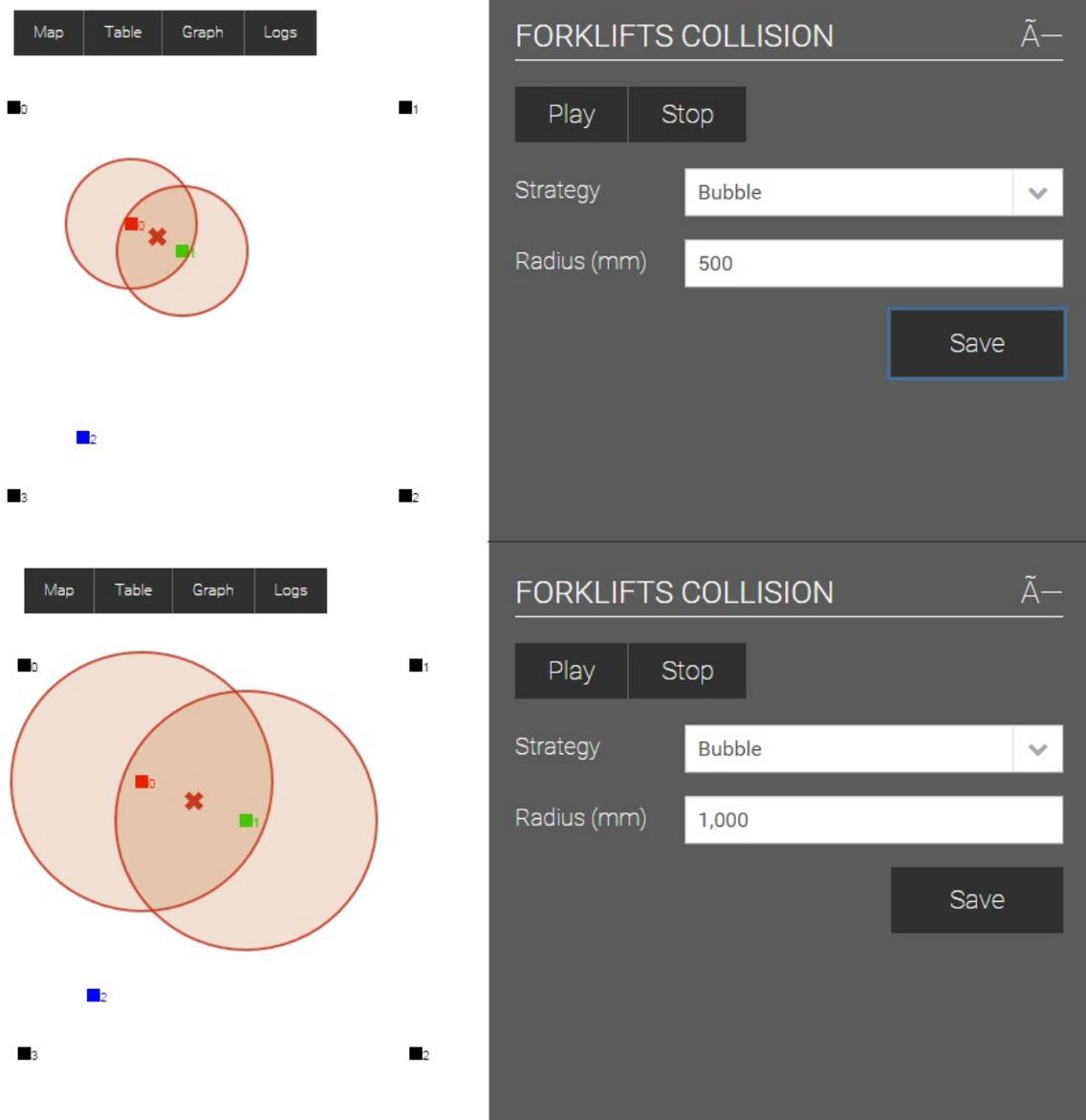


Immagine 6.0.2 : Anticollisione con differenti raggi per la bolla.

L'anticollisione viene visualizzata anche sulla tabella in una colonna specifica, in cui viene indicato se la collisione è presente o meno.

Successivamente, nelle versioni più aggiornate del programma, questa visualizzazione è stata arricchita colorando le righe dei tag in collisione di rosso.

localhost:8080/Horvs/web#!FORKLIFTCOLLISION_VIEW_NAME

  FORKLIFTS COLLISION  FORKLIFT-MAN COLLISION  AVOIDED ACCIDENTS  SIGNS & SEMAPHORES  ACCESS CONTROL

Map Table Graph Logs

Tag	CollisionAlarm	X	Y	Z
0	true	0,901	0,9	1,001
1	true	1,288	1,112	1,035
2	false	0,533	2,547	1,199

Immagine 6.0.3 : Anticollisione mostrata sulla tabella.

7 Conclusione

Al momento della redazione di questa tesi, il progetto è arrivato in una fase di demo.

Sono state progettate diverse funzioni, delle quali, al momento, sono state implementate due in particolare.

La prima è la funzione di anticollisione già citata nel capitolo precedente.

Dopo una prova sul campo della demo sono state apportate alcune modifiche grafiche: i tag in collisione vengono evidenziati da un cerchio di grandezza fissa, mentre solo la circonferenza esterna si ridimensiona in base al raggio della bolla, mentre in precedenza si ridimensionava anche il cerchio.

Inoltre, nonostante si sia mantenuta la logica in cui l'allarme scatta al momento in cui un tag entra nella bolla di un altro, a livello grafico, per motivi di chiarezza e pulizia, si è dimezzato il raggio delle circonferenze, facendo in modo che l'allarme, quindi l'evidenziamento dei tag, compaia a schermo quando le bolle dei due tag si toccano.

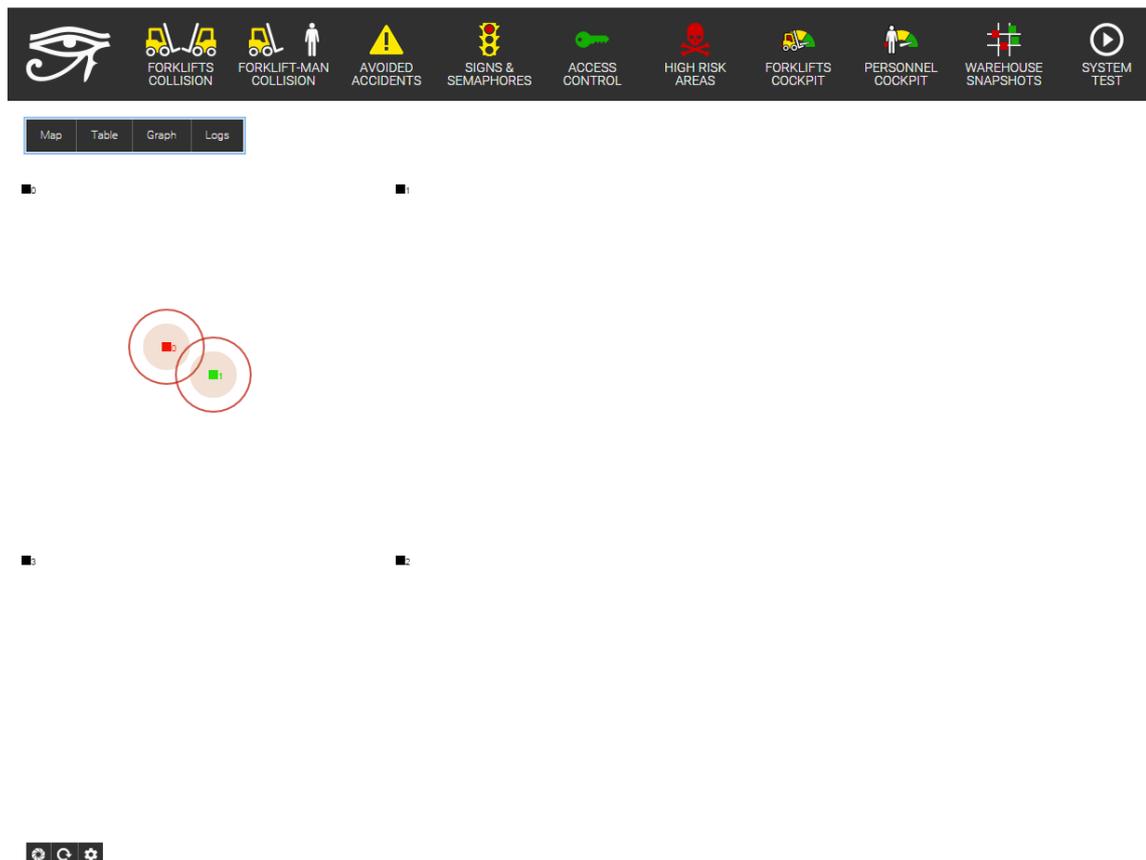


Immagine 7.0.1 : Grafica finale della collisione.

La seconda funzionalità implementata, e ultima nella versione del programma presa in considerazione in questa tesi, riguarda la possibilità di segnare alcune aree come pericolose e dare un segnale di allerta nel momento in cui un tag vi entrasse.

Al posto dei file di configurazione utilizzati nell'applicazione di test e nell'applicazione grafica, il programma principale (il Server) si basa su un database mySQL, per memorizzare i dati e le configurazioni.

Queste aree pericolose sono quindi inserite nel database e vengono richieste dalla GUI al Server.

Per questioni di semplicità, inizialmente vengono considerate solo quadrilateri regolari. Controllare che la posizione del tag non entri nell'area specificata diventa matematicamente molto più complesso in caso di figure qualsiasi, per cui, probabilmente, nel caso si debbano considerare aree di forme particolari, queste sarebbero ridotte a un insieme di rettangoli.

Queste aree vengono visualizzate sulla mappa e cambiano colore nel momento in cui un tag vi entra dentro; i tag in questione vengono poi evidenziati in modo simile a come visto nell'anticollisione.

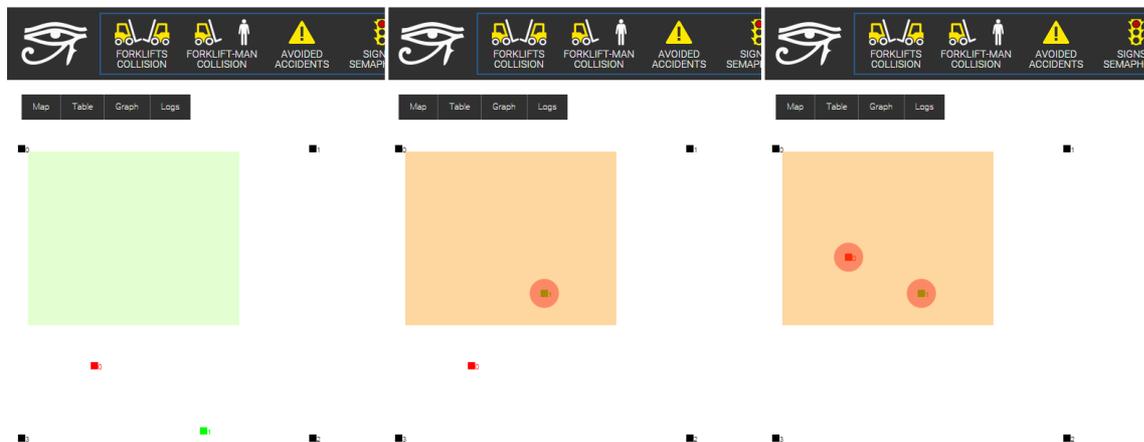


Immagine 7.0.2 : Visualizzazione grafica di Aree pericolose (High risk areas)

Per scegliere tra le diverse funzioni si utilizza la barra in alto nella grafica, in cui ogni tasto permette di visualizzare informazioni diverse.

Nel caso attuale sono attive la collisione tra carrelli e le aree pericolose.

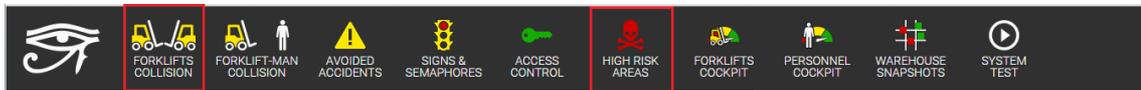


Immagine 7.0.3 : Funzioni progettate per la versione finale del programma.

La funzione “System Test” sarà una pagina del programma in cui verrà integrata l'applicazione di test argomento di questa tesi.

L'applicazione, integrata nel programma, potrà quindi essere usata anche sul campo per eseguire test e simulazioni.

Questa applicazione sarà quindi utilizzata anche in futuro, e dovrà essere supportata con aggiornamenti continui; soprattutto la parte di simulazione del Server.

Ogni volta che viene aggiunta una funzione nel programma che richiede la comunicazione tra Client e Server, andrà aggiornata l'interfaccia di metodi usata per la comunicazione, aggiungendo metodi nuovi per ricevere e rispondere a messaggi relativi a nuove funzioni.

L'interfaccia di simulazione del server andrà quindi aggiornata di conseguenza per restare sincronizzata con quella originale del programma server, in modo da poter simulare anche le nuove funzioni.

Anche le funzionalità principali dell'applicazione possono essere espanse, ad esempio con un sistema che permetta di eseguire i test reali, leggendo i file di log, di test eseguiti in movimento.

Il limite della funzione di lettura di log reali è che il tag doveva restare fermo per poter essere simulato dal tag di riferimento e osservare quindi gli errori.

Una espansione di questa funzione potrebbe permettere di muovere il tag di riferimento tramite uno spostamento tra quelli preimpostati, avendo eseguito il medesimo spostamento nella realtà.

In ogni caso il lavoro su questa applicazione proseguirà in parallelo allo sviluppo del sistema.

Il sistema stesso potrà entrare in commercio nei prossimi anni.

Elenco delle figure

1.1.1	<i>Prototipo di ancora. (Modello 3D)</i>	7
1.4.1	<i>Calcolo del Time of Flight tramite TwoWay Ranging.</i>	10
1.5.1	<i>Algoritmo Min-Max.</i>	12
1.5.2	<i>Algoritmo Multilaterazione.</i>	13
1.5.3	<i>Triangolazione.</i>	14
2.0.1	<i>Esempio di report.</i>	15
2.1.1	<i>Dichiarazione del metodo nativo per eseguire la trilaterazione.</i>	16
2.2.1	<i>Funzione che esegue il prodotto vettoriale.</i>	18
2.2.2	<i>Rappresentazione geometrica del calcolo di e_x, e_y ed e_z.</i>	18
2.2.3	<i>Calcolo della coordinata relativa X tramite trigonometria.</i>	19
2.2.4	<i>Formula inversa per ottenere il coseno.</i>	19
2.2.5	<i>Secondo triangolo per calcolare la coordinata Y.</i>	20
2.2.6	<i>Intersezione di quattro sfere su uno stesso piano.</i>	22
2.2.7	<i>Intersezione con una sfera su piano differente.</i>	23
2.2.8	<i>Distanza della quarta sfera dai punti.</i>	24
2.3.1	<i>Aree di stima della posizione causate dall'errore.</i>	26

2.3.2	<i>Shifting dei range.</i>	26
2.3.3	<i>Calcolo della GDOP.</i>	26
2.4.1	<i>Sfere concentriche.</i>	28
2.4.2	<i>Codice iniziale dell'incremento.</i>	29
2.4.3	<i>Comparazione tra diversi valori di incremento.</i>	30
2.4.4	<i>Codice finale dell'incremento.</i>	31
3.1.1	<i>Pulsanti.</i>	33
3.1.2	<i>Modalità di visualizzazione mappa, applicazione ferma.</i>	34
3.1.3	<i>Visualizzazione mappa con applicazione attiva.</i>	35
3.1.4	<i>Modalità di visualizzazione tabella, applicazione ferma.</i>	35
3.1.5	<i>Visualizzazione tabella, applicazione attiva.</i>	36
3.1.6	<i>Visualizzazione tabella, confronto dei risultati di due algoritmi.</i>	36
3.1.7	<i>Modalità di visualizzazione grafico, applicazione ferma.</i>	37
3.1.8	<i>Grafico a barre.</i>	38
3.1.9	<i>Grafico del rumore.</i>	38
3.1.10	<i>Grafico dello spostamento, con rumore applicato.</i>	39
3.1.11	<i>Schermata delle impostazioni.</i>	40

3.1.12	<i>Pannello di impostazioni per i tag.</i>	41
3.1.13	<i>Notifica contenente i report.</i>	41
3.1.14	<i>Pannello di impostazione per le ancore.</i>	41
3.1.15	<i>Pannello di impostazione per i log.</i>	42
3.1.16	<i>Pannello per caricare un file di test.</i>	42
3.1.17	<i>Impostazione dello step.</i>	42
3.1.18	<i>Impostazione del rumore.</i>	43
3.1.19	<i>Pannello delle impostazioni per il filtraggio.</i>	43
3.1.20	<i>Console di comando dell'applicazione.</i>	44
3.1.21	<i>Tasti per attivare e disattivare rumore e filtri.</i>	45
3.2.1	<i>Finestra di impostazione dei movimenti dei tag.</i>	45
3.2.2	<i>Schermata di aiuto per Anchors Tour.</i>	46
3.2.3	<i>Esempio di Anchors Tour su grafico XYZ.</i>	47
3.2.4	<i>Schermata di aiuto per Fixed Z Tour.</i>	47
3.2.5	<i>Esempio di Fixed Z Tour su grafico XYZ.</i>	48
3.2.6	<i>Schermata di aiuto per Circle.</i>	48
3.2.7	<i>Esempio di Circle su grafico XYZ.</i>	49

3.2.8	<i>Schermata di aiuto per Move To.</i>	49
3.2.9	<i>Esempio di Move To in grafico XYZ.</i>	50
3.2.10	<i>Schermata di aiuto per Z Sine.</i>	50
3.2.11	<i>Esempio di Z Sine su grafico XYZ.</i>	51
3.2.12	<i>Schermata di aiuto per Full Coverage.</i>	51
3.2.13	<i>Esempio di Full Coverage su grafico XYZ.</i>	52
3.3.1	<i>Home page di Vaadin.</i>	53
3.3.2	<i>Possibilità di selezionare i dati del grafico a server spento.</i>	54
4.0.1	<i>Confronto di risultati con e senza errori di ranging.</i>	57
4.1.1	<i>Esempio di log delle posizioni.</i>	60
4.1.2	<i>Esempio di log di report con rumore.</i>	61
4.1.3	<i>Esempio di log degli errori di ranging.</i>	62
4.1.4	<i>Esempio di grafico XY fatto con Excel.</i>	62
4.1.5	<i>Esempio di grafico della coordinata Z, fatto con Excel.</i>	63
4.1.6	<i>Esempio di grafico XY con rumore.</i>	64
4.1.7	<i>Esempio di grafico Z con rumore.</i>	64
4.2.1	<i>Esempio di log.</i>	65

4.2.2	<i>Esempio di grafici XY e Z, effettuati sui dati di un test reale.</i>	68
4.2.3	<i>Visualizzazione del grafico XYZ nella simulazione di un test reale.</i>	68
4.2.4	<i>Grafico degli errori di ranging (in mm).</i>	69
4.2.5	<i>Grafico degli errori nell'applicazione in funzione.</i>	70
5.0.1	<i>Istogramma delle distribuzioni gaussiana e uniforme.</i>	72
5.0.2	<i>Entrambi i tipi di rumore visualizzati sul grafico nell'applicazione.</i>	73
5.0.3	<i>Nuvola di punti visualizzata sul grafico XYZ.</i>	74
5.0.4	<i>Report con rumore.</i>	74
5.0.5	<i>Diagramma sulla variazione di distanza riportata in base all'intensità del segnale.</i>	75
5.1.1	<i>Serie temporale con riferimenti.</i>	77
5.1.2	<i>Porzione di codice in cui viene eseguita la media pesata.</i>	78
5.1.3	<i>Grafico del rumore durante l'attivazione del filtro.</i>	78
5.1.4	<i>Osservazione del filtraggio su grafico XYZ.</i>	79
6.0.1	<i>Simulazione dell'anticollisione.</i>	84
6.0.2	<i>Anticollisione con differenti raggi per la bolla.</i>	85

6.0.3	<i>Anticollisione mostrata sulla tabella.</i>	86
7.0.1	<i>Grafica finale della collisione.</i>	87
7.0.2	<i>Visualizzazione grafica di Aree pericolose (High risk areas)</i>	88
7.0.3	<i>Funzioni progettate per la versione finale del programma.</i>	89

Glossario

UWB Ultra Wide Band

RTLS Real Time Localization System

CLE Central Location Engine

PAN Personal Area Network

ToA Time of Arrival, stima della distanza basata sul tempo in cui il segnale viene ricevuto.

Time of Flight Tempo di volo, è il tempo che impiega un segnale per raggiungere la destinazione.

TwoWay Ranging Doppio scambio di messaggi per calcolare il Time of Flight.

TDoA Time Difference of Arrival, stima della distanza basata sulla differenza di tempo che impiegano più dispositivi a ricevere il segnale.

AoA Angle of Arrival

RSSI Received Signal Strength Indicator

JNI Java Native Interface

NaN Not a Number

GDoP Geometric Dilution of Precision

Bibliografia

- [1] DecaWave, <http://www.decawave.com/>
- [2] *APS016 Application Note Moving from TREK1000 to a product*, version 1.00
- [3] Ultra Wide Band, <http://www.etsi.org/technologies-clusters/technologies/radio/ultra-wide-band>
- [4] UltraWide Band application, <http://www.fastweb.it/internet/le-applicazioni-dell-ultra-wide-band/>
- [5] TwoWay Ranging, *Source Code Guide Decarangertls ARM source code*, version 1.00
- [6] JNI, https://it.wikipedia.org/wiki/Java_Native_Interface
- [7] JNI application,
<http://docs.oracle.com/javase/6/docs/technotes/guides/jni/spec/intro.html#wp725>
- [8] Teorema di Carnot, <http://www.math.it/formulario/trigonometria.htm>
- [9] GDOP, https://en.wikipedia.org/wiki/Dilution_of_precision_%28GPS%29
- [10] Tomcat, <http://tomcat.apache.org/>
- [11] Vaadin, <https://vaadin.com/home>
- [12] Vaadin mechanics, <https://en.wikipedia.org/wiki/Vaadin>
- [13] Vaadin features, <https://vaadin.com/features>
- [14] report format, *Source Code Guide Decarangertls ARM source code*, version 1.00
- [15] range bias on the reported distance, *APS011 Application Note Sources of error in DW1000 based two-way ranging (TWR) schemes*, version 1.0