UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria
Corso di Laurea in Ingegneria dell'Automazione

Corso di Progettazione dei Sistemi di Controllo
a.a. 2009/10

Motion Capture

Luca Fardin

Lorenzo Corso

Alberto Tonello

Padova, 15 Febbraio 2010

Indice

1 Abstract

Fino a questo momento il passive motion capture è stato implementato con un approccio di tipo centralizzato. Il grande onere computazione di questo sistema non permette un comportamento real-time.
Il progetto svolto si occupa di implementare il passive motion capture con un approccio di tipo distribuito.
Il problema è stato prima sviluppato nel caso semplificativo bidimensionale, per poi essere esteso al caso significativo tridimensionale. Sviluppando, in entrambi i casi, diversi algoritmi per la ricostruzione dei marker a partire dai piani immagine delle telecamere.
A partire dai marker ricostruiti, è stata fornita una semplice rappresentazione dell'incertezza nella ricostruzione del punto mediante informazioni sulla distanza di quest'ultimo dalle rette dalle quali è stato ricavato.
I risultati dei confronti fra l'approccio centralizzato e quello distribuito permettono di affermare che quest'ultimo riduce il costo computazione. Tuttavia risulta necessario un ulteriore approfondimento, in quanto le simulazioni Matlab non sono sempre attendibili per quanto riguarda il calcolo dei tempi di esecuzione dei programmi.
La tesi è stata suddivisa in cinque parti.
Nella prima parte, quella introduttiva, è stata riportata una breve introduzione al modello della telecamera ed alla procedura del motion capture. La terza sezione della prima parte è la più importante, essa contiene gli obbiettivi del progetto e l'approccio adottato nel suo svolgimento.
Nella seconda parte sono state introdotte ed analizzate nel dettaglio le funzioni implementate per ottenere il motion capture distribuito nel caso semplificato bidimensionale.
La parte successiva contiene l'introduzione ed i risultati delle simulazioni nel piano.
Nellla terza parte e nella quarta parte si estende quanto fatto nel caso bidimensionale al caso tridimensionale.
L'ultima parte contiene le conclusioni e gli sviluppi futuri.

2 Introduzione

2.1 Modello della telecamera

La telecamera è l'equivalente elettronico dell'occhio umano, essenziale per un sistema di visione. Consiste in tre elementi principali: l'ottica, il sensore e l'elettronica ad esso associata che controlla il segnale acquisito dagli elementi sensibili e fornisce in uscita un segnale analogico corrispondente all'immagine acquisita. Le principali caratteristiche delle telecamere sono:

- tipo di sensore: CCD o CMOS;
- forma del sensore: lineare o matriciale;
- sensibilità del sensore: B/N, colore, UV, infrarosso, raggi X;
- tipo di scansione: lineare o progressiva;
- risoluzione;
- tipo del segnale in uscita.

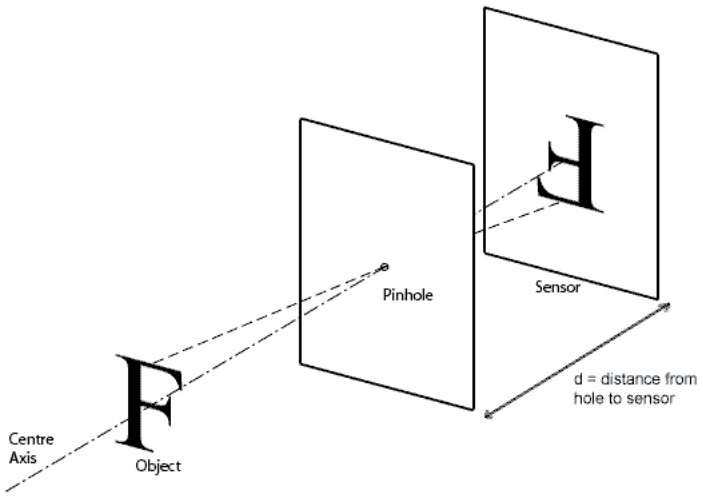

Figura 1: Proiezione immagine

Per poter interpretare il funzionamento di una telecamera dobbiamo basarci su un modello prospettico che possa approssimarla nel miglior modo. Una trasformazione di prospettiva proietta punti 3D su un piano, rivestendo un ruolo centrale nell'elaborazione dell'immagine fornendo una approssimazione al modo in cui l'immagine si forma guardando un mondo tridimensionale. Il modello che usualmente si utilizza, è il modello di *pin-hole*, per la sua semplicità d'interpretazione e applicazione. Infatti esistono svariati modelli che descrivono il fenomeno con una maggior precisione e, inevitabilmente, maggior complessità.
Il modello pin-hole approssima la telecamera come una scatola chiusa su cui, in una delle facce è stato praticato un foro (dal termine pin-hole, foro di spillo), detto *centro ottico* o *centro di proiezione*. Il piano che passa per questo punto, parallelo al piano xy, si chiama *piano focale*.

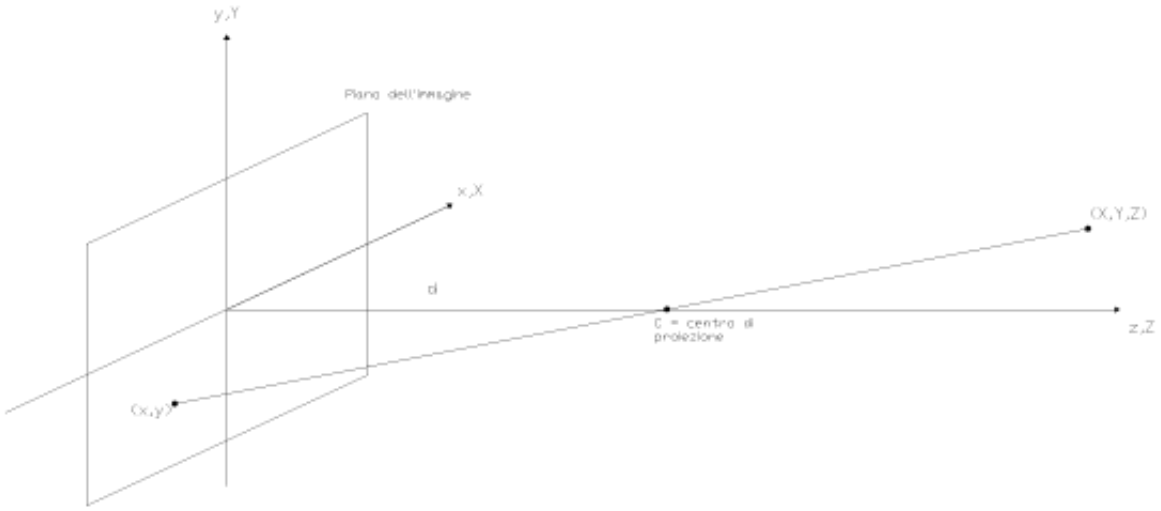

Figura 2: Modello PinHole

Mettendo un oggetto qualunque di fronte al piano focale, i raggi luminosi riflessi da quest'ultimo passano attraveso il foro e vanno a colpire un piano parallelo a quello focale, chiamato *piano immagine*. La distanza 'd' tra il piano focale e il piano immagine si chiama *distanza focale*. Si definisce il sistema di coordinate della telecamera (x, y, z) con il piano dell'immagine coincidente col piano xy e *l'asse ottico*, stabilito dall'asse longitudinale dell'obiettivo, lungo l'asse z. Siano (X, Y, Z) le coordinate di riferimento di un punto generico di una rappresentazione 3D, e si consideri Z > d, cioè l'immagine che ci interessa di davanti all'obiettivo (centro ottico).
Si ottiene la relazione che fornisce le coordinate (x, y) sul piano immagine della proiezione del punto 3D (X, Y, Z) utilizzando delle semplici formule dei triangoli:

$$x = -\frac{d \cdot X}{Z - d} = \frac{d \cdot X}{d - Z} \tag{1}$$

$$y = -\frac{d \cdot Y}{Z - d} = \frac{d \cdot Y}{d - Z} \tag{2}$$

Si possono notare i segni negativi delle formule, derivano dal fatto che i punti dell'immagine sono capovolti, come si può vedere dalla Figura 2. Importante evidenziare che queste equazioni sono non lineari perchè implicano la divisione per Z.
Spesso è conveniente esprimere queste equazioni in forma matriciale che si può facilmente ottenere utilizzando le coordinate omogenee. Le coordinate omogenee di un punto di coordinate cartesiane (X, Y, Z) sono definite come (kX, kY, kZ, k) con k una costante albitraria diversa da zero. Un punto nel sistema di riferimento cartesiano può essere espresso col vettore:

$$\mathbf{w} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3}$$

La sua forma omogenea:

$$\mathbf{w}_o = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \tag{4}$$

Si capisce come si possano ricavare le coordinate cartesiane da quelle omogenee dividendo le prime tre righe delle componenti omogenee per la quarta. Definiamo la matrice di trasformazione della prospettiva:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{bmatrix} \tag{5}$$

Definiamo:

$$\mathbf{c}_o = \mathbf{P}\mathbf{w}_o = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} = \begin{bmatrix} kX \\ kY \\ kZ \\ -\frac{kZ}{d} + k \end{bmatrix} \tag{6}$$

Gli elementi di c_o sono le coordinate della telecamera in forma omogenea. Come si è visto in precedenza, si possono ricavare le coordinate in forma cartesiana in qualsiasi punto del sistema di coordinate della telecamera ottenendo in forma vettoriale:

$$\mathbf{c} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{dX}{d-Z} \\ \frac{dY}{d-Z} \\ \frac{dZ}{d-Z} \end{bmatrix} \tag{7}$$

Le prime due componenti di **c** sono le coordinate (x, y) del piano dell'immagine di un punto 3D proiettato di coordinate (X, Y, Z). Per rappresentare nuovamente un punto dell'immagine in 3D si utilizza la trasformazione prospettica inversa, ricavando facilmente la matrice inversa di **P**:

$$\mathbf{w}_o = \mathbf{P}^{-1}\mathbf{c}_o \tag{8}$$

La terza componente della formula (2.7), non è di nessun interesse in termini del modello e agisce come variabile indipendente nella trasformazione prospettica inversa. Si nota come questa trasformazione non sia biunivoca, infatti conoscendo le coordinate cartesiane ma non conoscendo tutte le coordinate del punto in 3D, per esempio la sua coordinata Z, non è possibile ricavare completamente il punto 3D.
Dobbiamo tener presente che in queste formule non si è tenuto conto delle lenti poste davanti al centro ottico in modo da poter regolare la messa a fuoco dell'immagine in esame. Per quanto riguarda la regolazione della luce che raggiunge il piano immagine si usa un diaframma. Per quanto semplice sia l'utilizzo di questo modello, bisogna tener presente che non considera alcuni problemi, per citarne alcuni:

- distorsioni geometriche: spostamento dei punti sul piano immagine dovuti alla presenza delle lenti;
- sfocamento: dovuto alla non perfetta messa a fuoco dell'immagine, poichè non tutti gli oggetti saranno inquadrati alla stessa distanza dal piano immagine;

- vignetting effect: causato dalla presenza del diaframma, che diminuisce l'intensità della luce riflessa passando dal centro ai bordi dell'immagine;
- offuscamento dovuto alla limitata banda ottica delle lenti.

Comunque la maggior parte dei fattori non considerati nel modello si possono ritenere trascurabili.

2.2 Motion capture

Il termine motion capture è utilizzato per descrivere il processo di registrazione del movimento e, successivamente, modellizzazione digitale di tale movimento.
In una sessione di motion capture, i movimenti di uno o più attori vengono campionati più volte per secondo. I dati dell'animazione vengono poi mappati in un modello tridimensionale che può essere utlizzato per differenti applicazioni.
Le principali applicazioni nelle quali viene applicata la tecnica del motion capture si sviluppano in ambito clinico e nell'intrattenimento.
Nel primo caso viene usata per la valutazione funzionale dei pazienti per quanto riguarda diversi fattori biometrici. Inoltre, in ambito sportivo, essa è utilizzata per analizzare il movimento degli atleti.
Per quanto riguarda l'intrattenimento il motion caption viene utilizzato nella creazione di videgiochi e di personaggi cinematografici. Nei videogiochi è spesso utilizzato per animare atleti ed altri personaggi. Nei film viene utilizzato per creare effetti in computer grafica.

Motion tracking, o motion capture, iniziò ad essere usato come strumento nell'analisi fotogrammetrica nelle ricerche bimeccaniche negli anni settanta ed ottanta. Poi si espanse nell'educazione, nell'allenamento, nello sport, fino ad arrivare, recentemente, all'animazione computerizzata, in conseguenza della crescente maturazione tecnologica.

Per quanto riguarda lo svolgimento del progetto sono di maggiore interesse i sistemi di motion capture che fanno uso di dispositivi ottici, nello specifico quelli passivi. I sistemi ottici utilizzano dati provenienti da sensori di immagine per triangolare la posizione di un soggetto nello spazio tra più videocamere calibrate per ottenere la sovrapposizione dei campi visivi.
L'acquisizione dei dati è tradizionalmente implementata usando dei marker speciali attaccati ad un attore. Tuttavia, sistemi più recenti, sono in grado di generare dati accurati tramite il tracking dalle caratteristiche di una superficie identificate in modo dinamico per ogni particolare soggetto.
L'espansione dell'area di motion capture è ottenuta aggiungendo più telecamere. Questi sistemi producono dati con tre gradi di libertà per ogni marker, e l'informazione sull'orientazione è ottenuta dall'orientazione relativa di tre o più marker. I sistemi ottici di motion capture fanno uso di differenti tecniche, che verranno in seguito analizzate.

- **Marker passivi:**
  I sistemi ottici passivi fanno uso di marker ricoperti da un materiale retro-riflettente, per riflettere la luce generata in prossimità della lente delle telecamere.
  La soglia delle telecamere può essere impostata affinchè venga rilevata e campionata solamente la luce riflessa dai marker, ignorando tutto il resto.
  Il centroide del marker viene stimato come una posizione all'interno del piano bidimensionale dell'immagine catturata dalla telecamera. Il valore sulla scala di grigi di ogni pixel può essere utilizzato per ottenere una accuratezza sub-pixel trovando il centroide della gaussiana.
  Un oggetto con dei marker posti in posizioni conosciute viene utilizzato per calibrare le videocamere ed ottenere la loro posizione, successivamente viene misurata la distorsione della lente di ogni singola camera. Qualora due camere calibrate vedano un marker, la sua posizione tridimensionale può essere ricostruita. Tipicamente un sistema di motion

capture a marker passivi è costituito da un numero di videocamere cha va da sei a ventiquattro. Esistono anche sistemi con più di trecento camere per tentare di ridurre l'errore dovuto allo scambio dei marker. Talvolta sono necessarie altre camere per ottenere una copertura completa dell'area intorno al soggetto in movimento, o nel caso siano presenti più soggetti.
I venditori hanno sviluppato dei software per ridurre problemi quali lo scambio dei marker, dal momento che tutti i marker risultano identici. A differenza dei sistemi a marker attivi o dei sistemi magnetici, i sistemi a marker passivi non necessitano dell'utilizzo di cavi o apparecchi elettronici da parte del soggetto. Al contrario, centinaia di palle di gomma vengono ricoperte di nastro riflettente, che necessita di essere sostituito periodicamente. I marker vengono posti direttamente sulla pelle in ambito biomeccanico, oppure vengono attaccati con del velcro a delle tute progettate specificatamente per il motion capture.

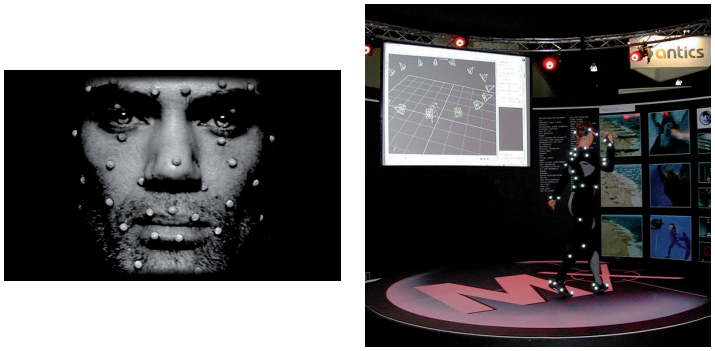

Figura 3: Motion capture con sistemi a marker passivi.

- **Marker attivi:**
  I sistemi ottici attivi triangolano la posizione illuminando un LED per volta molto velocemente, oppure illuminando più LED ed utilizzando un software per identificarli in funzione della loro posizione relativa. Al contrario di riflettere luce che viene generata esternamente, i marker stessi vengono alimentati per emettere luce. Rispetto ai sistemi a marker passivi, i sistemi a marker attivi, mediamente, sono caratterizzati da una maggiore distanza ed un maggiore volume nel quale può essere effettuato il motion capture.

- **Marker attivi tempo modulati:**
  I sistemi a marker attivi possono essere ulteriormente migliorati accendendo un marker per volta, o facendo il tracking di più marker nel tempo e modulando l'ampiezza o la larghezza di pulsazione per ottenere l'identificazione dei marker. Sistemi modulati con dodici megapixel di risoluzione spaziale mostrano movimenti più precisi rispetto a sistemi ottici a quattro megapixel, avendo sia maggiore risoluzione spaziale che temporale.
  L'identificazione univoca dei marker permette di eliminare il problema di scambio dei marker e permette di fornire dati più puliti rispetto a tutte le altre tecnologie.
  Un altro vantaggio di questi sistemi è quello di poter operare all'esterno, anche sotto la

luce diretta del sole, grazie a LED forniti di processore ed ad una sincronizzazione radio. Inoltre, processare i dati provenienti da un sistema con identificazione univoca dei dati permette la riduzione di ripuliture manuali o filtrate dei dati, garantendo così un minor costo computazionale.
Questa maggiore accuratezza e precisione richiedono più processi rispetto a sistemi passivi, tuttavia i processi addizionali vengono svolti direttamente sulla telecamera, garantendo una maggiore risoluzione ed una maggiore velocità.

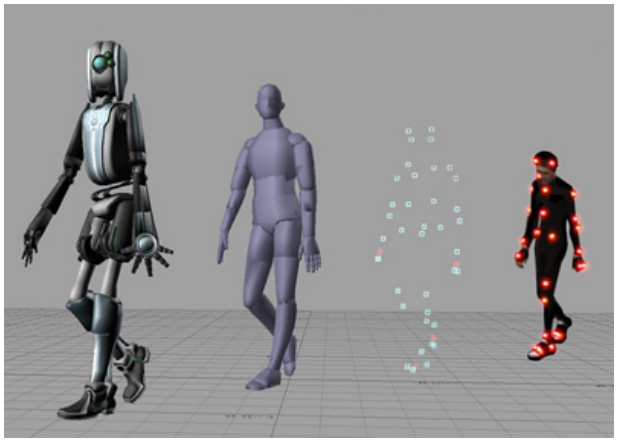

Figura 4: Motion capture con sistema a marker attivi.

- **Marker impercettibli semi-passivi:**
  Questi sistemi invece di utilizzare videocamere ad alta velocità, fanno uso di economici proiettori multi-LED ad alta velocità. Questi speciali proiettori multi-LED codificano otticamente lo spazio. Invece di utlizzare marker riflettenti o LED attivi emettitori di luce, questo sistema utilizza marker tag fotosensibili per decodificare i segnali ottici. Attaccando questi marker tag, provvisti di foto sensori, ai punti della scena, essi sono in grado calcolare, oltre alla locazione e l'orientazione di ogni punto, anche l'illuminazione incidente e la riflettanza.
  Queste tracking tag possono lavorare in ambienti illuminati dalla luce solare e possono essere applicate ad oggetti risultando praticamente invisibili. Questo sistema supporta un numero illimitato di marker tag in una scena, con ogni tag identificato univocamente al fine di eliminare problemi di riacquisizione dei marker. Rispetto ai tradizionali sistemi che fanno uso di videocamere ad alta velocità, questo approccio richiede una riduzione significante della larghezza di banda nell'acquisizione dei dati.

- **Markerless:**
  Le emergenti tecniche e ricerche nell'ambito della visione computazionale stanno portando ad una rapida diffusione dell'approccio markerless al motion capture. I sistemi markerless non richiedono che il soggetto indossi equipaggiamenti speciali per il tracking.

Figura 5: Motion capture con marker tag semi-passivi.

Questo approccio fa uso di speciali algoritmi, designati per permettere al sistema di analizzare flussi multipli di ingressi provenienti da sensori ottici e di identificare le forme umane, spezzandole in elementi costitutivi adatti per il tracking.

Esistono sistemi di motion capture che non fanno uso di dispositivi ottici.
Tra questi vi sono quelli che utilizzano sistemi inerziali, basati sull'uso di sensori inerziali miniaturizzati, modelli biomeccanici ed algoritmi di fusione.
Un altro approccio prevede l'uso di una struttura articolata fornita di sensori che il soggetto deve indossare, e che permette la misura dei movimenti relativi del soggetto.
Un ultimo approccio utilizza sistemi magnetici per calcolare posizione ed orientazione dal flusso magnetico relativo di tre avvolgimenti ortogonali sia sul trasmettitore che su ogni ricevitore.

2.3 Approccio distribuito

Il progetto svolto si occupa specificatamente del motion capture attuato mediante sistemi che utilizzano marker passivi.
Fino a questo momento l'approccio adottato per affrontare il passive motion capture è stato di tipo centralizzato. Un gran numero di telecamere hanno lo scopo di riprendere un ambiente chiuso, all'interno del quale sono presenti dei soggetti provvisti di marker passivi riflettenti. Ogni telecamera fornisce un numero costante di immagini al secondo, che vengono salvate ed elaborate successivamente. Il problema principale consiste nel fatto che l'elaborazione dei frame di ogni telecamera del sistema viene svolta in maniera centralizzata. Questo tipo di approccio richiede un costo computazionale molto elevato, che si traduce in tempi molto lunghi di elaborazione. In aggiunta a questo problema, i sistemi di motion capture passivi centralizzati richiedono, spesso, un filtraggio manuale dei dati provenienti dall'elaborazione, rendendo ancora più oneroso il procedimento. Basti pensare che, per poche ore di riprese, risultano necessari giorni di elaborazione.
Per ovviare a questo problema si è pensato di adottare un approccio distribuito al passive motion capture. L'obbiettivo è quello di ridurre sensibilmente i tempi di elaborazione rispetto al metodo centralizzato, al fine di ottenere, se possible, l'elaborazione dei dati in tempo reale.
Per raggiungere l'obbiettivo proposto, l'approccio distribuito prevede l'utilizzo di videocamere dotate di un microprocessore, sul quale effettuare alcune delle elaborazioni direttamente sulla telecamera. Eseguendo parallelamente in ogni telecamera alcune delle elaborazioni che prima venivano svolte sequenzialmente da un solo processore, si riduce l'onere computazionale e, di conseguenza, il tempo necessario per l'elaborazione dei dati.

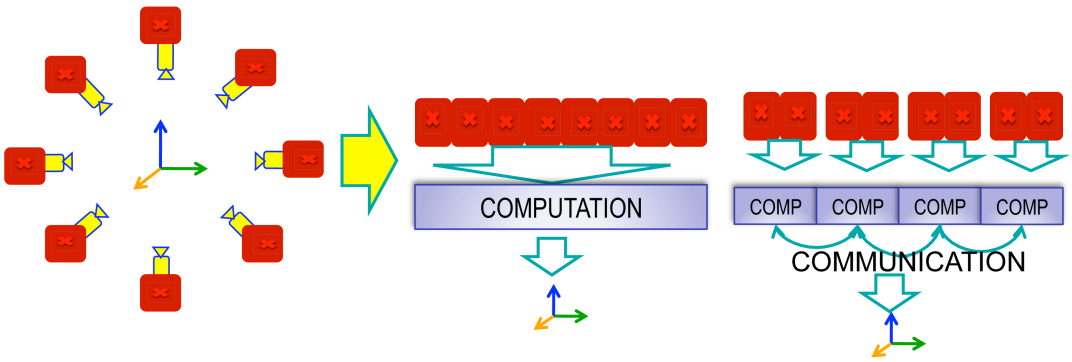

Figura 6: Approccio centralizzato e distribuito al passive motion caption.

Per lo scambio di informazioni fra le telecamere si utilizza uno schema di comunicazione ad albero, come quello riportato in Figura 7.
Nel primo passaggio vengono associate le telecamere a due a due. Ognuna di esse ha già elaborato il proprio frame, ricavando i centri delle proiezioni dei marker sul piano immagine e, successivamente, le rette passanti per i centri individuati ed il centro focale della telecamera.
Ogni coppia di videocamere confronta la distanza fra le rette individuate dalla prima e quelle individuate dalla seconda. Qualora due rette risultino sufficientemente vicine si può affermare che esse hanno individuato un marker nel loro punto di minima distanza. Dal confronto fra

la coppia di telecamere si ha in uscita un insieme di marker ricostruiti ed un insieme di rette unmatched. Seguendo il ramo dell'albero, l'insieme individuato viene elaborato con l'insieme in uscita ad un' altra coppia di telecamere, dando nuovamente in uscita un insieme di punti e rette. Questo procedimento di associazione viene ripetuto fino ad arrivare all'ultimo passaggio, dal quale si ha in uscita l'insieme finale di marker ricostruiti e rette unmatched.
Lo schema garantisce che tutte le rette vengano elaborate tra loro in uno dei livelli dell'albero, evitando in questo modo la perdita di informazione.

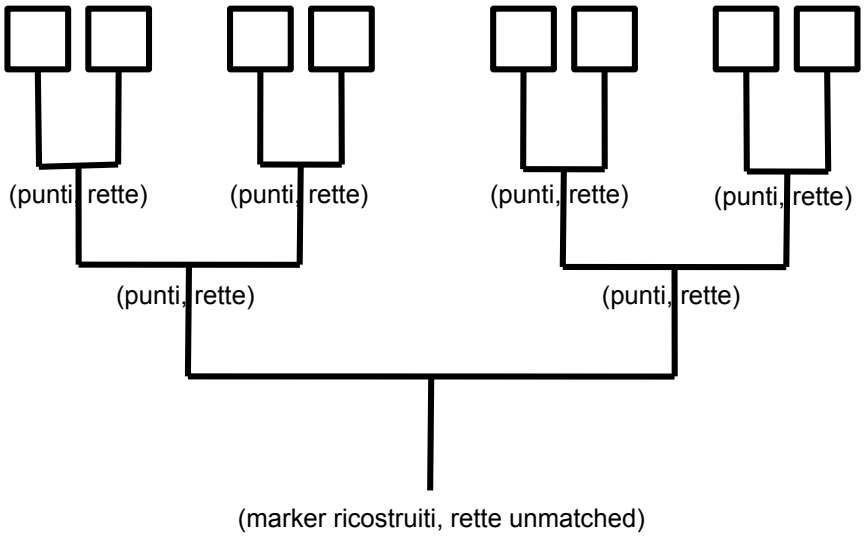

Figura 7: Schema ad albero utilizzato per il passive motion capture con otto telecamere.

Nel corso del progetto è stato per prima cosa analizzato il caso bidimensionale, per poi estendere i risultati al caso tridimensionale.

3 Progetto nel piano

Per prima cosa è stato affrontato il problema del motion capture nel piano.
Per ottenere un approccio distribuito al problema del motion capture si è pensato di organizzare il progetto nell'implementazione di tre algoritmi, secondo lo schema in Figura 8

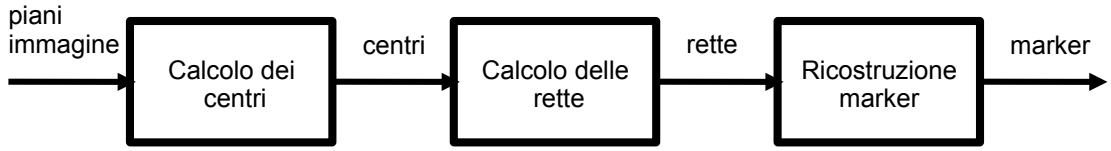

Figura 8: Schema a blocchi delle funzioni da implementare nel piano.

- **Calcolo dei centri:** il primo problema da affrontare prevede l'implementazione di un algoritmo per la determinazione dei centri delle proiezioni dei marker sul piano immagine delle telecamere.
- **Calcolo delle rette:** il secondo aspetto prevede l'implementazione del codice per ottenere le rette passanti per il centro calcolato precedentemente ed il centro focale della telecamera.
- **Ricostruzione della posizione dei marker nel piano:** questo è l'aspetto più importante e significativo del progetto. In questa fase sono stati implementati diversi algoritmi per la determinazione dei marker nel piano a partire dalle rette, calcolate precedentemente, di ogni telecamera.

Di seguito verranno esposte nel dettaglio le soluzioni adottate per risolvere i tre problemi.
Si vuole far notare, innanzitutto, che i primi due algoritmi vanno implementati in ogni telecamera e vengono fatti girare parallelamente in ognuna di esse per ogni frame. Al contrario il terzo problema, nelle differenti implementazioni proposte, prevede la comunicazione fra le telecamere, mediante il consueto schema ad albero che stabilisce la gerarchia nello scambio di informazioni.

3.1 Calcolo dei centri

Il primo problema è stato risolto mediante l'algoritmo implementato nella funzione denominata "calc_centri_2D". Si noti che le posizioni dei centri delle proiezioni dei marker sul piano immagine sono fornite rispetto al sistrma di riferimento dello stesso piano, e non rispetto al sistema di riferimento assoluto della stanza.
La chiamata alla funzione è stata riportata di seguito. Successivamente si procederà all'analisi dettagliata di ingressi ed uscite.

$$[centri, num_{pixel}] = calc\_centri\_2D(marker_{tot}, num_{zeri}, pixel_{dim}, vett_{piano})$$

I parametri che la funzione riceve in ingresso sono:

- Il valore $marker_{tot}$ rappresenta il numero totale dei marker presenti nella stanza. Questo parametro ha la sola funzione di inizializzare le variabili interne alla funzione, al fine di non dover reinizializzare il vettore contenente i centri, all'interno del ciclo for, qualora se ne trovi uno nuovo.

- Il parametro num_{zeri} rappresenta il numero massimo di zeri ammissibile tra due successive serie di uno per considerarle la proiezione dello stesso marker sul piano immagine. In questo modo si tiene conto del fatto che alcuni pixel possono non accendersi nonostante "vedano" il marker.

- In $pixel_{dim}$ viene salvata la dimensione dei pixel del piano immagine. Questo parametro risulta necessario affinchè la posizione del centro calcolato risulti proporzionale alla dimensione del sensore della telecamera.

- il vettore $vett_{piano}$ è il parametro più importante. Esso contiene la rappresentazione vettoriale del piano immagine della telecamera. I pixel "accesi" vengono rappresentati nel vettore da un uno, mentre quelli "spenti" da zero. Per ottenere una rappresentazione corretta la dimensione del vettore è pari al numero di pixel del piano immagine.

Le due uscite della funzione sono:

- Nel primo parametro in uscita *centri* sono salvati i centri delle proiezioni dei marker sul piano immagine della telecamere. Questo vettore contiene i valori che serviranno nella successiva funzione.

- Il secondo vettore in uscita, num_{pixel}, contiene il numero di pixel "accesi" del rispettivo centro, sono nella stessa posizione del vettore. Questo parametro fondamentale è strettamente legato alla distanza del marker dalla telecamera, più il marker è vicino maggiore sarà il numero di pixel che esso "accende". Per questo motivo il valore fornisce una stima sull'incertezza nell'individuazione del centro della proiezione del marker.

L'algoritmo implementato dalla funzione "calc_centri_2D" è stato strutturato in due parti distinte.
La prima parte dell'algoritmo prevede il calcolo dei centri di tutte le serie di uno che sono presenti nel vettore del piano immagine in ingresso. Dal momento che l'accensione dei pixel può essere soggetta ad errore, nel senso che dei pixel si "accendono" quando in realtà non dovrebbero ed altri, al contrario, rimangono "spenti" anche se dovrebbero accendersi, i centri precedentemente trovati non corrispondono alle reali proiezioni dei marker sul piano immagine.
In Figura 9 è riportato un piano immagine esemplificativo sul quale è stata applicata la prima parte dell'algoritmo. Si possono notare i centri calcolati su ogni serie di pixel accesi.

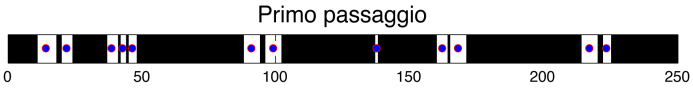

Figura 9: Primo passaggio dell'algoritmo per il calcolo dei centri nel piano.

Per determinare gli effettivi centri delle proiezioni dei pixel sul piano immagine è stata implementata la seconda parte dell'algoritmo. Quando due o piu centri risultano essere sufficientemente vicini, quanto vicini è deciso dal secondo parametro in ingresso alla funzione, essi

vengono riuniti in un unico centro. Il nuovo centro è costituito dalla somma dei centri vicini pesati in base al numero di pixel dai quali sono costituiti. Al centro ricalcolato in questo modo, viene associato il numero di pixel risultante dalla somma dei pixel dei centri dai quali è stato ricavato. Un'altra funzione della seconda parte dell'algoritmo è quella di eliminare i centri che, anche dopo l'operazione di unione dei centri vicini, risultano essere formati da un solo pixel. Infatti si considera che un marker effettivamente visibile dalla telecamere viene proiettato sul piano immagine da almeno due pixel "accesi".
In Figura 10 è riportato il precedente piano immagine dopo essere stato sottoposto alla seconda parte dell'algoritmo. Ora i centri ottenuti dall'operazione di unione corrispondono effettivamente alla proiezioni dei marker sul piano immagine. Si noti come il pixel isolato non venga considerato un centro valido.

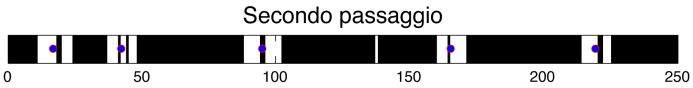

Figura 10: Secondo passaggio dell'algoritmo per il calcolo dei centri nel piano.

Il piano immagine utilizzato nelle precedenti figura ha unicamente lo scopo di chiarire in modo grafico i passaggi dell'algoritmo. Le sue dimensioni sono minori rispetto a quelle dei piani immagine utilizzati nel corso del progetto per rendere più immediata l'analisi dei passaggi.

3.2 Calcolo delle rette

Si utilizza la funzione 'calcola_centri_2D' in modo che in uscita restituisca un insieme di celle pari al numero delle telecamere. Ogni singola cella contiene le varie coordinate dei centri calcolati sul piano immagine, riferiti al sistema di riferimento della telecamera.
Utilizzando per ogni singola cella la funzione 'assoluto' è possibile calcolare la posizione delle coordinate dei centri sul piano immagine rispetto al riferimento assoluto, cioè alla stanza. Oltre alla posizione dei punti sul piano si calcola la posizione del punto focale di ogni telecamera rispetto al sistema di riferimento assoluto.
Come parametri in ingresso alla funzione vengono passati i valori delle singole celle e la corrispettiva posizione nel sistema di riferimento assoluto della telecamera e l'angolo α di rotazione attorno all'asse Z. Oltre a questi viene richiesta la dimensione del piano immagine e la lunghezza focale della telecamera. I dati vengono richiesti con unità di misura di millimetri e radianti. La funzione calcola la matrice di rototraslazione della telecamera rispetto al riferimento assoluto:

$$T_{21} = \begin{bmatrix} \cos(\alpha) & -sen(\alpha) & 0 & x_t \\ sen(\alpha) & \cos(\alpha) & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Quest'ultima viene moltiplicata per la posizione dei centri rispetto alla telecamera e la posizione del punto focale della telecamera ottenendo i punti nel sistema di riferimento assoluto:

$$\begin{bmatrix} x_{assoluto} \\ y_{assoluto} \\ z_{assoluto} \\ 1 \end{bmatrix} = T_{21} \cdot \begin{bmatrix} x_{coordinate} \\ y_{coordinate} \\ z_{coordinate} \\ 1 \end{bmatrix}, \begin{bmatrix} x_{f,assoluto} \\ y_{f,assoluto} \\ z_{f,assoluto} \\ 1 \end{bmatrix} = T_{21} \cdot \begin{bmatrix} x_f \\ y_f \\ z_f \\ 1 \end{bmatrix} \tag{10}$$

Quindi i parametri in uscita sono le coordinate dei centri ed il punto focale per ogni telecamera riferiti al sistema di riferimento assoluto.
Da notare che la coordinata Z sarà sempre pari a zero poichè ci stiamo riferendo ad un piano 2D.

Conoscendo quest'ultimi si può calcolare in modo semplice l'equazione della retta passante per i due punti, riferiti al sistema assoluto utilizzando la funzione 'retta' .
Quest'ultima riceve in ingresso i parametri $x_{c,as}$ $y_{c,as}$ e $x_{f,as}$ $y_{f,as}$ restituendo in uscita il valore dei coefficienti a,b,c dell'equazione della retta in forma implicita:

$$ax + by + c = 0 \tag{11}$$

Con:

$$\begin{aligned} a &= y_{c,as} - y_{f,as} \\ b &= x_{f,as} - x_{c,as} \\ c &= y_{c,as} \cdot (x_{c,as} - x_{f,as}) + x_{c,as} \cdot (y_{f,as} - y_{c,as}) \end{aligned} \tag{12}$$

Conoscendo l'equazione delle rette per ogni singolo centro calcolato dal piano immagine con realtiva incertezza, possiamo plottare il numero delle telecamere, i marker nella stanza e i vari raggi per ogni singola telecamera.
Possiamo notare in Figura 11 la posizione reale dei 4 marker, raffigurati come dei pallini blu, e la posizione delle varie telecamera. La rappresentazione della telecamera è stata leggemente

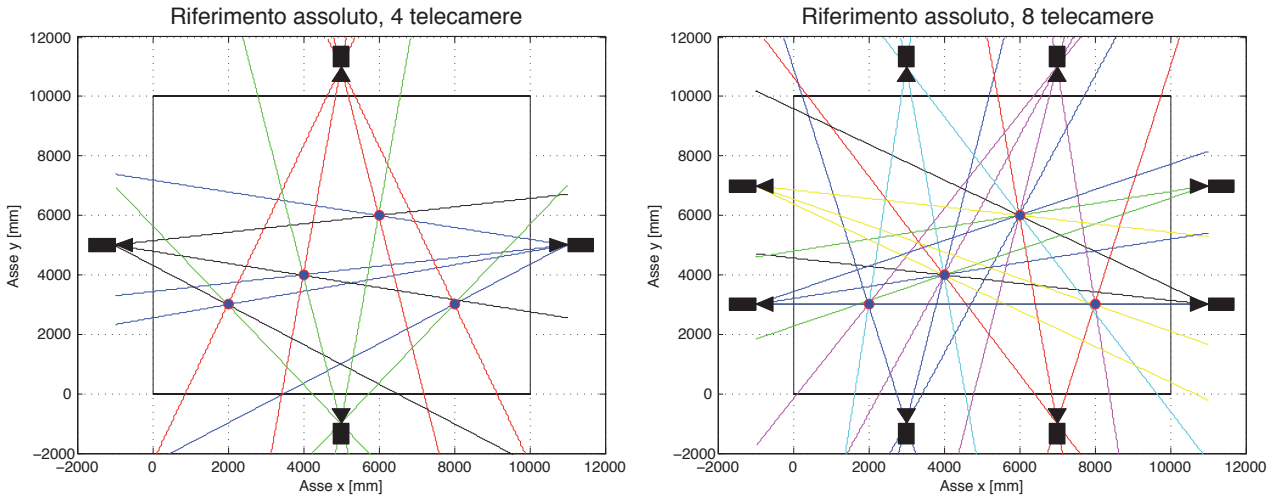

Figura 11: Stanza con 4 e 8 telecamere, 4 marker.

ingrandita per rendere al meglio l'idea, essendo la loro forma scalabile di dimensione a piacimento. Ogni marker genera un raggio nella telecamera se è all'interno dell'angolo di visione di quest'ultima. I raggi generati non passano nel punto esatto dei marker in quanto ricostruiti tenendo conto dell'incertezza generata sul piano immagine.
In questo esempio le telecamere sono impostate con una visuale massima da 0 a 10 metri, cioè tutta la lunghezza della stanza. Questo dato in seguito verrà modificato per attenersi alla realtà.

3.3 Ricostruzione della posizione dei marker nel piano

In questa sezione si vuole ricostruire la posizione dei marker nel piano a partire dalle rette determinate nella precedente sezione. L'obbiettivo è quello di individuare degli algoritmi che possano essere implementati nello schema di comunicazione ad albero, mantenendo contenuti i tempi di elaborazione.
Gli algoritmi implementati verranno presentati e descritti nel dettaglio nelle seguenti sottosezioni. Nella sezione sulle simulazioni ed i confronti ne verranno valutate le prestazioni.

3.3.1 Algoritmi distribuiti

Come prima cosa è fondamentale determinare il numero minimo di rette necessarie per individuare la posizione di un marker nel piano. Si è scelto di utilizzare almeno tre rette per la ricostruzione del marker, in quanto, con due soli raggi, risulta impossibile discriminare tra intersezioni corrispondenti ad un marker reale o meno, come evidenziato in Figura 12.

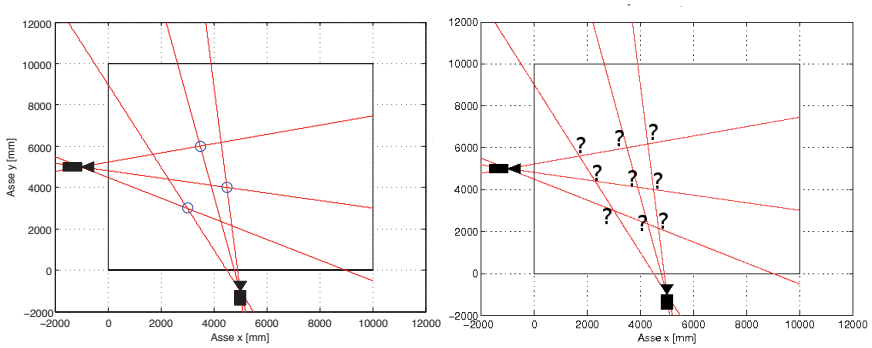

Figura 12: Impossibilità di ricostruzione della posizione dei marker con 2 telecamere

Dopo aver determinato il numero minimo di rette necessarie per la ricostruzione di un marker nel piano, risulta di fondamentale importanza individuare un criterio di assegnazione della posizione del marker rispetto ai raggi dai quali è determinato.
Si è scelto di determinare la posizione del marker nel piano mediante la tecnica dei minimi quadrati. Si consideri la posizione del marker determinata da

$$P = \begin{bmatrix} x & y \end{bmatrix} \; : \; \min_{x,y} \sum d_i^2 \text{ con } d_i^2 = \frac{(a_i x + b_i y + c_i)^2}{a_i^2 + b_i^2}$$

Sviluppando i quadrati e definendo

$$k_i = a_i^2 + b_i^2 \; , \; \overline{a}_i = \frac{a_i}{k_i} \; , \; \overline{b}_i = \frac{b_i}{k_i} \; , \; \overline{c}_i = \frac{c_i}{k_i}$$

$$A = \sum_i \overline{a}_i^2 \; , \; B = \sum_i \overline{b}_i^2 \; , \; C = \sum_i \overline{c}_i^2 \; , \; D = \sum_i \overline{a}_i \overline{b}_i \; , \; E = \sum_i \overline{a}_i \overline{c}_i \; , \; E = \sum_i \overline{b}_i \overline{c}_i$$

Si ottiene

$$\frac{\partial}{\partial x} = 2Ax + 2Dy + 2E$$
$$\frac{\partial}{\partial y} = 2By + 2Dx + 2F$$

Dalle quali, ponendo le precedenti derivate pari a zero, si ottengono le coordinate del punto di minima distanza

$$y = \frac{DE - FA}{BA - D^2} \ , \ x = \frac{-Dy - E}{A}$$

Si vuole far notare che è possibile estendere l'algoritmo individuato anche al caso in cui le rette siano maggiori di tre. Anche in questa situazione la posizione del marker ricostruita nel piano risulta essere quella del punto di minima distanza fra le rette utilizzate.

L'algoritmo è stato implementato nella funzione denominata "ric_2D", la cui chiamata è stata riportata di seguito

$$[raggi, punti] = ric\_2D(raggi_1, punti_1, raggi_2, punti_2, \epsilon, vis_{min}, vis_{max}, larghezza, altezza)$$

I paramentri in ingresso alla funzione sono

- $raggi_1$ e $raggi_2$ rappresentano i due insiemi di raggi non ancora usati nella ricostruzione.
- $punti_1$ e $punti_2$ contengono i punti ricostruiti dalla funzione implementata nei precedenti livelli dell'albero.
- ϵ è la distanza massima tra la posizione stimata del punto e le rette utilizzate per la ricostruzione entro cui il marker viene considerato esistente.
- vis_{min}, vis_{max}, $larghezza$, $altezza$ sono parametri corrispondenti al campo visivo della videocamera ed alle dimensioni della stanza. Sono valori necessari per evitare di ricostruire punti non esistenti verificando a posteriori la validità della stima.

Le uscite della funzione risultano essere

- Il parametro *raggi* contiene l'insieme delle rette non ancora processate.
- Il valore *punti* contiene l'insieme dei punti ricostruiti.

Di seguito viene descritto nel dettaglio il comportamento della funzione nelle sue due differenti implementazioni.
Nella parte iniziale della funzione si esegue un confronto tra $raggi_1$ e $punti_2$ e tra $raggi_2$ e $punti_1$ in cui si contralla se un raggio 'passi' vicino ad un punto già ricostruito. In tal caso si assume che quel raggio sia stato generato dalla proiezione del medesimo marker, per affrontare questa situazione sono stati individuati due differenti approcci.
La prima implementazione prevede l'utilizzo di tre rette per la determinazione della posizione del marker nel piano. Per questo motivo il nuovo raggio non viene preso in considerazione e viene eliminato.
Nel secondo approccio la nuova retta viene utilizata per migliorare la stima della posizione del marker nel piano. In questo caso la posizione del punto viene determinata dall'algoritmo ai minimi quadrati applicato al nuovo insieme di rette che individuano il punto.
Ci si aspetta che la prima implementazione abbia tempi di calcolo inferiori rispetto alla seconda, a discapito però di una ricostruzione meno accurata della posizione del marker. Inoltre la struttura dati, nella seconda implementazione, risulta essere più complessa. Questo fatto è

dovuto alla necessità di salvare non solo le coordinate del marker ricostruito ma anche i dati di tutti i raggi utilizzati per il calcolo del punto stesso.
Nella parte successiva della funzione, partendo dal presupposto che non ci possono essere tre rette che diano luogo ad un punto valido all'interno dello stesso gruppo di raggi, si processano prima due raggi appartenenti a $raggi_1$ e uno da $raggi_2$ e, successivamente, si esegue il viceversa. Si presta inoltre attenzione a non confrontare raggi relativi alla stessa telecamera evitando così inutili sprechi di tempo.
Se dallo stimatore ai minimi quadrati si ottiene un punto la cui distanza massima tra esso e le rette che lo hanno generato è minore di ϵ esso viene individuato come un marker e quindi viene salvata la posizione del marker nel piano, insieme ad i relativi raggi nel caso della seconda realizzazione, e si epurano $raggi_1$ e $raggi_2$ dei dati usati nella ricostruzione.

3.3.2 Algoritmi distribuiti pesati

Anche in queste implementazioni si considerano necessarie almeno tre rette per la ricostruzione di un marker nel piano.
In questa sottosezione si considera un algoritmo differente per la ricostruzione del marker nel piano. Questo approccio prevede l'utilizzo della tecnica ai minimi quadrati pesati per la ricostruzione della posizione del marker, determinata da

$$P = \begin{bmatrix} x & y \end{bmatrix} \; : \; \min_{x,y} \sum_i \alpha_i d_i^2 \text{ con } \sum_i \alpha_i = 1$$

Si è assunto di pesare maggiormente l'informazione proveniente dalle tecamere che inquadrano da più vicino il marker. Per fare ciò si valuta la dimensione in pixel della proiezione del marker sul piano immagine che viene fornita dalla funzione calc_centri_2D.
Si consideri la retta i-esima, a cui è associato un numero di pixel pari a np_i, il peso viene calcolato come

$$\alpha_i = \frac{np_i}{\sum_j np_j}$$

Anche in questo algoritmo, la funzione di ricostruzione del marker nel piano prevede il confronto fra i punti individuati ed i raggi provenienti dall'altro ramo dell'albero.
Come nella precedente sottosezione, si considerano due differenti implementazione dell'algoritmo qualora il numero di rette che individuano il marker sia maggiore di tre.
Nella prima implementazione vengono utilizzati solamente tre raggi per la ricostruzione del marker, per questo motivo le nuove rette individuate non vengono prese in considerazione e vanno eliminate.
Nel secondo algoritmo le nuove rette che individuano il marker vengono utilizzate per migliorarne la stima della posizione. In questo approccio la posizione del marker viene ricostruita applicando l'algoritmo ai minimi quadrati pesati al nuovo set di rette.
Da questo nuovo approccio ci si aspetta un miglioramento nella precisione di ricostruzione della posizione del punto a fronte di un piccolo incremento dei tempi di calcolo rispetto alle rispettive realizzazioni adottate in precedenza.

3.3.3 Algoritmo centralizzato

Per poter svolgere dei confronti tra l'approccio centralizzato e quello distribuito si rende necessaria l'implementazione di un algoritmo che elabora i dati in modo centralizzato.

In questo caso i frame provenienti da tutte le telecamere vengono salvati e, successivamente, si procede alla ricostruzione processando tutti i dati insieme. La posizione del marker nello spazio viene ricostruita mediante l'algoritmo ai minimi quadrati applicato ad almeno quattro rette. Si è scelto di utilizzare quattro raggi in quanto, con un numero minore di raggi, si ottenevano troppi punti non validi. Questo fatto è stato evidenziato anche da un'azienda di motion capture, nella quale si ricorre anche a più di cinque raggi per individuare un marker nello spazio.

3.4 Rappresentazione dell'incertezza

In questa sezione si vuole fornire una rappresentazione dell'incertezza nella ricostruzione dei marker nel piano.
Dal momento che il marker si trova in uno spazio bidimensionale si è scelto di caratterizzare l'incertezza mediante un'area con centro nel marker ricostruito. Per semplicità si è scelto di rappresentare l'incertezza nel piano per mezzo di una circonferenza, in quanto necessita solamente di un centro e di un raggio per essere caratterizzata. Quindi, nel piano, l'incertezza risulta essere determinata da tre valori.
Il centro della circonferenza è dato dai valori sull'asse delle ascisse e su quello delle ordinate del

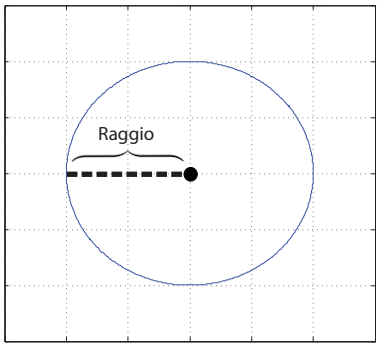

Figura 13: Raggio di incertezza nel caso bidimensionale.

marker ricostruito. Il raggio è determinato dalla distanza tra il marker e le rette dalle quali è stato ricavato. A seconda del tipo di implementazione utilizzata per la ricostruzione del marker nel piano, il calcolo del raggio della circonferenza varia.

- Si consideri l'implementazione della funzione per la ricostruzione dei marker nel piano che fa uso di tre rette per la determimazione della posizione del marker. In questa situazione il raggio della circonferenza è dato dalla maggiore fra le distanze tra il marker individuato e le rette dalle quali è stata determinata la sua posizione.
  Per quanto riguarda l'implementazione che utilizza più rette per migliorare la ricostruzione del marker nel piano, il raggio della circonferenza rappresentante l'incertezza è dato dalla maggiore fra le distanze tra il marker ricostruito e le rette che ne hanno determinato la posizione. Vedi Figura 14.

- Si consideri ora l'implementazione della funzione per la ricostruzione dei marker nel piano che fa uso di tre rette per determinare la posizione del punto nel piano, pesando in funzione del numero di pixel associato alle rette. Anche in questo caso il raggio di incertezza è dato dalla maggiore delle distanze tra il marker ricostruito e le tre rette dalle quali è stato determinato.
  Lo stesso principio è stato usato per determinare il raggio di incertezza nella funzione che pesa le rette e usa tutte quelle che individuano il marker. 15

Nella sezione riguardante i confronti e le simulazioni si procederà ad una analisi più approfondita della rappresentazione dell'incertezza proposta.

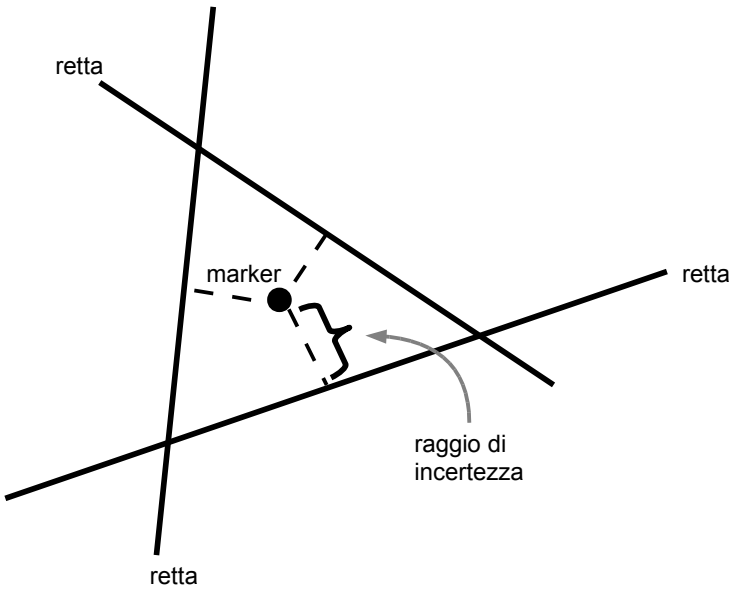

Figura 14: Raggio di incertezza con raggi non pesati.

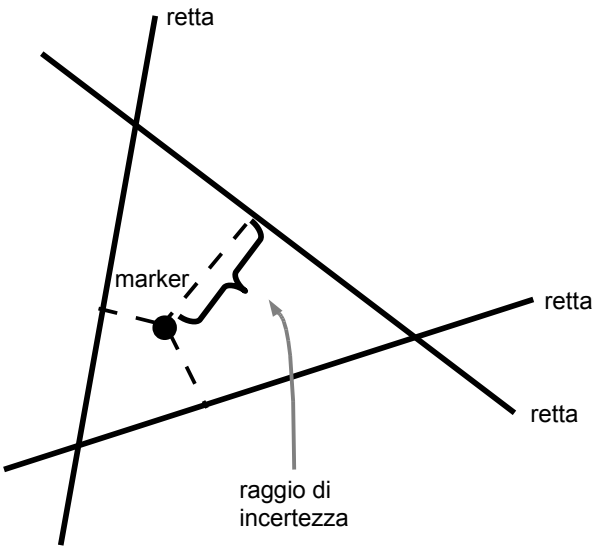

Figura 15: Raggi di incertezza con raggi pesati.

4 Confronti e simulazioni nel piano

4.1 Implementazione delle simulazioni nel piano

Dalla funzione principale si possono utilizzare e simulare le varie funzioni che successivamente verranno descritte nel dettaglio.
Le varie simulazioni vengono svolte in un sistema di riferimento assoluto, cioè la stanza di dimensione 10*10 metri dove sono presenti i marker. Tale misura è eventualmente scalabile. Da questa funzione principale, si sceglie la posizione della telecamera e la relativa inclinazione rispetto alla stanza.
Con il parametro camera_pos = $[x_t \ y_t \ z_t \ \alpha]$ si fissano le coordinate x,y,z della telecamera rispetto al sistema di riferimento assoluto e la rotazione α rispetto all'asse z. Quest'ultima è equivalente alla rotazione attorno all'asse y con la notazione di Figura 2.
Questa notazione è stata introdotta per essere coerenti con gli assi del simulatore 2D sviluppato dal gruppo 2. Quindi il piano immagine corrispondente risulta sul piano formato dall'asse x e z della telecamera e l'asse y uscente.
Essendo nel caso bidimensionale il valore di z_t sarà sempre nullo per tutte le telecamere.

Inoltre è possibile impostare la dimensione del sensore e il numero di pixel. Per le simulazioni abbiamo impostato una lunghezza di 10 millimetri del sensore e un numero di pixel pari a 2000, ottenendo quindi una dimensione del singolo pixel pari a 0,005 millimetri. La lunghezza focale f è stata impostata come la metà della dimensione del sensore come da specifiche del progetto.

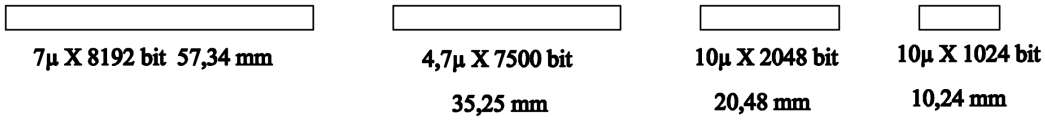

Figura 16: Dimensioni sensore lineare.

Con queste dimensioni ci siamo attenuti a quelle che possono essere le misure reali da utilizzare, come viene raffigurato in Figura 16. Infatti il caso bidimensionale può essere pensato nella realtà, come l'utilizzo di un sensore lineare fomato da un pixel in altezza e 2000 pixels in larghezza. Nell'appendice possiamo trovare un'approfondimento sulle dimesioni reali dei sensori delle telecamere.
Dalla funzione principale si imposta anche la lunghezza massima e minima di visione della telecamera e il valore della varianza sigma con la quale i piani immagine verranno 'sporcati' dalla funzione 'genera piano' che verrà spiegata nel dettaglio in seguito. Infine si impostano il numero di marker e la loro relativa posizione nel sistema di riferimento assoluto.
Tutti questi parametri sono chiaramente impostabili e scalabili a proprio piacimento secondo le specifiche di progetto.

A questo punto si utilizza la funzione fornita dal gruppo 2:

$$[coordinate, hidden, distance] = camera\_simulation\_2d(\alpha, camera\_pos, punti, K, f, width)$$

I parametri in ingresso sono:

- α: angolo di rotazione attorno all'asse z;
- *camera_pos*: posizione della telecamera rispetto al sistema di riferimento assoluto;
- *punti*: posizione dei marker nel sistema di riferimento assoluto;
- K: matrice dei parametri intrinseci delle telecamere:

$$K = \begin{bmatrix} k_x & 0 & v \\ 0 & k_y & u \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

  Con k_x larghezza dei pixel, k_y altezza dei pixel, u, v rappresentano la traslazione dell'origine del piano immagine rispetto al centro ottico, espresse nella stessa unità di misura di k_x e k_y;
- *width*: dimesione del sensore in pixels.

Essa restituisce in uscita le coordinate sul piano immagine dei rispettivi marker, dati dalla loro proiezione. Restituisce anche la distanza dal piano immagine di ogni singolo marker e un valore booleano che indica se il marker si trova davanti o dietro alla telecamera.
In questo modo usando la funzione su ogni telecamera si trovano le coordinante e le distanze da poter utilizzare con la funzione 'genera piano' .
La funzione 'genera_piano' utilizza le coordinate calcolate sul piano immagine della proizione dei marker e la distanza di quest'ultimi dalla telecamera, per poter generare dei piani immagini il più possibile simili alla realtà. Le coordinate e la distanza vengono fornite, come visto in precedenza, dal simulatore. La funzione:

$$[piano\_gen] = genera\_piano(P, width, dist\_min, dist\_max, \sigma)$$

riceve in ingresso:

- P: coordinate calcolate sul piano immagine e la distanza dalla telecamera di ogni marker;

$$P = \begin{bmatrix} x_{coord\_1} & dist_1 \\ \vdots & \vdots \\ x_{coord\_n} & dist_n \end{bmatrix}$$

- *width*: dimesione del sensore in pixels;
- *dist_min*: distanza minima di visione della telecamera;
- *dist_max*: distanza massima di visione della telecamera;
- σ: varianza dell'errore da utilizzare per 'sporcare' il piano immagine generato.

In uscita restituisce:

- *piano_gen*: piano immagine composto da vettore 1xwidth contenente valori 0 e 1.

Le misure della distanza massima e minima di visione della telecamera vengono espresse in millimetri.
Per generare l'errore sul piano immagine, viene considerato che la proiezione dei marker sul piano ha una dimensione che varia da 2 pixels a 50 pixels come da specifiche.
Questi valori vengono considerati con una distanza massima e minima di visione della telecamera che varia rispettivamente da 2 metri a 5 metri.
Quindi, diminuendo il parametro σ, il piano immagine presenterà un rumore maggiore, viceversa aumentandolo il rumore viene ridotto. Ad esempio, un marker a distanza 2 metri dalla telecamera presenta un numero di pixels sul piano immagine pari a 50, con un rumore che varia al variare di σ.

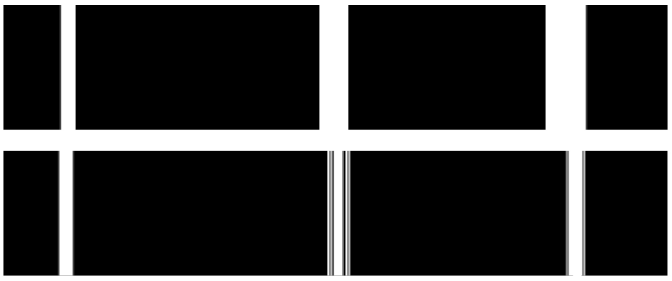

Figura 17: Sopra: sezione di piano immagine con poco rumore. Sotto: sezione di piano immagine molto rumore.

Nella Figura 17 viene rappresentato una sezione dello stesso piano immagine, con poco rumore e con molto rumore. La sezione non comprende tutta la lunghezza del piano immagine, ma solo un'ingrandimento di una parte con la proiezione di tre marker, per comprendere al meglio il rumore aggiunto sul piano.

A questo punto si procede con la simulazione delle funzioni implementate per la ricostruzione della posizione dei marker nel piano.
Vengono di seguito riportati i risultati ottenuti considerando sedici telecamere e dodici marker. Si assume inoltre la visuale della telecamera da uno a sette metri.
Infine è stato posto epsilon pari a 10 mm.
Per ognuna delle soluzioni di ricostruzione implementate vengono racchiusi in una tabella i dati dei valori ricostruiti ed in un'altra i tempi di esecuzione. I tempo sono stati rilevati e mediati su cento simulazioni di ricostruzione.
In Figura 18 viene riportata la disposizione di telecamere e marker ed una relativa ricostruzione, I marker reali sono rappresentati mediante un 'o' blu mentre la ricostruzione con un '+' rosso.

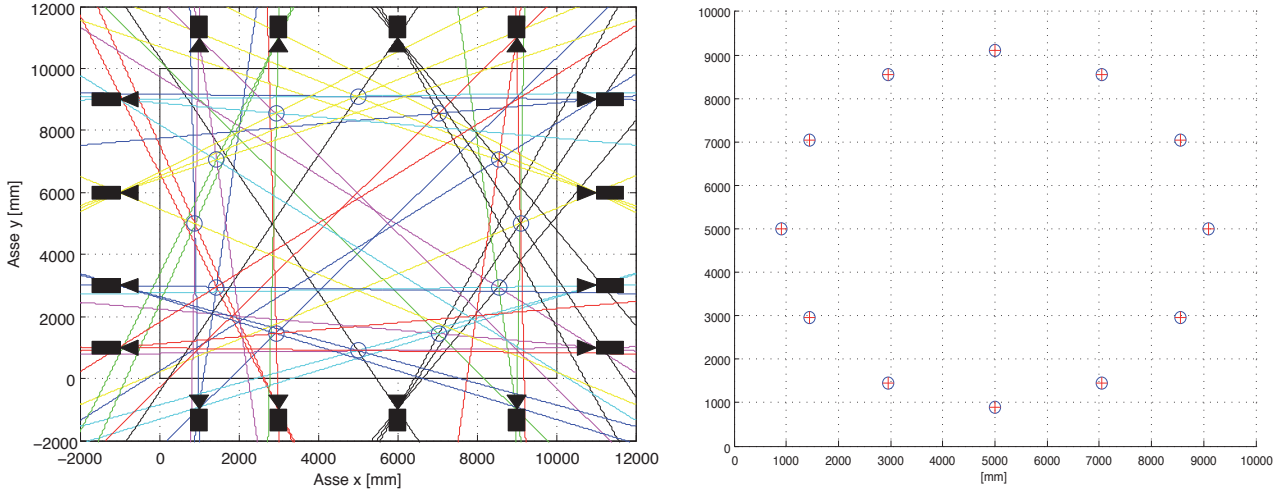

Figura 18: Ricostruzione di 12 marker con 16 telecamere, approccio distribuito.

4.1.1 Implementazione distribuita con eliminazione dei raggi

Eseguendo la ricostruzione con i parametri sopra riportati si ottiene un errore medio di ricostruzione di 2.6779 mm come si potrebbe ricavare dalla seguente tabella dove vengono riportate le cordinate x e y della posizione reale del marker, la sua stima x_{ric} e y_{ric}, l'errore di ricostruzione del marker ed il residuo nell'algoritmo ai minimi quandrati rappresentante il raggio della circonferenza di incertezza.
Essendo necessari almeno tre raggi per individuere un punto, nei nodi più profondi dell'albero non viene effettuata nessuna operazione tranne l'unione dei dati di due telecamere; per questo i tempi vengono riportati a partire dal secondo livello, dal basso, dell'albero.

Secondo passo albero	Terzo passo albero	quarto passo albero
0.0243	———	———
0.0237	0.1409	0.0729
0.0238	0.1458	———
0.0237	———-	———

x	y	x_{ric}	y_{ric}	errore	residuo
5000	9100	5003.4152	9102.6851	4.3443	0.95987
7050	8550.7042	7047.6276	8551.2461	2.4335	0.0086121
8550.7042	7050	8552.072	7048.6939	1.8913	0.58894
9100	5000	9095.4278	4997.1329	5.3967	0.80624
8550.7042	2950	8550.9562	2951.1401	1.1677	0.58564
7050	1449.2958	7047.8014	1447.2117	3.0295	0.49172
5000	900	4998.8195	898.8015	1.6822	0.82146
2950	1449.2958	2949.1336	1447.558	1.9418	0.27091
1449.2958	2950	1447.6159	2950.6573	1.804	1.63
900	5000	904.6968	4998.2502	5.0121	0.72466
1449.2958	7050	1447.5954	7051.2974	2.1389	2.3642
2950	8550.7042	2950.1783	8551.9847	1.2929	0.51071

Tabella 1: Risultati della simulazione dell'algoritmo distribuito che elimina i raggi.

Le operazioni effettuate allo stesso livello dell'albero vengono svolte in modo distribuito e contemporaneamente per cui come stima del tempo richiesto per la ricostruzione è possibile considerare la somma dei tempi massimi impiegati per ciascun livello.
Ad esempio in questo caso il tempo richiesto è pari a 0.0243+0.1458+0.0729=0.243 secondi.

4.1.2 Implementazione distribuita con rivalutazione dei raggi

Mediante questo algoritmo si ottiene un errore medio pari a 1.9265 mm che risulta inferiore rispetto al caso precente.

x	y	x_{ric}	y_{ric}	errore	residuo
5000	9100	4999.0992	9098.6813	1.597	3.6877
7050	8550.7042	7047.6276	8551.2461	2.4335	0.0086121
8550.7042	7050	8552.072	7048.6939	1.8913	0.58894
9100	5000	9099.2945	5001.2734	1.4558	6.6199
8550.7042	2950	8550.9562	2951.1401	1.1677	0.58564
7050	1449.2958	7047.8014	1447.2117	3.0295	0.49172
5000	900	4998.7168	900.4756	1.3685	4.7762
2950	1449.2958	2949.1336	1447.558	1.9418	0.27091
1449.2958	2950	1447.6159	2950.6573	1.804	1.63
900	5000	902.1544	4997.9155	2.9977	1.866
1449.2958	7050	1447.5954	7051.2974	2.1389	2.3642
2950	8550.7042	2950.1783	8551.9847	1.2929	0.51071

Tabella 2: Risultati della simulazione dell'algoritmo distribuito che tiene conto dei raggi.

Il tempo di esecuzione risulta 0.248 secondi ovvero leggermente superiore al caso precendente. Questo è dovuto al maggior numero di operazioni e alla più complessa struttura dati necessari

all'implementazione di questo algoritmo.

Secondo passo albero	Terzo passo albero	quarto passo albero
0.0245	———	———
0.0239	0.1448	0.0758
0.0239	0.1477	———
0.0238	———-	———

4.1.3 Implementazione distribuita e pesata con eliminazione dei raggi

L'errore medio risulta 2.6687 mm e quindi si nota un miglioramento nella precisione di ricostruzione dei marker rispetto a pesare ugualmente tutti i residui.

x	y	x_{ric}	y_{ric}	errore	residuo
5000	9100	5003.6893	9102.1744	4.2825	1.5394
7050	8550.7042	7047.6247	8551.2452	2.4361	0.01096
8550.7042	7050	8552.1972	7048.7317	1.959	0.66193
9100	5000	9095.8567	4996.9027	5.173	1.2929
8550.7042	2950	8551.0446	2950.8765	0.94026	0.63338
7050	1449.2958	7047.7575	1447.3487	2.9699	0.58346
5000	900	4999.176	898.6548	1.5775	0.99748
2950	1449.2958	2949.0716	1447.4632	2.0544	0.26113
1449.2958	2950	1447.6653	2950.2523	1.65	1.732
900	5000	904.567	4998.5646	4.7873	0.87996
1449.2958	7050	1447.41	7052.018	2.762	2.2607
2950	8550.7042	2950.0594	8552.136	1.4331	0.50934

Tabella 3: Risultati della simulazione dell'algoritmo distribuito e pesato che elimina i raggi.

Il tempo impiegato dall'algoritmo risulta pari a 0.2532 secondi quindi leggermente superiore alla realizzazione distribuita che pesa ugualmente tutti i raggi. Anche in questo caso l'aumento del tempo è dovuto sia al maggior numero di parametri da memorizzare che alle ulteriori operazioni da effettuare.

Secondo passo albero	Terzo passo albero	quarto passo albero
0.0264	———	———
0.0254	0.1459	0.0756
0.0251	0.1512	———
0.0253	———-	———

4.1.4 Implementazione distribuita e pesata con rivalutazione dei raggi

In quest'ultima realizzazione si è ottenuto un errore medio di 1.9187 mm. Questo metodo risulta il più preciso per quanto riguarda la ricostruzione del punto anche se, come come riportato nella tabella dei tempi, la controparte è un tempo di esucuzione lievemente superiore.
Il tempo richiesto è di 0.2587 secondi.

x	y	x_{ric}	y_{ric}	errore	residuo
5000	9100	5001.9321	9098.8892	2.2287	7.1788
7050	8550.7042	7047.6247	8551.2452	2.4361	0.01096
8550.7042	7050	8552.1972	7048.7317	1.959	0.66193
9100	5000	9099.1058	4998.6457	1.6229	6.1951
8550.7042	2950	8551.0446	2950.8765	0.94026	0.63338
7050	1449.2958	7047.7575	1447.3487	2.9699	0.58346
5000	900	4998.1996	900.5914	1.895	5.0223
2950	1449.2958	2949.0716	1447.4632	2.0544	0.26113
1449.2958	2950	1447.6653	2950.2523	1.65	1.732
900	5000	901.9083	4999.8742	1.9125	6.5417
1449.2958	7050	1447.41	7052.018	2.762	2.2607
2950	8550.7042	2950.0594	8552.136	1.4331	0.50934

Tabella 4: Risultati della simulazione dell'algoritmo distribuito e pesato che tiene conto dei raggi.

Secondo passo albero	Terzo passo albero	quarto passo albero
0.0271	———	———
0.0255	0.1481	0.0786
0.0254	0.1530	———
0.0255	———-	———

4.1.5 Implementazione centralizzata

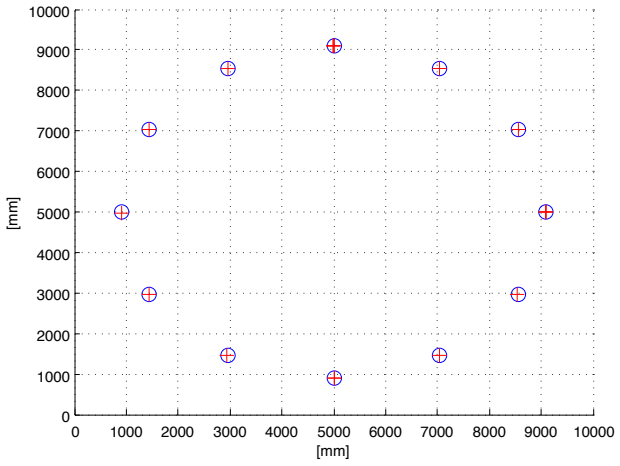

Figura 19: Ricostruzione centralizzata dei marker nel piano mediante quattro raggi.

Ricostruendo i punti in maniera centralizzata si è ottenuto un errore medio di 4.5350 mm mentre il tempo di elaborazione dell'algoritmo centralizzato risulta nettamente superiore al caso decentralizzato e pari a circa 43 secondi.

x	y	x_{ric}	y_{ric}	errore
5000	9100	4993.32	9097.6847	7.0699
7050	8550.7042	7047.6048	8548.394	3.3277
8550.7042	7050	8554.2069	7047.1182	4.5359
9100	5000	9099.6173	5003.3684	3.3901
8550.7042	2950	8547.5803	2952.5864	4.0557
7050	1449.2958	7047.8221	1448.2317	2.424
5000	900	5004.6732	899.01143	4.7766
2950	1449.2958	2948.7082	1449.1979	1.2955
1449.2958	2950	1447.8607	2951.2343	1.8929
900	5000	900.24677	4982.4534	17.5483
1449.2958	7050	1450.985	7049.754	1.707
2950	8550.7042	2952.2808	8551.4413	2.3969

Tabella 5: Risultati della simulazione dell'algoritmo centralizzato.

4.1.6 Risultati dei confronti nel piano

Un problema dell'approccio distribuito risulta essere la possibile ricostruzione del medesimo marker due o più volte. Questo accade qualora un marker venga individuato e ricostruito da rette appartenenti a rami diversi dell'albero.
Una possibile soluzione è eseguire un controllo alla fine del processo dove si individuano i punti che si riferiscono al medesimo marker e se ne opera una media. Nel caso in cui al punto ricostruito siano associati i raggi usati per calcolarlo, come nel secondo metodo di ricostruzione illustrato, è possibile ricalcolare il punto a partire da tutti i raggi.
Confrontando i risultati ottenuti nelle implementazioni degli algoritmi distribuiti con quello centralizzato si nota immediatamente il divario dei tempi di elaborazione. Ci si aspettava un netto miglioramento delle prestazioni mediante l'approccio distribuito, tuttavia la differenza è molto più elevata del previsto. Questo può essere dovuto principalmente a due motivi. Il primo deriva dal fatto che nell'implementazione centralizzata sono state utilizzate quattro rette per la ricostruzione dei marker nel piano. Il motivo principale, tuttavia, è dovuto al fatto che nell'algoritmo centralizzato le rette utilizzate per la ricostruzione di un marker non vengono eliminate, ma continuano ad essere confrontate. Per questo motivo il divario tra i tempi di elaborazione tra implementazione distribuita e quella centralizzata aumenta al crescere della dimensione del problema.
Per quanto riguarda l'errore di ricostruzione medio si nota che l'ordine di grandezza è lo stesso sia nell'approccio centralizzato che in quelli distribuiti, anche se, in questi ultimi, risulta essere mediamente minore.
Ricordando che il residuo nelle tabelle corrisponde al raggio della circonferenza rappresentante l'area di incertezza nell'individuazione del marker, si nota che, nella maggior parte dei casi, il marker reale non si trova all'interno dell'area di incertezza individuata. Questo significa che quest'ultima non è una buona caratterizzazione dell'incertezza.

4.2 Relazione tra numero di pixel ed errore di ricostruzione

In questa fase del progetto si vuole individuare quale relazione sussiste tra il numero di pixel della proiezione del marker sul piano immagine e la distanza tra il marker reale e quello calcolato. Il numero di pixel è un parametro in uscita alla funzione "calc_centri_2D", ed è proporzionale alla distanza del marker dalla telecamera. La distanza tra il marker reale ed il marker ricavato mediante l'utilizzo delle funzione implementate rappresenta l'errore di ricostruzione del punto nel piano.
La logica suggerisce che più il marker è vicino alla telecamera, minore dovrebbe essere l'errore nella sua ricostruzione. Questo fatto potrebbe essere usato per pesare l'informazione proveniente dai piani immagine sui quali viene proiettato il marker, in relazione alla distanza del marker dalla rispettiva telecamera.
Per individuare il tipo di relazione, l'ideale sarebbe stato avere a disposizione una serie di piani immagine reali, provenienti da telecamere utilizzate per il motion capture.
Dal momento che questo non è stato possibile, si è deciso di adottare un approccio empirico. Questo approccio prevede l'utilizzo di due sole telecamere per la ricostruzione di una serie di marker posti ad uguale distanza dalle telecamere. Mediante l'utilizzo del simulatore implementato dal gruppo due ed alla funzione per generare i piani, sono stati generati i piani immagine delle due telecamere. Grazie alla funzione per il calcolo dei centri ed a quella per il calcolo dei raggi è stato possibile ricavare le rette passanti per il centro delle proiezioni ed il punto focale della telecamera. L'intersezione fra le rette provenienti dalle due telecamere fornisce la posizione dei marker ricostruiti.
Si è deciso di effettuare la simulazione prima con un numero di marker pari a dieci, poi con un numero di marker pari a diciannove.
In Figura 20 sono riportate le posizioni dei marker reali, circonferenza blu, e quella dei marker ricostruiti, croce blu. A causa delle sue dimensioni ridotte, l'errore di ricostruzione non risulta apprezzabile in figura. Tuttavia esso verrà analizzato nel dettaglio successivamente.

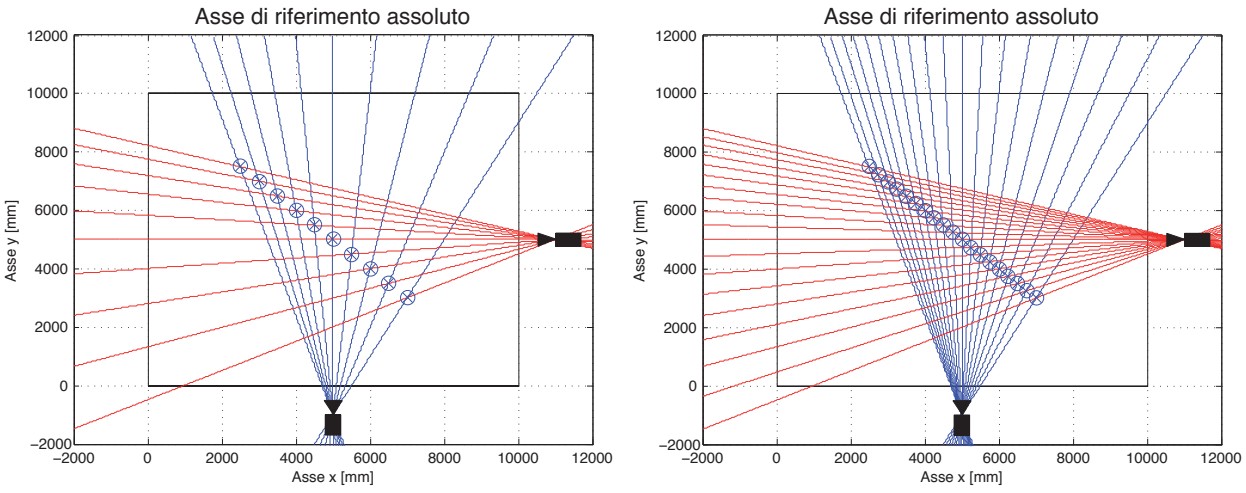

Figura 20: Approccio empirico con dieci marker nella prima immagine e diciannove marker nella seconda.

Dalla conoscenza della posizione dei marker ricostruiti e di quelli originali si calcola l'errore di ricostruzione. Per ottenere un risultato significativo la simulazione descritta sopra è stata ripetuta cento volte.
In Figura 21 sono riportati i risultati ottenuti dalle cento simulazioni effettuate. I pallini di colore diverso corrispondono ad un marker diverso, ogni pallino dello stesso colore rappresenta una diversa simulazione.
In ordinata è riportato l'errore di ricostruzione in millimetri, mentre in ascissa è stato riportato il numero di pixel che costituiscono la proiezione del marker sul piano immagine. La rette spezzata di colore nero è costituita dalla media delle cento realizzazioni, per questo motivo risulta essere di notevole interesse e verrà analizzata nel dettaglio.

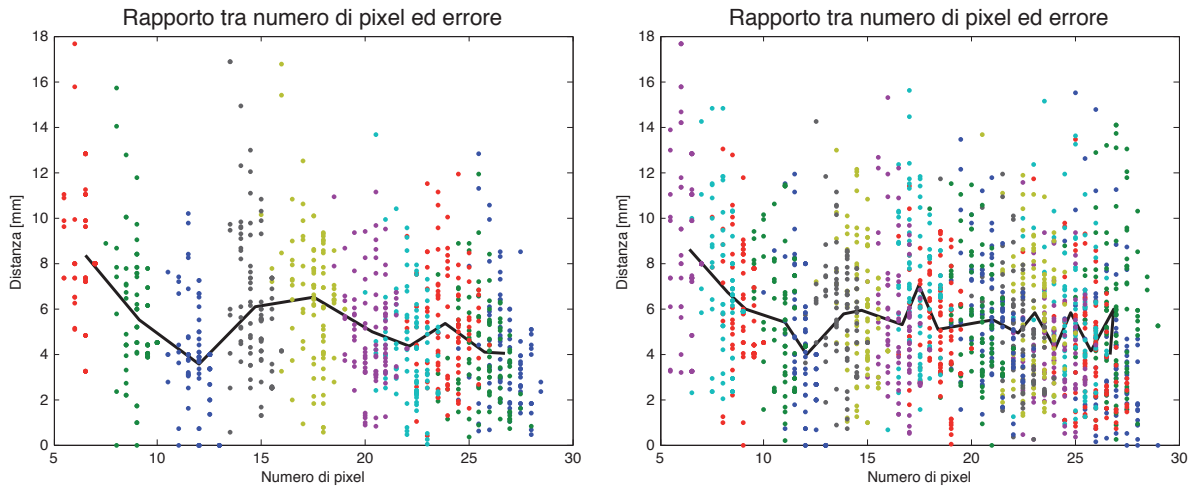

Figura 21: Grafico del rapporto tra numero di pixel ed errore di ricostruzione per dieci e per diciannove marker.

Osservando l'andamento della media delle cento realizzazioni, sia per quanto riguarda la simulazione con dieci marker che per quella con diciannove marker, si nota che l'errore di ricostruzione tende a diminuire al crescere del numero di pixel della proiezione del marker sul piano immagine. Questo comportamento è prevedibile, infatti l'errore di accensione dei pixel risulta essere più pesante quando la proiezione del marker è costituita da pochi pixel.
Si ricordi che il numero di pixel associato al centro sul piano immagine è proporzionato alla distanza del marker dalla telecamera. Per questo motivo si ha che al diminuire della distanza del marker dalla telecamera si ha una diminuzione dell'errore di ricostruzione.
Procedendo con l'analisi della simulazione effettuata, si nota che la retta della media delle cento realizzazioni risulta soggetta ad oscillazioni inaspettate. A questo comportamento, non previsto, si è tentato di dare una spiegazione.
Un motivo può essere dovuto al fatto che la funzione per generare i piani non funziona in modo adatto, fornendo in uscita dei piani immagine che non sono sporcati in modo gaussiano. Un altro motivo di questo comportamento può essere il fatto che non viene considerato l'errore dovuto alla discretizzazione della posizione del centro della proiezione del marker sul piano immagine. Quest'ultima spiegazione è supportata dal fatto che l'errore di ricostruzione,

osservando la retta nei due casi, risulta essere legato alla posizione del marker e non solo alla sua distanza dalle telecamere. Per osservare questo comportamento sono state riportate le due simulazioni con differente numero di marker.
In conclusione si è deciso di adottare dei pesi proporzionali al numero di pixel. Dal momento che questo approccio non ha portato nessun vantaggio verranno adottati dei pesi direttamente proporzionali al numero di pixel anche nel caso tridimensionale.

5 Progetto nello spazio

Dopo aver analizzato il caso bidimensionale, si è passato al problema nello spazio tridimensionale, quello di effettivo interesse.
Anche in questo caso il problema da affrontare è stato suddiviso in tre sottoproblemi distinti, aventi la stessa struttura di quelli trattati nel caso bidimensionale, il cui schema a blocchi è riportato in Figura 22

Figura 22: Schema a blocchi delle funzioni da implementare nello spazio.

- **Calcolo dei centri:** il primo algoritmo da implementare è quello che fornisce la posizione dei centri delle proiezioni dei marker sul piano immagine della telecamera.
- **Calcolo delle rette:** il secondo problema da affrontare prevede l'implementazione di un algoritmo per la determinazione delle rette passanti per i centri calcolati nella precedente funzione ed il centro focale della telecamera.
- **Ricostruzione della posizione dei marker nello spazio:** l'ultimo problema da affrontare è quello di maggior interesse. In questa fase sono stati implementati diversi algoritmi per determinare la posizione dei marker nello spazio, a partire dalle informazioni ricavate nei precedenti punti per ogni telecamera presente.

Nelle seguenti sottosezioni verranno esposte nel dettaglio le tecniche adottate per risolvere i tre problemi proposti.
Come nel caso bidimensionale, anche nello spazio i primi due algoritmi vengono implementati in ogni telecamera, per cui l'elaborazione di ogni frame avviene parallelamente in ogni telecamera. Per quanto riguarda il terzo algoritmo il comportamento è diverso, infatti esso prevede il passaggio di informazioni fra le telecamere mediante una gerarchia stabilita dal consueto schema ad albero.

Nell'ultima parte della sezione verrà analizzata nel dettaglio la rappresentazione dell'incertezza nella ricostruzione dei marker nello spazio individuata nel corso del progetto.

5.1 Calcolo dei centri

Il primo algoritmo è stato implementato nella funzione denominata "calc_centri_3D". Come prima precisazione è necessario notare che le posizioni dei centri individuati sono fornite rispetto al sistema di riferimento del piano immagine della telecamera.
Di seguito è riportata la chiamata alla funzione. Successivamente si procederà all'analisi dettagliata di ingressi ed uscite.

$$[centri_x, centri_y, num_{pixel}] = calc\_centri\_3D(mat_{piano}, pixel_{dimx}, pixel_{dimy}, marker_{tot})$$

La funzione per il calcolo dei centri delle proiezioni dei marker sul piano immagine della telecamere ha come ingressi

- La matrice mat_{piano} contiene la rappresentazione matriciale del piano immagine della telecamera. Anche nello spazio tridimensionale i pixel "accesi" sono rappresentati da un uno, mentre quelli "spenti" da uno zero.
- I due valori $pixel_{dimx}$ e $pixel_{dimy}$ contengono rispettivamente la dimensione della base e dell'altezza del pixel. Questi parametri sono fondamentali per mantenere la corretta proporzione fra la posizione del centro calcolato e la dimensione del sensore della telecamera.
- Il parametro $marker_{tot}$ contiene il numero parametro dei marker presenti nella stanza. Come per il caso bidimensione, anche in questa situazione il parametro ha il solo scopo di inizializzare le variabili interne della funzione.

Si può notare che, a differenza del caso bidimensionale, nello spazio tridimensionale non è richiesto il passaggio del numero di zeri ammissibile tra due sequenza di uno. Questo è dovuto al fatto che la struttura dell'algoritmo prevede che tutti gli uno fra loro collegati rappresentino la proiezione del medesimo marker sul piano immagine. Successivamente verrà spiegato il motivo di questo comportamento.

Le uscite della funzione sono

- I vettori $centri_x$ e $centri_y$ contengono rispettivamente la posizione dei centri delle proiezioni dei marker sul piano immagine sull'asse delle ascisse e su quello delle ordinate. Questi vettori verranno utlizzati nella sottosezione successiva per il calcolo delle rette.
- Il vettore num_{pixel} contiene il numero esatto di pixel "accesi" della proiezione del rispettivo marker sul piano immagine. Questo parametro è strettamente legato alla distanza del marker dalla telecamera. Per questo motivo il parametro verrà utilizzato in seguito per stimare l'incertezza nell'individuazione del marker.

Dopo aver presentato ingressi ed uscite della funzione è bene spiegare brevemente la struttura dell'algoritmo implementato. Per chiarire il comportamento dell'algoritmo, esso verrà applicato al piano immagine mostrato in Figura 23
Analogamente a quanto fatto nel caso bidimensionale, anche nel caso tridimensionale la funzione è stata strutturata in due parti distinte.
La prima parte è costituita da un algoritmo ricorsivo avente lo scopo di individuare i pixel appartenenti alla proiezione dello stesso marker. Il procedimento consiste nell'indagare tutti gli elementi della matrice rappresentazione del piano immagine. Appena viene individuato un uno, viene richiamato l'algoritmo ricorsivo che, scorrendo tutti gli uni collegati fra loro, restituisce in uscita una matrice della stessa dimensione di quella iniziale ma con gli uno collegati fra loro che rappresentano un solo marker e tutti gli altri elementi nulli. Applicando questo procedimento sugli elementi della matrice originale si ottiene un numero di matrici pari al numero di marker visibili dalla telecamera, della stessa dimensione della matrice originale. In ogni matrice sono presenti solamente i pixel che rappresentano la proiezione di un solo marker.

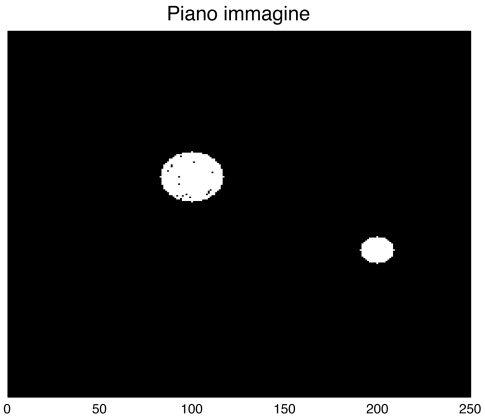

Figura 23: Piano immagine utilizzato per chiarire il comportamento dell'algoritmo.

A questo punto si può applicare la seconda parte dell'algoritmo su ogni matrice precedentemente individuata. In questa fase viene applicato l'algoritmo per il calcolo dei centri delle proiezioni dei marker nel piano per righe e per colonne della matrice. A questo punto è sufficiente fare la media per l'asse delle ascisse e per quello delle ordinate per individuare le coordinate x ed y del centro.
L'applicazione della funzione per il calcolo dei centri delle proiezioni dei marker sul precedente piano immagine produce il calcolo dei due centri, secondo le figure riportate in seguito. Nella Figura 24 è riportata la proiezione del primo marker, ottenuta con il primo passaggio dell'algoritmo, sulla quale la seconda parte calcola il centro.

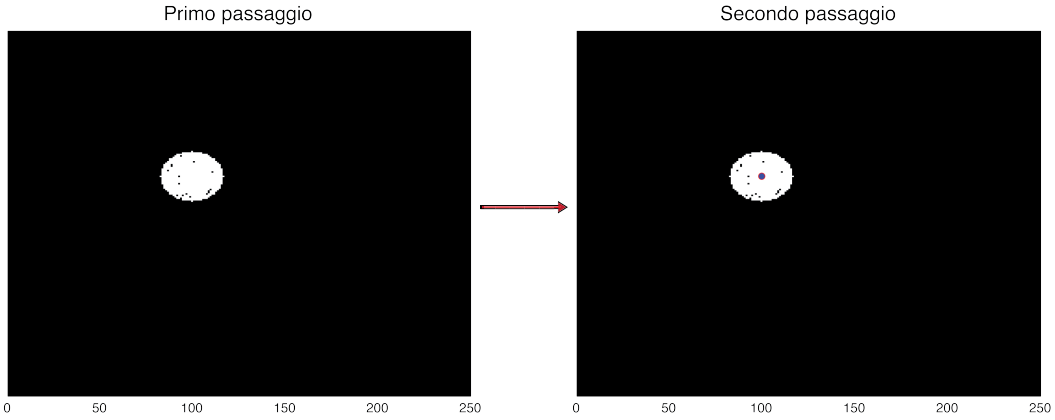

Figura 24: Algoritmo per il calcolo del centro della proiezione del primo marker.

In Figura 25 è riportata la proiezione del secondo marker, ottenuta mediante l'algoritmo ricorsivo, sulla quale viene calcolato il centro mediante il secondo passaggio.

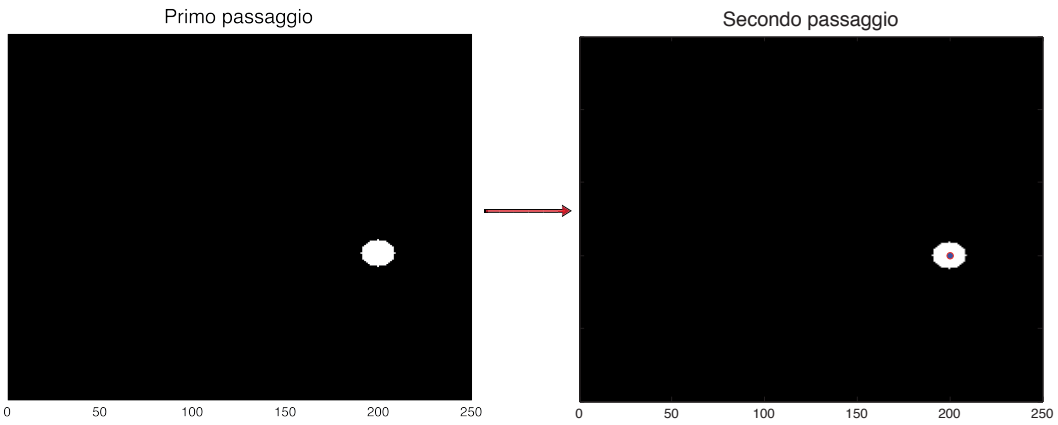

Figura 25: Algoritmo per il calcolo del centro della proiezione del secondo marker.

5.2 Calcolo delle rette

Si utilizza la funzione 'assoluto3D' per riportare le coordinate calcolate sul piano immagine di ogni singola telecamera nel sistema di riferimento assoluto. Come nel caso bidimensionale la funzione 'calcola_centri_3D' restituisce un insieme di celle pari al numero di telecamere contenenti le varie coordinate dei centri. Per ogni singola cella si utilizza la funzione:

$$[x_{assoluto}, y_{assoluto}, z_{assoluto}] = assoluto3D(x_t, y_t, z_t, R, coordinate, height, width, f)$$

Riceve in ingresso:

- x_t,y_t,z_t: coordinate della telecamera rispetto al sistema di riferimento assoluto;
- R: matrice di rotazione per ogni singola telecamera, utilizzando la notazione di Cardano con rotazioni successive secondi gli assi X $\rightarrow$ Y $\rightarrow$ Z:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -sen(\alpha) \\ 0 & sen(\alpha) & \cos(\alpha) \end{bmatrix}, R_y = \begin{bmatrix} \cos(\beta) & 0 & sen(\beta) \\ 0 & 1 & 0 \\ -sen(\beta) & 0 & \cos(\beta) \end{bmatrix},$$

$$R_z = \begin{bmatrix} \cos(\delta) & -sen(\delta) & 0 \\ sen(\delta) & \cos(\delta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_z \cdot R_y \cdot R_x \tag{14}$$

- coordinate: coordinate calcolate sul piano immagine date dalla proiezioni dei marker;
- height,width: altezza e larghezza del piano immagine in pixels;
- f: lunghezza focale della telecamera.

La funzione, utilizzando le coordinate omogenee, calcola la matrice di trasformazione:

$$T = \begin{bmatrix} & & & x_t \\ & R & & y_t \\ & & & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Quest'ultima viene moltiplicata per le coordinate dei centri, calcolate nel sistema di riferimento della telecamera:

$$\begin{bmatrix} x_{assoluto} \\ y_{assoluto} \\ z_{assoluto} \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} x_{centri} \\ y_{centri} \\ z_{centri} \\ 1 \end{bmatrix}$$

Le coordinata z_{centri} calcolata sul piano immagine 2D è sempre pari a zero. Quindi la funzione in uscita restituisce:

- $x_{assoluto}$,$y_{assoluto}$,$z_{assoluto}$: coordinate dei centri calcolati sul piano immagine, riportate nel sistema di riferimento assoluto.

Quindi in uscita la funzione restituisce un insieme di celle contenenti la posizione dei centri calcolati sul piano immagine, riportati nel sistema di riferimento assoluto. In questo modo è possibile calcolare l'equazione parametrica delle rette generate dalla varie proiezioni dei marker nelle telecamere.

5.3 Ricostruzione dei marker nello spazio

5.3.1 Algoritmo distribuito

La funzione 'Ricostruzione_punti_3D' ricostruisce la posizione dei marker nello spazio a partire dai centri calcolati sul piano immagine di ogni singola telecamera.

Si adotta il consueto schema ad albero per la gerarchia nello scambio di informazione tra

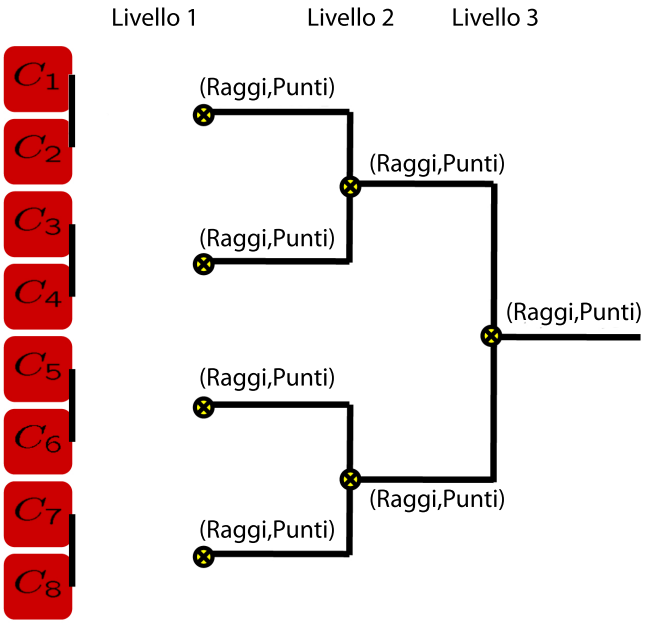

Figura 26: Strategia di ricostruzione a albero.

le telecamere, riportato in Figura 26. Come si può notare, vengono utilizzate 8 telecamere e calcolati tre livelli di associazione degli insiemi di raggi e punti. Nel caso si utilizzino più telecamere i livelli di associazione aumenteranno di conseguenza.

Nel primo livello dell'albero vengono ricostruiti i marker utilizzando le rette ricavate dai piani immagine delle telecamere, queste ultime risultano essere associate a coppie. La funzione restituisce un insieme contenente i valori dei marker ricostruiti ed i raggi di entrambe le telecamere che non vengono associati per la ricostruzione.

Nella prima implementazione dell'algoritmo per la ricostruzione dei marker nello spazio, i raggi utilizzati per la ricostruzione di un marker valido vengono eliminati. Al contrario, i raggi non utilizzati vengono salvati per un'eventuale associazione ad un livello successivo. La funzione:

$$[Tempo, Punti, Rette] = Ricostruzione\_punti\_3D(x_{as}, y_{as}, z_{as}, camera\_pos, m, dist\_min, \varepsilon)$$

riceve in ingresso:

- x_{as},y_{as},z_{as}: coordinate dei centri riportate nel sistema di riferimento assoluto;
- $camera\_pos$: posizione delle telecamere nel sistema di riferimento assoluto;
- m: numero delle telecamere;

- *dist_min*: distanza minima accettata tra due rette per la ricostruzione del punto;
- ε: distanza minima accettata per la mediazione dei punti.

La funzione restituisce:

- *Tempo*: vettore dei tempi necessari per la ricostruzione dei punti, dei singoli livelli e il tempo complessivo;
- *Punti*: vettore dei marker ricostruiti e la loro distanza minima;
- *Rette*: celle di rette non utilizzate per la ricostruzione dei punti.

La funzione calcola i parametri dell'equazione parametrica di una retta passante per due punti. Il primo punto è dato dalla posizione della telecamera, mentre il secondo è dato dalla posizione del centro della proiezione del marker sul piano immagine, le coordinate di entrambi i punti sono date rispetto al sistema di riferimento assoluto della stanza. Si supponga di essere nel primo livello dell'albero, e di avere due rette determinate dalle equazioni:

$$s: \begin{cases} x_1 = x_{0,1} + lt_1 \\ y_1 = y_{0,1} + mt_1 \\ z_1 = z_{0,1} + nt_1 \end{cases}, r: \begin{cases} x_2 = x_{0,2} + kt_2 \\ y_2 = y_{0,2} + vt_2 \\ z_2 = z_{0,2} + wt_2 \end{cases} \tag{15}$$

Per ricostruire la posizione dei marker nello spazio, si utilizzano le due rette individuate. Si vogliono determinare i valori dei parametri t_1 e t_2, che determinano un punto su ogni retta, tali per cui la retta passante per i due punti individuati sia ortogonale ad entrambe le rette di partenza. I due punti determinati dai parametri calcolati risultano essere i punti di minima distanza fra le due rette. La posizione del marker nello spazio è, quindi, data dal punto di minima distanza fra le due rette originali, che coincide con il punto di minima distanza fra i due punti precedentemente ricavati.
Per poter implementare un'algoritmo in funzione di t_1 e t_2, si ricava un sistema lineare:

$$M = \begin{bmatrix} -(l^2+m^2+n^2) & (l\cdot k + m\cdot v + n\cdot w) \\ -(l\cdot k + m\cdot v + n\cdot w) & -(k^2+v^2+w^2) \end{bmatrix}, \tag{16}$$

$$R = \begin{bmatrix} -l\cdot(x_{02}-x_{01}) - m\cdot(y_{02}-y_{01}) - n\cdot(z_{02}-z_{01}) \\ -k\cdot(x_{02}-x_{01}) - v\cdot(y_{02}-y_{01}) - w\cdot(z_{02}-z_{01}) \end{bmatrix}, \tag{17}$$

$$T = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = M^{-1}\cdot R \tag{18}$$

Ottenuti i valori di t_1 e t_2 si sostituisco nelle equazioni delle rette (15) ricavando i punti a minima distanza P=(x_p,y_p,z_p) $\in$ s e Q=(x_q,y_q,z_q) $\in$ r. Si calcola il punto di minima distanza tra le due rette:

$$Punto = \frac{P+Q}{2} \tag{19}$$

Quindi si calcola la minima distanza fra le due rette:

$$d(Q,P)^2 = (x_q - x_p)^2 + (y_q - y_p)^2 + (z_q - z_p)^2 \tag{20}$$

Se la minima distanza è minore del parametro $dist\_min$ allora il marker ricavato viene accettato e memorizzato. Le rette che lo hanno generato vengono eliminate in modo da non essere considerate in altre ricostruzioni. Se il marker viene scartato le rette non utilizzate per la ricostruzione vengono memorizzate e passate al secondo livello dell'albero. Quindi dal secondo livello dell'albero fino al suo termine, viene fatto un controllo sulle rette non utilizzate nei livelli antecedenti, con lo stesso algoritmo utilizzato.
Dei marker ricostruiti vengono memorizzate le loro coordinate e la loro distanza minima, in modo da valutarne l'incertezza.
Si utilizzano due approcci per mediare i marker ricostruiti, il primo facendone una mediazione su tutti i marker ricostruiti al termine dell'albero, l'altro mediandoli sui vari livelli dell'albero. Questo secondo metodo viene implementato con la funzione 'Ricostruzione_punti_mediati_3D'. Gli ingressi e le uscite di quest'ultima sono identiche alla precedente funzione, con l'unica differenza sulla mediazione dei punti nei vari livelli dell'albero.
Il risultato può variare dalla posizione dei marker nella stanza, ma non mostra grosse variazioni sulla ricostruzione.
Per la mediazione dei punti ricostruiti, nei due metodi, si utilizza la funzione:

$$[punti\_finali] = media\_punti\_gen(\varepsilon, Punti\_tot)$$

La funzione riceve in ingresso:

- ε: distanza minima accettata per la mediazione dei punti;
- $Punti\_tot$: punti ricostruiti, sui livelli o al termine dell'albero.
- $punti\_finali$: punti ricostruiti mediati.

La funzione per la ricostruzione dei punti, restituisce anche il parametro Tempo con il quale si possono valutare le diverse tempistiche di ricostruzione, considerate su ogni livello di albero e il tempo complessivo per la ricostruzione calcolato sul percorso dell'albero che richiede il maggior tempo d'esecuzione. Il tempo varia significativamente all'aumentare del numero di telecamere e marker nella stanza. Inoltre influisce molto la potenza del processore della CPU utilizzato e il linguaggio con il quale il codice viene implementato.
Nella Figura (27,sinistra) si possono notare la posizione dei marker e la disposizione delle telecamere nel sistema di riferimento assoluto, cioè la stanza. Nella Figura (27,destra) si possono notare i raggi di ogni singola telecamera, con la relativa incertezza, che passano in prossimità della posizione reale dei marker. In questo esempio le telecamere sono impostate con una distanza di visione pari alla dimensione della stanza.
Nella Figura (28,sinistra) viene raffigurata la ricostruzione dei punti con la funzione localizzata, mediando i punti ricostruiti nei vari livelli dell'albero. Nella Figura (28,destra) viene raffigurata la ricostruzione dei punti con la funzione localizzata, mediando i punti ricostruiti al termine dell'albero.

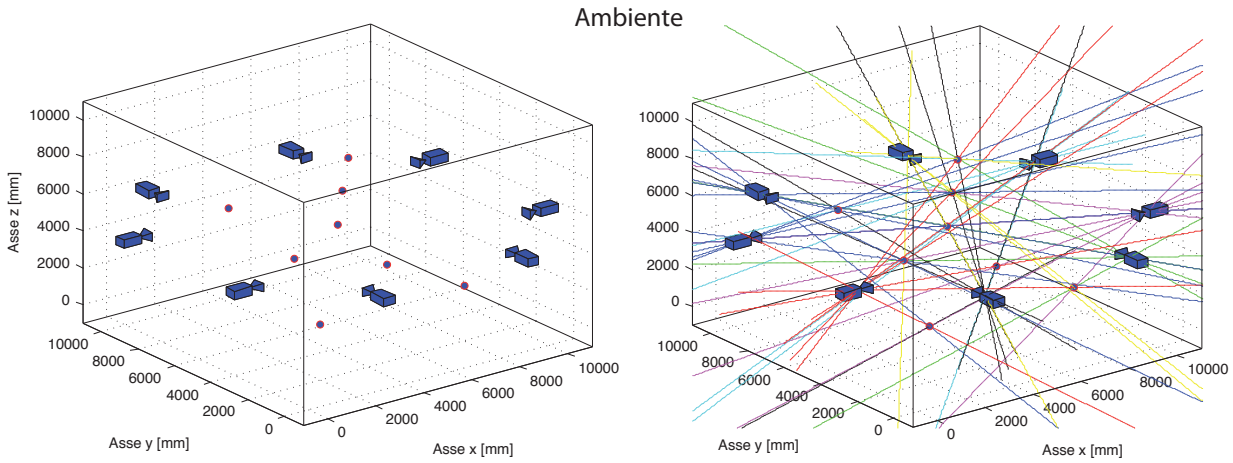

Figura 27: Stanza con 8 telecamere e 8 marker. Senza raggi e con raggi e relativa incertezza

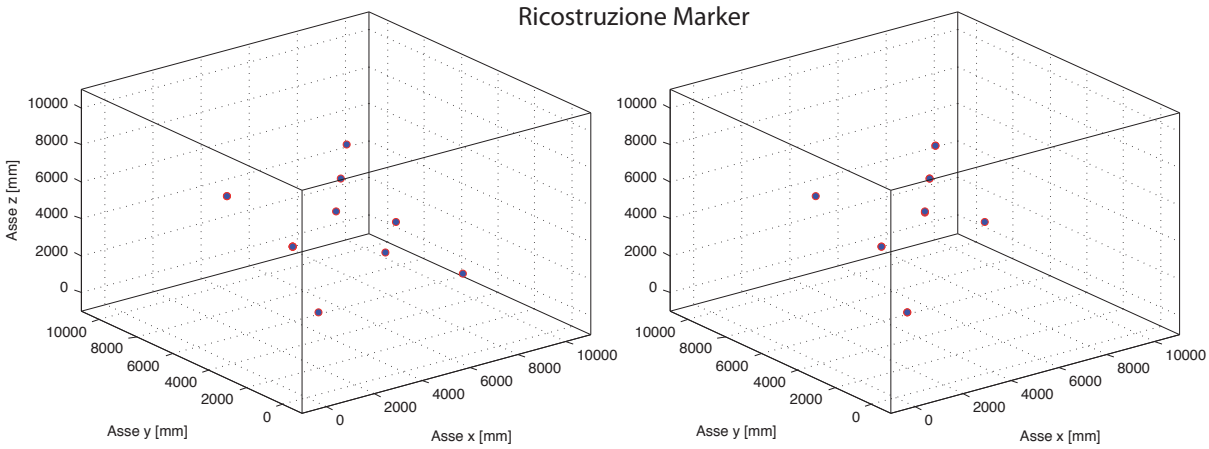

Figura 28: Marker ricostruiti con funzione localizzata. Sinistra: marker mediati nell'albero. Destra: marker mediati al termine dell'albero

5.3.2 Algoritmo distribuito pesato

Si implementa un'algoritmo distribuito ad albero come il precedente, con la differenza che la posizione dei marker riscostruiti viene pesata con il numero di pixels, presenti sul piano immagine, dei raggi utilizzati per ricostruirlo. La funzione:

$$[Tempo, Punti\_pesati, Rette] = Ricostruzione\_punti\_pesati\_3D(x_{as}y_{as}, z_{as}, num\_pixel,$$
$$camera\_pos, m, dist\_min, \varepsilon)$$

riceve in ingresso:

- x_{as},y_{as},z_{as}: coordinate dei centri riportate nel sistema di riferimento assoluto;
- num_pixel: numero di pixels per ogni centro, calcolato con la funzione 'calcola_centri_3D';
- $camera\_pos$: posizione delle telecamere nel sistema di riferimento assoluto;
- m: numero delle telecamere;
- $dist\_min$: distanza minima accettata tra due rette per la ricostruzione del punto;
- ε: distanza minima accettata per la mediazione dei punti.

La funzione restituisce:

- $Tempo$: vettore dei tempi necessari per la ricostruzione dei punti, dei singoli livelli e il tempo complessivo;
- $Punti$: vettore dei marker ricostruiti e la loro distanza minima;
- $Rette$: celle di rette non utilizzate per la ricostruzione dei punti.

La funzione implementa l'algoritmo duale alla precedente, pesando in modo differente il punto di minima distanza dalle due rette utilizzate. Infatti per la ricostruzione della posizione del marker non si utilizza l'equazione 19 ma bensi:

$$Punto = P \cdot \frac{num\_pixel_s}{num\_pixel_{tot}} + Q \cdot \frac{num\_pixel_r}{num\_pixel_{tot}}; \tag{21}$$

$$num\_pixel_{tot} = num\_pixel_s + num\_pixel_r \tag{22}$$

Si utilizza il valore num_pixel_s e num_pixel_r, cioè il numero di pixel sul piano immagine della retta r e s che hanno generato il punto. Quindi si valuta la distanza minima in modo differente:

$$d(Punto, P) = Punto - P; d(Punto, Q) = Punto - Q; \tag{23}$$

$$dist = \max(d(Punto, P), d(Punto, Q)); \tag{24}$$

$$d^2 = x_{dist}^2 + y_{dist}^2 + z_{dist}^2 \tag{25}$$

In questo caso viene presa in considerazione la distanza massima dal punto ricostruito alla retta, in modo da considerarne il caso peggiore.
In questo modo si può valutare la ricostruzione dei marker e la loro incertezza confrontandoli con i risultati ricavati dalla funzione precedente.

5.3.3 Algoritmo centralizzato

Per riprodurre un algoritmo comune nella Motion Captur, si implementa una funzione per la ricostruzione dei marker in modo centralizzato.

$$[Punti] = Ricostruzione\_centralizzato(x_{as}, y_{as}, z_{as}, camera\_pos, m, dist\_min, \varepsilon)$$

La funzione riceve in ingresso:

- x_{as},y_{as},z_{as}: coordinate dei centri riportate nel sistema di riferimento assoluto;
- $camera\_pos$: posizione delle telecamere nel sistema di riferimento assoluto;
- m: numero delle telecamere;
- $dist\_min$: distanza minima accettata tra due rette per la ricostruzione del punto;
- ε: distanza minima accettata per la mediazione dei punti.

La funzione restituisce:

- $Punti$: vettore dei marker ricostruiti in modo centralizzato e la loro distanza minima;

A differenza della funzione localizzata, questa funzione implementa l'algoritmo precedentemente analizzato senza utilizzare uno schema ad albero. Cioè i raggi prodotti dalla telecamera vengono analizzati con tutte le telecamere restanti. Questo per tutte le telecamere presenti nella stanza senza che alcun raggio venga eliminato. I marker ricostruiti vengono mediati con la funzione 'media_punti_gen' precedentemente analizzata.

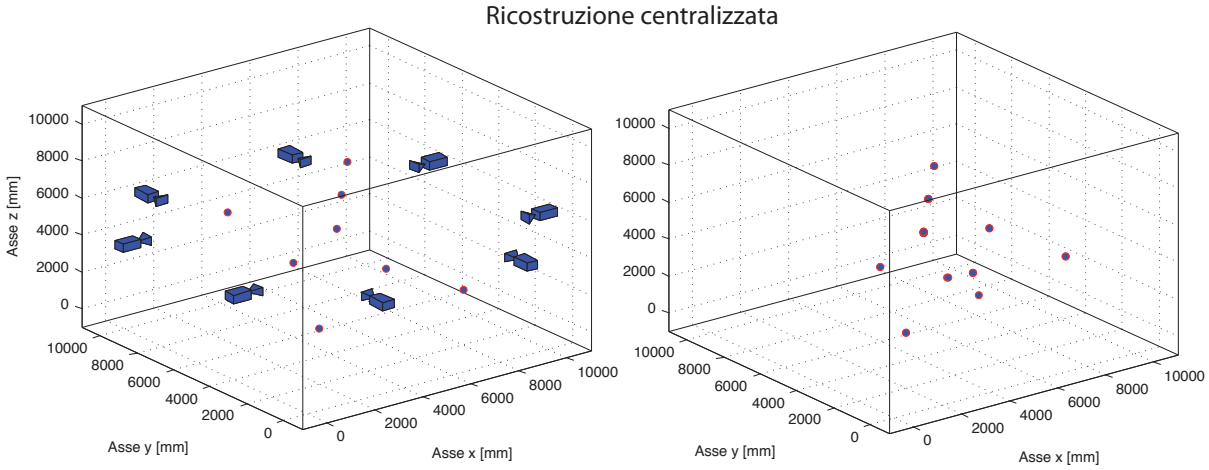

Figura 29: Marker ricostruiti con funzione centralizzata. Sinistra: posizione reale marker. Destra: ricostruzione marker.

Nell'esempio di Figura 29, impostando la lunghezza di visione delle telecamere con la dimensione della stanza, si nota che i marker vengono ricostruiti in un numero maggiore di quello dei marker reali. Questi approfondimenti verranno discussi successivamente nel capitolo 'confronti e simulazioni'.

5.4 Rappresentazione dell'incertezza

In questa parte del progetto si vuole fornire una rappresentazione dell'incertezza nella ricostruzione del marker nello spazio, analogamente a quanto fatto nel caso bidimensionale.
Si è deciso di caratterizzare l'incertezza come un volume nello spazio centrato nel marker ricostruito. Per semplicità si è scelto di utlizzare una sfera, in quanto, per essere caratterizzata, necessita solamente del centro e del raggio, che, nello spazio, si traducono in quattro parametri. Il centro della sfera è dato dalla posizione del marker ricostruito. Il raggio è stato ricavato a

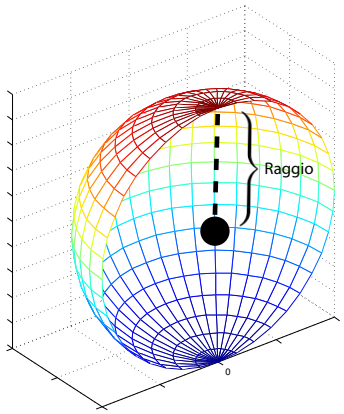

Figura 30: Rappresentazione dell'incertezza nello spazio.

partire dalla distanza tra il marker ricostruito e le due rette dalle quali è stato ricavato. A seconda del tipo di implementazione utilizzata per la ricostruzione del marker, il calcolo del raggio della sfera varia lievemente.

- Per prima cosa si prende in considerazione l'implementazione più semplice dell'algoritmo per la ricostruzione dei marker nello spazio, che utilizza solamente due rette per determinare la posizione del marker. In questo caso, il raggio della sfera che rappresenta l'incertezza è dato dalla metà della distanza tra le due rette utilizzate nella determinazione del marker, come in Figura 31.

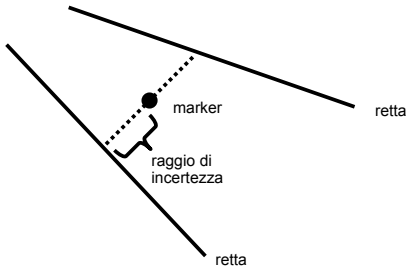

Figura 31: Determinazione del raggio della sfera di incertezza per il primo algoritmo.

- Ora si consideri l'implementazione che determina la posizione del marker nello spazio pesando le due rette in funzione del numero di pixel del centro della proiezione del marker dal quale sono individuate. In questa situazione il raggio della sfera caratterizzante

l'incertezza si ricava dal calcolo della distanza tra il marker ricostruito e le due rette che lo individuano. Dal momento che l'algoritmo è pesato le distanza tra il marker e le due rette saranno diverse, per considerare il caso peggiore, si prenda come raggio la maggiore tra le due, come mostrato in Figura 32.

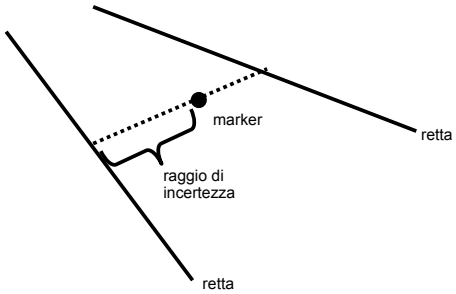

Figura 32: Determinazione del raggio della sfera di incertezza per il secondo algoritmo.

- In questo caso si considera l'implementazione dell'algoritmo che media, in ogni livello dell'albero, i punti che si riferiscono allo stesso marker. Con questo approccio il calcolo del raggio della sfera si basa sulla distanza tra i due punti da mediare e sui raggi, precedentemente calcolati mediante uno dei due metodi precedenti, che li caratterizzano. Anche in queso caso si considera il caso peggiore, per cui il raggio assocciato al nuovo marker sarà dato dalla metà della distanza tra i due vecchi marker sommata ad entrambi i due raggi ad essi associati, come in Figura 33.

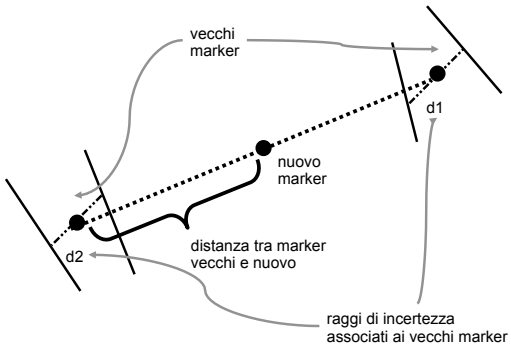

Figura 33: Determinazione del raggio della sfera di incertezza per il terzo algoritmo.

Ora non rimane che verificare se la rappresentazione dell'incertezza individuata è tale da assicurare che il marker reale si trovi sempre all'interno del volume di incertezza. Questa verifica verrà discussa nella sezione successiva.

6 Confronti e simulazioni nello spazio

6.1 Implementazione delle simulazioni nello spazio

Come nel caso bidimensionale, anche nel caso tridimensionale si utilizza un main principale dove poter utilizzare e simulare le varie funzioni. La dimensione della stanza è stata scelta come nel caso bidimensionale 10x10 metri, eventualmente scalabile. A differenza la posizione delle telecamere viene specificata con il parametro :

$$camera\_pos = \begin{bmatrix} x_t & y_t & z_t & \alpha & \beta & \gamma \end{bmatrix}$$

- x_t, y_t, z_t coordinate della telecamera rispetto al sistema di riferimento assoluto;
- α = rotazione rispetto asse x;
- β = rotazione rispetto asse y;
- γ = rotazione rispetto asse z.

Queste rotazioni si avvalgono delle notazione adottata in Figura 2.
A differenza del caso bidimensionale la coordinata z_t non sarà pari a zero ma avrà un valore reale. Le varie dimensioni scelte per la lunghezza focale, dimensione del pixel rimangono invariate dal caso bidimensionale, con la possibilità di modifica. Per quanto riguarda la dimensione del sensore si è scelto una altezza e larghezza pari rispettivamente a 10x10 millimetri e un numero di pixels per lato pari a 2000x2000 pixels, cioè una telecamera con 4 MegaPixels. Si sono scelte queste dimensioni per poter scostarsi il meno possibile dalla realtà, comunque eventualmente scalabili dal Main a piacimento. Possiamo notare alcune dimensioni standard di sensori matriciali in commercio in Figura 34 e la loro forma nella Figura 46.
Anche in questo caso per poter sapere le coordinate della proiezione dei marker sul piano immagine, si utilizza la funzione fornita dal gruppo di lavoro 2:

$$[coordinate, hidden, distance] = camera\_simulation(R, camera\_pos, punti, K, f, heigth, width)$$

La funzione riceve in ingresso:

- la matrice di rotazione R, come nell'equazione 14;
- camera_pos: posizione di ogni singola telecamera nel sistema di riferimento assoluto;
- punti: la posizione dei marker nel sistema di riferimento assoluto;
- K: matrice dei parametri intrinseci della telecamera:

$$K = \begin{bmatrix} k_x & 0 & v \\ 0 & k_y & u \\ 0 & 0 & 1 \end{bmatrix} \tag{26}$$

  Con k_x larghezza dei pixel, k_y altezza dei pixel, u, v rappresentano la traslazione dell'origine del piano immagine rispetto al centro ottico, espresse nella stessa unità di misura di k_x e k_y;

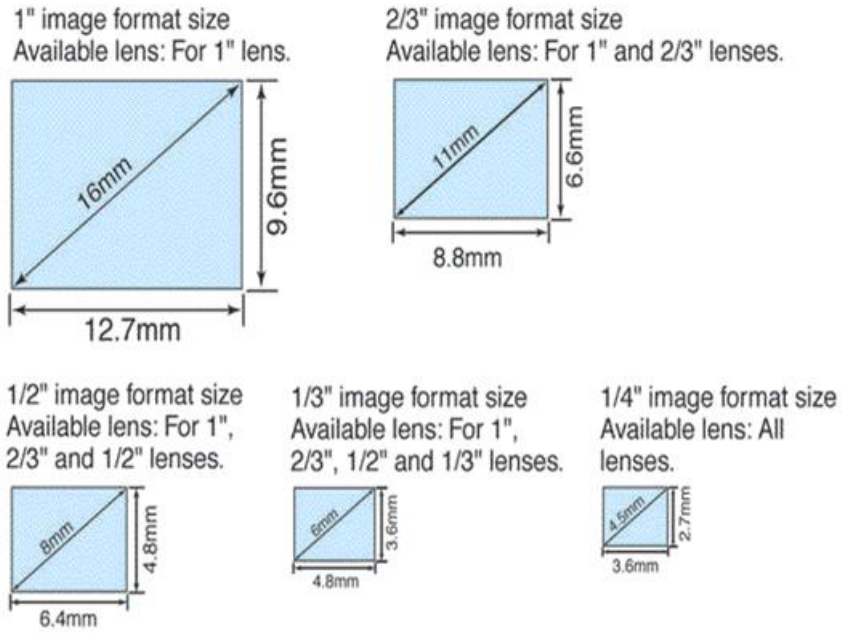

Figura 34: Area del sensore.

- f: lunghezza focale;
- heigth, width: rappresentano la dimensione in pixel del piano immagine.

In uscita la funzione restituisce le coordinate 2D sul piano immagine della proiezione dei marker, rispetto al sistema di riferimento traslato dalla matrice K. Anche i marker non visibile dalla telecamera hanno la loro proiezione sul piano immagine, per questo viene restituito il secondo parametro booleano hidden che per ogni coordinata riportata sul piano immagine restituisce 0 se il marker è visibile oppure 1 se non.
Determinate le coordinate possiamo utilizzare la funzione genera_piano_3D.
La funzione viene implementata pressapoco come nel caso bidimensionale:

$$[piano] = genera\_piano\_3D(P, heigth, width, d\_min\_vis, d\_max\_vis, sigma)$$

riceve in ingresso:

- P: coordinate calcolate sul piano immagine e la distanza dalla telecamera di ogni marker;

$$P = \begin{bmatrix} x_{coord\_1} & y_{coord\_1} & dist_1 \\ \vdots & \vdots & \vdots \\ x_{coord\_n} & y_{coord\_n} & dist_n \end{bmatrix}$$

- heigth: altezza del sensore in pixels;
- width: larghezza del sensore in pixels;
- dist_min_vis: distanza minima di visione della telecamera;

- dist_max_vis: distanza massima di visione della telecamera;
- σ: varianza dell'errore da utilizzare per 'sporcare' il piano immagine generato.

In uscita restituisce:

- piano_gen: piano immagine composto da vettore heigth x width contenente valori 0 e 1.

Le misure della distanza massima e minima di visione della telecamera vengono espresse in millimetri.

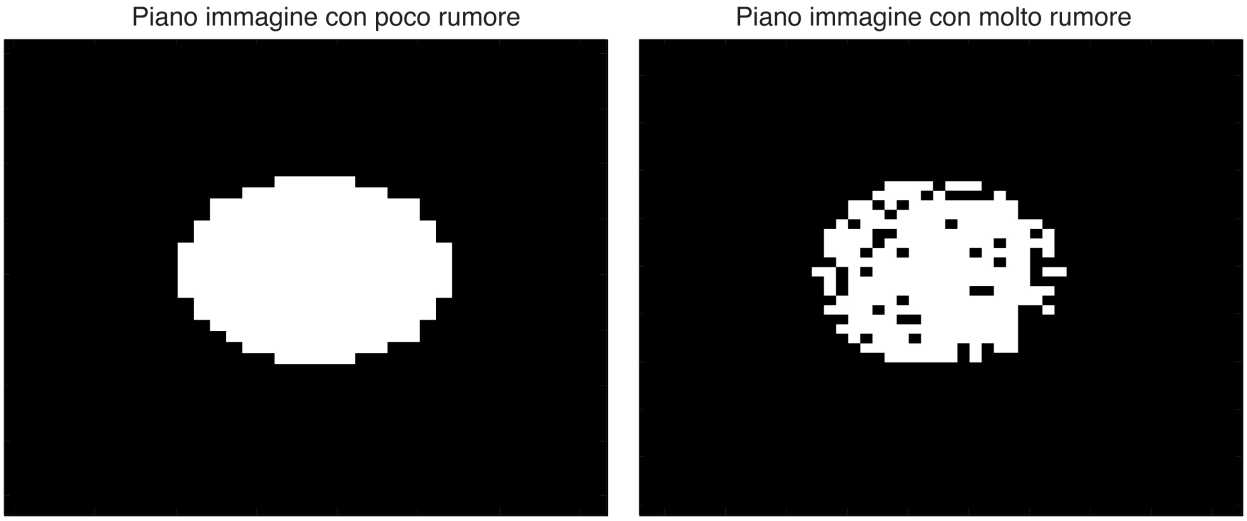

Figura 35: Sezione di piano immagine generato.

Nella Figura 35 viene rappresentata la sezione di un piano immagine generato con un σ molto elevato, quindi con un rumore generato sul piano immagine praticamente nullo e una sezione di piano con un σ molto basso, quindi con molto rumore. Il piano immagine,con il relativo rumore, di ogni singola telecamera, viene utilizzato dalla funzione 'calc_centri_3D' la quale restituisce i centri dati dalla proiezione dei marker nel sistema di riferimento della telecamera. Le coordinate riferite al sistema di riferimento della telecamera vengono riportate nel sistema di riferimento assoluto tramite la funzione 'assoluto3D'. A questo punto è possibile plottare i raggi prodotti dalle telecamere, loro relativa posizione nella stanza e la posizione dei marker.

Per poter eseguire i confronti e le simulazioni delle varie funzioni implementate, si sceglie la posizione reale dei marker nella configurazione rappresentata in Figura 36. Si noti la stanza con 8 marker nella loro posizione reale e la posizione delle telecamere disposte attorno ad essa. Nelle varie simulazioni si considera una distanza minima di visione della telecamera pari a 2 metri e una distanza massima pari a 7 metri. Si utilizza un valore medio di rumore generato sui vari piani immagini, almeno che non venga specificato il valore del parametro σ.
Nella Tabella 6 viene riportata la posizione reale dei vari marker nella stanza.

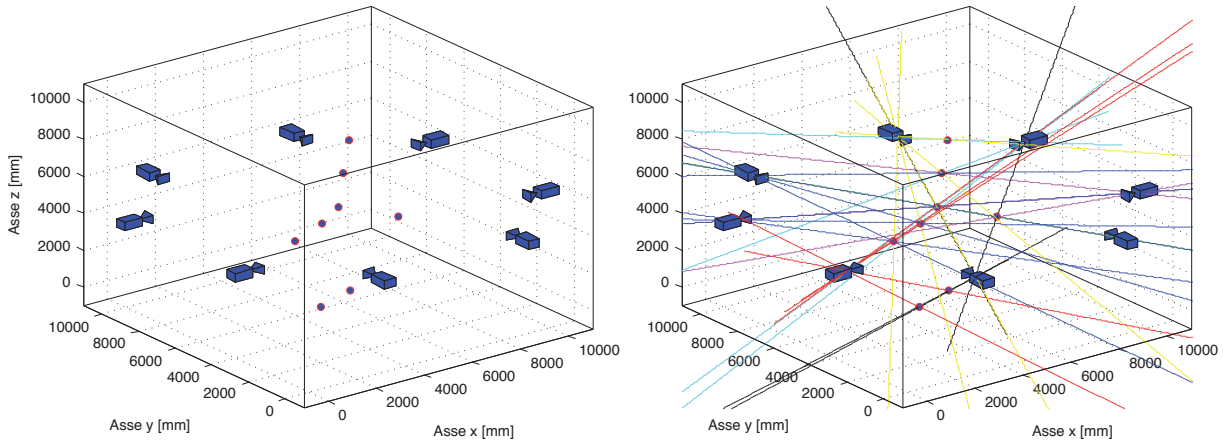

Figura 36: Sinistra: stanza con posizione telecamere e posizione reale marker. Destra: raggi prodotti dalla posizione reale dei marker.

	Posizione reale marker							
x	2000	5000	6000	7000	4000	6000	4800	3900
y	2000	5000	6000	7000	6000	3000	5600	2900
z	2000	5000	6000	7000	3000	5000	3900	1800

Tabella 6: Posizione reale marker, in millimetri

6.1.1 Implementazione distribuita che media i punti

Come prima implementazione si utilizza la funzione 'Ricostruzione_punti_3D' con la configurazione in Figura 36. La funzione, descritta nel dettaglio in precedenza, ricostruisce la posizione dei marker mediante un approccio distribuito, mediando i marker ricostruiti al termine dell'albero.

Nella Figura 37 viene visualizzata la posizione reale e la posizione ricostruita dei marker. Per apprezzare la posizione dei marker ricostruiti, viene rappresentata in Figura 38 la visione della stanza dall'alto, cioè nel piano X-Y. Si nota un marker ricostruito in basso a destra della Figura 38 che non compare nella configurazione reale dei marker.
Questo è dovuto all'utilizzo di due sole rette per la ricostruzione del marker e può variare dalla

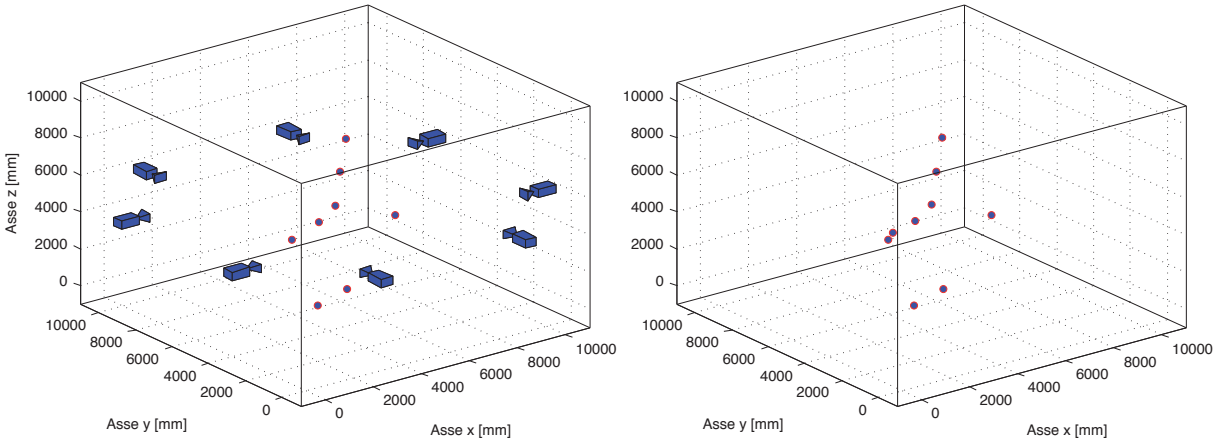

Figura 37: Sinistra: stanza con posizione telecamere e posizione reale marker. Destra: marker ricostruiti.

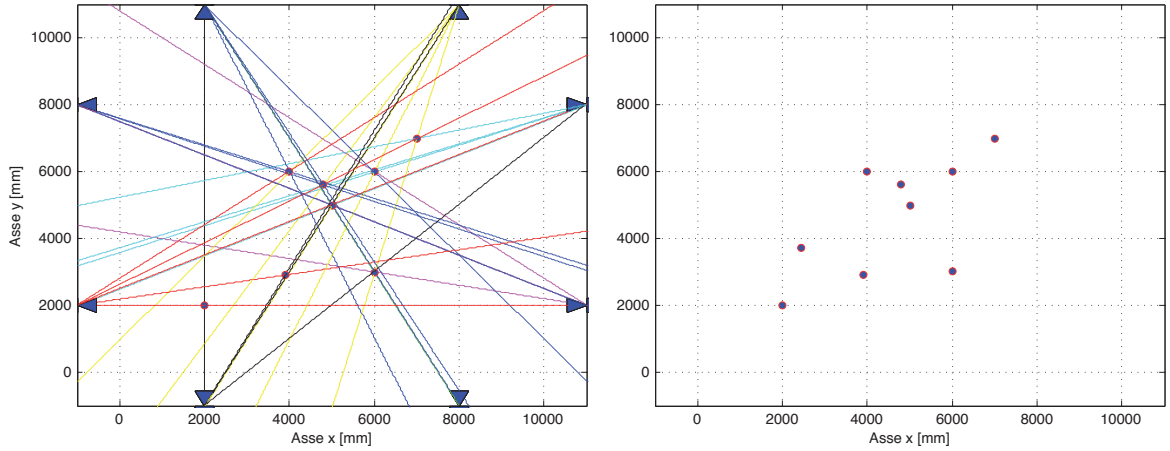

Figura 38: Sinistra: visione stanza nel piano X-Y con posizione telecamere e posizione reale marker. Destra: piano X-Y con posizione marker ricostruiti.

posizione e il numero dei marker nella stanza.
Con l'aumentare del numero di raggi prodotti, quindi aumentando il numero di marker e il numero delle telecamere nella stanza, aumenta la probabilità di ricostruire dei marker che non sono presenti nella realtà.

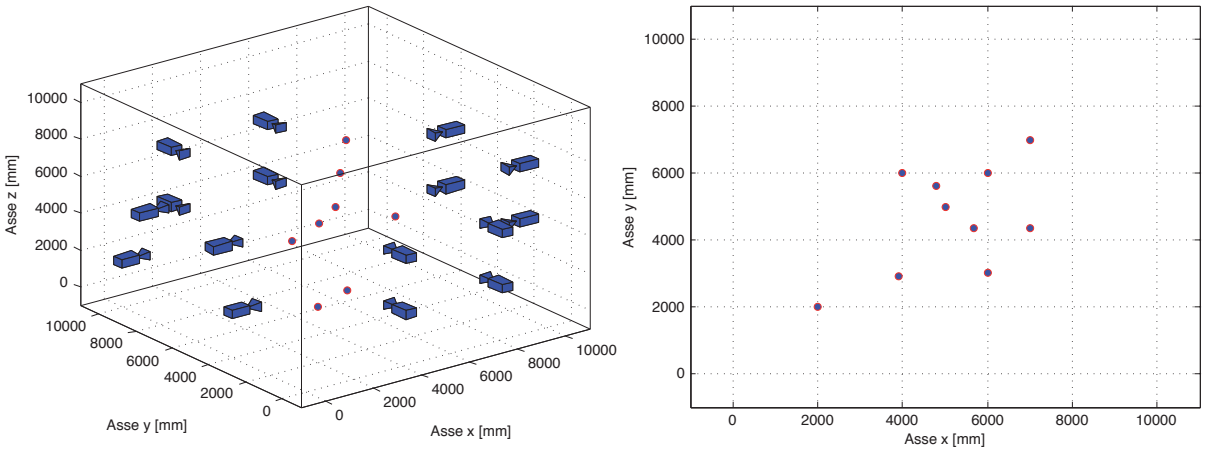

Figura 39: Sinistra:stanza con 16 telecamere e posizione reale marker. Destra: piano X-Y con posizione marker ricostruiti.

Infatti, come possiamo notare dalla Figura 39, con un numero maggiore di telecamere e la stessa configurazione di marker, il numero di ricostruiti è maggiore.
Nella Tabella 7 vengono riportati i valori della posizione reale e la posizione ricostruita dei marker, l'errore commesso nella ricostruzione e il relativo residuo. Quest'ultimo equivale alla distanza massima dalle rette che lo hanno ricostruito.

x	y	z	x_{ric}	y_{ric}	z_{ric}	errore	residuo
2000	2000	2000	1998,9	1999,4	2009,7	9,8179	0,4
5000	5000	5000	5002,5	4999,5	4999,8	2,5634	11
6000	6000	6000	6004,7	5998,6	5999,5	4,9275	2
7000	7000	7000	7003,6	6997,9	6999,8	4,1814	3,2
4000	6000	3000	4002,4	5999,9	2999,7	2,4051	3,1
6000	3000	5000	5997,8	3004,7	4999,6	5,1862	0,9
4800	5600	3900	4802,5	5601,4	3903,5	4,5335	2,1
3900	2900	1800	3899,5	2901,9	1800,3	1,9934	0,5

Tabella 7: Valori ottenuti utilizzando l'algoritmo localizzato con un rumore medio generato sul piano immagine. Valori in millimetri.

Nella Tabella 8 vengono riportati i tempi impiegati dall'algoritmo per ricostruire i marker nei vari livelli dell'albero. Il tempo complessivo d'esecuzione dell'algoritmo, si calcola con la somma dei tempi maggiori nei vari livelli. Il tempo complessivo risulta pari a 0,0548 secondi.
Questo algoritmo, come vedremo in seguito, richiede un tempo leggermente maggiore in confronto a quello richiesto dagli altri algoritmi distribuiti. Questo è dovuto al tempo richiesto per

Primo livello albero	Secondo livello albero	Terzo livello albero
0,021172	———	———
0,00334	0,024777	0,008887
0,000612	0,002178	———
0,00081	——-	———

Tabella 8: Tempo in secondi impiegato nei vari livelli dell'albero con 8 telecamere.

la mediazione dei marker ricostruiti.
Nella Tabella 9 vengono riportate le varie distanze dalla retta e il marker ricostruito, cioè i residui considerati, al variare del rumore generato sul piano immagine.
I valori riportati nelle colonne della tabella, corrispondono ai residui dei marker ricostruiti nell'ordine riportato nelle colonne di Tabella 6.
Si nota che all'aumentare del rumore, la distanza dalle rette utilizzate e il marker ricostruito aumenta di conseguenza, aumentando quindi l'incertezza sulla ricostruzione.

Rumore	**Residuo del marker ricostruito**							
basso	1,8	12,5	3,4	2,7	4	0,2	3,7	2
medio	1,9	12,9	4,1	2,9	4,2	0,5	4,4	2,1
alto	17,5	13,5	4,2	21,2	4,2	2,5	4,4	3,2

Tabella 9: Residui in millimetri dei marker ricostruiti al variare del parametro σ

6.1.2 Implementazione distribuita con eliminazione dei raggi

Come seconda implementazione si utilizza l'algoritmo distribuito senza mediazione dei marker ricostruiti. Quindi le rette, utilizzate per la ricostruzione dei marker nei vari livelli dell'albero, vengono eliminate senza tenerne conto in livelli successivi.
Nella Tabella 10 vengono riportati il valore della posizione dei marker reale e ricostruiti, l'errore di ricostruzione e i relativi residui.

x	y	z	x_{ric}	y_{ric}	z_{ric}	errore	residuo
2000	2000	2000	1998,9	1999,4	2009,7	9,8179	0,4
5000	5000	5000	5000,1	5000,1	4999,3	6	1,3
6000	6000	6000	6004,	5996,8	5999,2	5.4154	0,2
7000	7000	7000	7003,6	6997,9	6999,8	4,1814	3,2
4000	6000	3000	4000,1	6000,1	2999,8	4.7224	0,4
6000	3000	5000	5997,8	3004,7	4999,6	5,1862	0,9
4800	5600	3900	4802	5603	3903,4	4.7079	2
3900	2900	1800	3899,5	2901,9	1800,3	1,9934	0,5

Tabella 10: Valori ottenuti utilizzando l'algoritmo localizzato con un rumore medio generato sul piano immagine. Valori in millimetri.

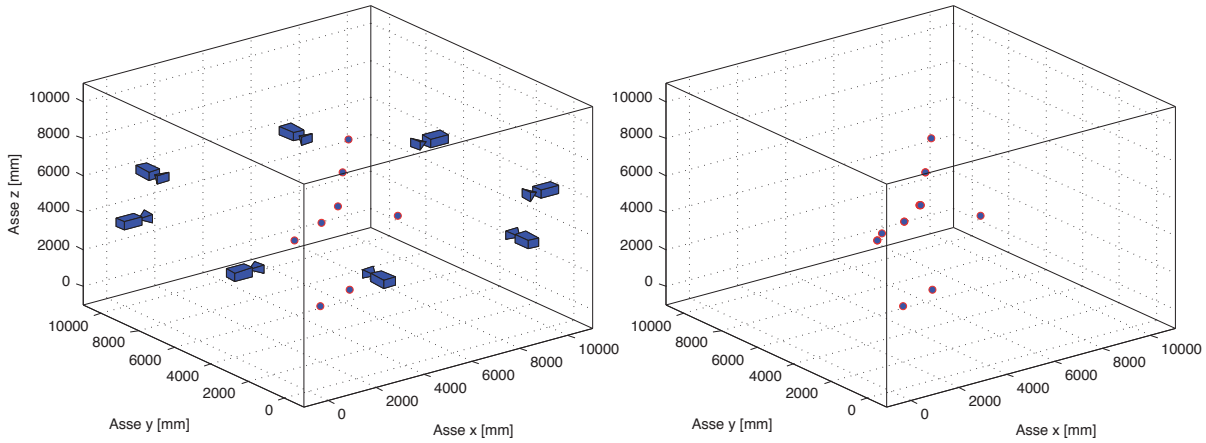

Figura 40: Sinistra: stanza con 8 telecamere e posizione reale marker. Destra: posizione marker ricostruiti.

Primo livello albero	Secondo livello albero	Terzo livello albero
0,002247	———	———
0,000385	0,0026667	0,003016
0,000589	0,000316	———
0,000892	———-	———

Tabella 11: Tempo in secondi impiegati nei vari livelli dell'albero con 8 telecamere.

Nella Tabella 11 vengono riportati i tempi impiegati dall'algoritmo per ricostruire i marker nei vari livelli dell'albero. Il tempo complessivo risulta pari a 0,007929 secondi. Il tempo d'esecuzione richiesto è nettamente inferiore rispetto le altre due implementazioni distribuite, dato dal fatto che non presenta mediazione dei marker, eliminando le rette utilizzate per ricostruirli.
Nella Tabella 12 vengono riportate le varie distanze dalla retta e il marker ricostruito, cioè i residui considerati, al variare del rumore generato sul piano immagine.
I valori riportati nelle colonne della tabella, corrispondono ai residui dei marker ricostruiti nell'ordine riportato nelle colonne di Tabella 6.
Come nella precedente, si nota che all'aumentare del rumore, la distanza dalle rette utilizzate e il marker ricostruito aumenta di conseguenza, aumentando quindi l'incertezza sulla ricostruzione.

	Distanza marker ricostruito							
basso	0,1	1	0,1	1,4	0,4	0,4	1,7	0,2
medio	0,4	1,3	0,2	3,2	0,4	0,9	2	0,5
alto	1,5	3,5	0,5	5	1,1	1,2	2,5	0,8

Tabella 12: Distanza in millimetri dei marker ricostruito al variare del parametro σ

6.1.3 Implementazione distribuita e pesata

Si eseguono le simulazioni utilizzando la funzione 'Ricostruzione_punti_pesati_3D'.
La funzione ricostruisce i marker utilizzando le rette pesate con il numero di pixel sul piano immagine che le hanno generate. In questo caso si considera la distanza massima tra il marker e le rette che lo hanno ricostruito, in modo da considerne il caso peggiore per quanto riguarda l'incertezza.

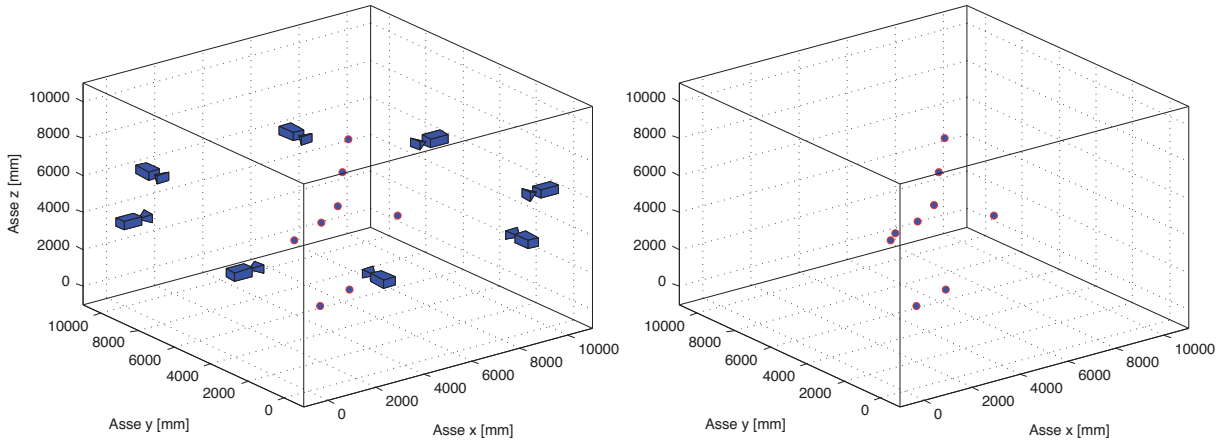

Figura 41: Sinistra: stanza con 8 telecamere e posizione reale marker. Destra: posizione marker ricostruiti.

Nelle Figure 41 e 42 si nota che la posizione dei marker ricostruiti è simile alla posizione della posizione reale, a differenza di un marker ricostruito che nella realtà non era presente.

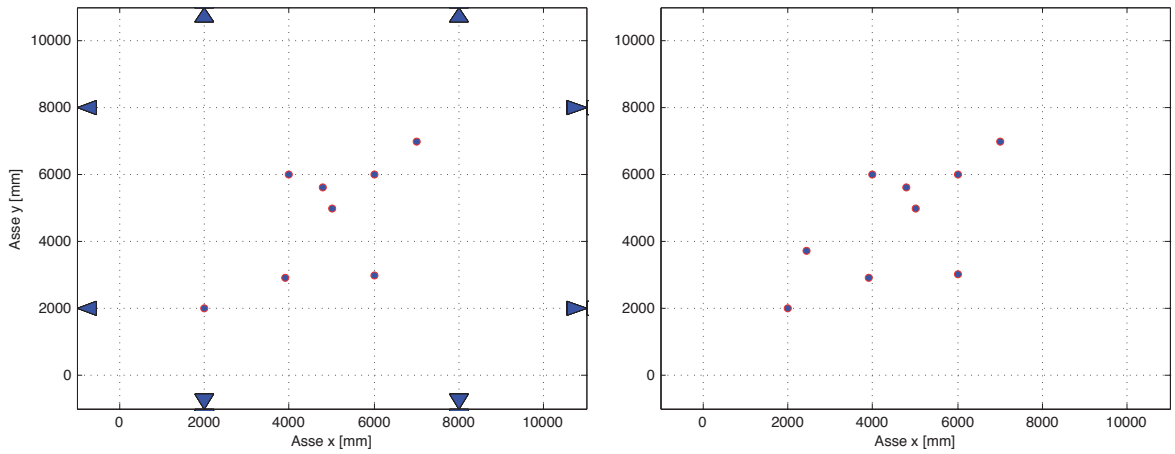

Figura 42: Sinistra: vista del piano X-Y della stanza con 8 telecamere e posizione reale marker. Destra: vista piano X-Y con posizione marker ricostruiti.

Nella Tabella 13 vengono riportati il valore della posizione dei marker reale e ricostruiti, l'errore di ricostruzione e i relativi residui.

x	y	z	x_{ric}	y_{ric}	z_{ric}	errore	residuo
2000	2000	2000	1998,9	1999,4	2009,7	9,8179	0,2
5000	5000	5000	5002,5	4999,5	4999,7	2,5776	11,1
6000	6000	6000	6004,7	5998,6	5999,5	4,9275	2
7000	7000	7000	7003,6	6997,9	6999,8	4,1868	1,7
4000	6000	3000	4002,4	5999,9	2999,6	2,3962	3
6000	3000	5000	5997,8	3004,7	4999,3	5,2176	0,7
4800	5600	3900	4802,5	5601,4	3903,5	4,5238	1,9
3900	2900	1800	3899,6	2902	1800,3	2,0291	0,7

Tabella 13: Valori ottenuti utilizzando l'algoritmo pesato con un rumore medio generato sul piano immagine.

Primo livello albero	Secondo livello albero	Terzo livello albero
0,020784	———	———
0,015404	0,028904	0,004091
0,000877	0,003912	———
0,001261	———-	———

Tabella 14: Tempo in secondi impiegato nei vari livelli dell'albero con 8 telecamere.

Nella Tabella 14 vengono riportati i tempi impiegati dall'algoritmo per ricostruire i marker nei

vari livelli dell'albero. Il tempo complessivo, dato dalla somma dei tempi maggiori nei vari livelli, risulta pari a 0,053779 secondi.
Quest'ultimo risulta leggermente inferiore al primo algoritmo distribuito ma nettamente superiore al secondo che elimina i raggi. Questo dovuto alle operazioni aggiunte per poter pesare i vari raggi con il numero di pixels che li hanno generati.
Nella Tabella 15 vengono riportate le varie distanze dalla retta e il marker ricostruito, cioè i residui considerati, al variare del rumore generato sul piano immagine.
I valori riportati nelle colonne della tabella, corrispondono ai residui dei marker ricostruiti nell'ordine riportato nelle colonne di Tabella 6.
Come ci si aspetta, all'aumentare del rumore la distanza dalle rette utilizzate e il marker ricostruito aumenta di conseguenza, aumentando quindi l'incertezza sulla ricostruzione.

	Distanza marker ricostruito							
basso	0,9	12,3	3,4	1,4	4	0,4	3,7	1,2
medio	17,7	14,1	3,8	1,5	3,9	0,9	4,4	1,4
alto	17,7	13,5	4	20	4	2	4,2	2,1

Tabella 15: Distanza in millimetri dei marker ricostruito al variare del parametro σ

6.1.4 Implementazione centralizzata

In questa sezione si implementa l'algoritmo di ricostruzione centralizzato utilizzando la funzione 'Ricostruzione_centralizzato', descritta in precedenza.
La funzione utilizza tutti i raggi prodotti dalle telecamere senza utilizzare un'approccio ad albero. In questo modo tutti i raggi, di ogni singola telecamera, vengono confrontati tra di loro senza che vengano eliminati una volta ricostruito il marker. Nella Tabella 16 vengono riportati

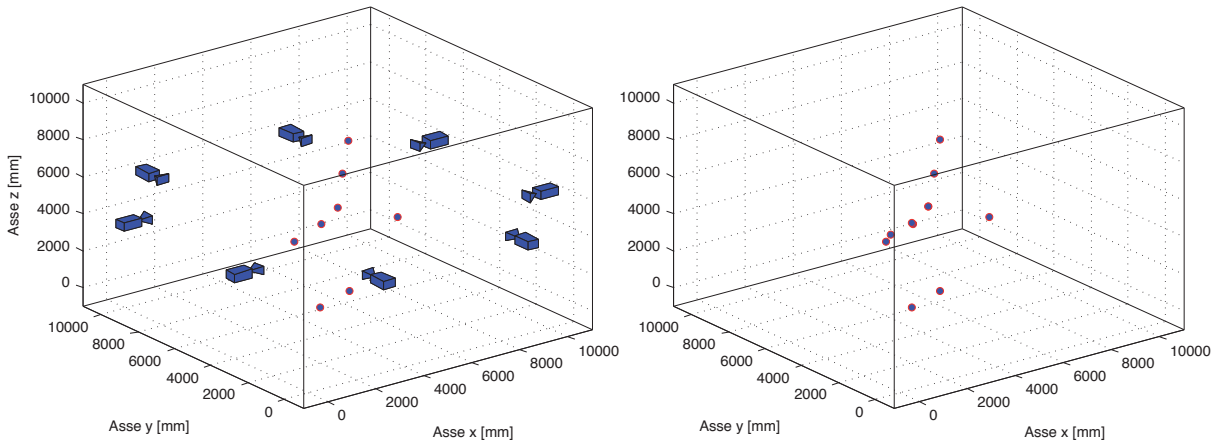

Figura 43: Sinistra: stanza con 8 telecamere e posizione reale marker. Destra: posizione marker ricostruiti con funzione centralizzata.

il valore della posizione dei marker reale e ricostruiti, l'errore di ricostruzione e i relativi residui.

x	y	z	x_{ric}	y_{ric}	z_{ric}	errore	residuo
2000	2000	2000	1998,9	1999,4	2009,7	9,8179	0,4
5000	5000	5000	5002,2	4999,5	4999,8	8.9502	19,4
6000	6000	6000	6004,5	5998	5999,6	4,8942	3,7
7000	7000	7000	7003,6	6997,9	6999,8	4,1814	3,2
4000	6000	3000	4003,4	5999,2	2999,2	3,5606	7,3
6000	3000	5000	5998,8	3002,5	4999,4	2,7944	3,8
4800	5600	3900	4801,5	5600,1	3903,4	3.6937	7,6
3900	2900	1800	3899,5	2901,9	1800,3	1,9934	0,5

Tabella 16: Valori ottenuti utilizzando l'algoritmo centralizzato con un rumore medio generato sul piano immagine. Valori in millimetri.

Il tempo necessario per l'esecuzione dell'algoritmo centralizzato è pari a 0,0809 secondi.
Quest'ultimo, come ci si aspettava, risulta essere il più elevato di tutti gli algoritmi implementati, sebbene la differenza non sia cosi rilevante come si stimava in una prima analisi a livello teorico.
Nella Tabella 17 vengono riportate le varie distanze dalla retta e il marker ricostruito, cioè i residui considerati, al variare del rumore generato sul piano immagine.
I valori riportati nelle colonne della tabella, corrispondono ai residui dei marker ricostruiti nell'ordine riportato nelle colonne di Tabella 6.
Come ci si aspetta, all'aumentare del rumore la distanza dalle rette utilizzate e il marker ricostruito aumenta di conseguenza, aumentando quindi l'incertezza sulla ricostruzione.

Rumore	**Distanza marker ricostruito**							
basso	0,6	10,7	6,2	2,8	8,2	4,4	4,4	2
medio	1,5	11,7	7,8	3,3	8,8	4,9	4,4	2,1
alto	17,8	27	8	21	9	6,3	12,2	3,2

Tabella 17: Distanza in millimetri del marker ricostruito al variare del parametro σ

6.1.5 Risultati dei confronti nello spazio

Un problema dell'approccio distribuito risulta essere la possibile ricostruzione del medesimo marker due o più volte. Questo accade qualora un marker venga individuato e ricostruito da rette appartenenti a rami diversi dell'albero. Una possibile soluzione è eseguire un controllo alla fine del processo dove si individuano i punti che individuano il medesimo marker e se ne opera una media.
Confrontando i risultati ottenuti nelle implementazioni degli algoritmi distribuiti con quello centralizzato si nota immediatamente i diversi tempi di elaborazione. Ci si aspettava un miglioramento delle prestazioni mediante l'approccio distribuito, sebbene la differenza è molto più elevata di quello che ci si poteva aspettare.
L'algoritmo centralizzato associa a due a due i raggi prodotti dalle telecamere senza eliminare le rette utilizzate per la ricostruzione dei marker, ma bensì continua ad utilizzarle per confronti successivi.
Per questo il tempo d'esecuzione è maggiore a confronto con il tempo utilizzato delle funzioni distribuite e aumenta al crescere del numero di telecamere e dal numero di marker. Ricordando che il residuo nelle tabelle corrisponde al raggio della sfera rappresentante il volume di incertezza nell'individuazione del marker, si nota che, nella maggior parte dei casi, il marker reale non si trova all'interno del volume di incertezza individuato. Questo significa che quest'ultima non è una buona caratterizzazione dell'incertezza.
La leggera differenza che presentano le prime tre implementazioni è dovuta alle diverse operazioni per la mediazione dei punti per quanto riguarda la prima implementazione e per poter calcolare i vari pesi per la ricostruzione nella terza.

7 Conclusioni e sviluppi futuri

In conclusione, osservando i risultati ottenuti nelle simulazioni nel piano e nello spazio, si può affermare che l'approccio distribuito al passive motion capture permette una sensibile diminuzione del tempo di elaborazione.
Si noti che, sia nell'implementazione centralizzata che in quella distribuita, nel calcolo dei tempi di elaborazione degli algoritmi non si è tenuto conto delle funzioni per i calcolo dei centri e delle rette. Questo fatto è di notevole interesse in quanto, nell'approccio distribuito, le funzioni in questione vengono svolte in parallelo su ogni telecamera, per questo motivo l'aumento del tempo di elaborazione è relativamente basso. Al contrario, nell'approccio centralizzato, queste funzioni vengono elaborate in modo sequenziale dal calcolatore, per cui l'aumento del tempo di calcolo risulta proporzionale al numero di frame da elaborare. Questo può causare un aumento sensibile del costo computazione.
Nonostante i risultati ottenuti non si è in gradi di dire se l'implementazione distribuita del passive motion capture può essere svolto in tempo reale. Questa insicurezza è dovuta al fatto che i tempi di elaborazione delle simulazioni Matlab sono solamente indicativi. Per verificare le reali possibilità dell'approccio distribuito risulta necessario testarne una implementazione in linguaggio C, e verificare se si riesce ad ottenere un comportamento real-time.

Per quanto riguarda la rappresentazione dell'incertezza nella ricostruzione dei marker, sia nel piano che nello spazio, i risultati ottenuti non sono stati soddisfacenti.
Oltre al fatto di non essere stati in grado di individuare una relazione tra numero di pixel ed

errore di ricostruzione, per i motivi precedentemente forniti nell'opportuna sezione, non si è stati in grado di determinare un'area di incertezza che contenesse il marker reale. Anche in questo caso i motivi possono essere molteplici. Un motivo può essere un non corretto funzionamento delle funzioni per generare i piani immagine delle telecamere. Un altro motivo può essere determinato dalla precisione legata alla dimensione del pixel del piano immagine, questo fatto non viene preso in considerazione nel corso del progetto. Tuttavia più piccoli sono i pixel più è facile che essi non si accendano, come spiegato in appendice. Per questo motivo risulta necessario trovare un compromesso accettabile.
Soprattutto per quanto riguarda quest'ultima parte del progetto risulta necessario un ulteriore approfondimento.

Nel corso del progetto si è voluto implementare l'algoritmo ai minimi quadrati per ricostruire la posizione dei marker nello spazio, parallelamente a quanto fatto nel caso bidimensionale. Si è implementato l'algoritmo ai minimi quadrati per trovare il punto di minima distanza fra al più tre rette. Per motivi di tempo non si è riusciti ad implementare un algoritmo ai minimi quadrati per la ricostruzione dei marker, quindi risulta necessario un ulteriore approfondimento e sviluppo di questa tecnica.

A Appendice

A.1 Tipologie di telecamere digitali

Le telecamere digitali forniscono all'uscita un segnale video digitale, cioè una sequenza di numeri che corrispondono al valore di luminosità o colore dei pixel. Per utilizzare le telecamere digitali si possono usare dei appositi frame grabber (schede di acquisizione) che sono naturalmente diversi da quelli delle telecamere analogiche. Il vantaggio di utillizzare queste telecamere è il loro minor rumore rispetto a quelle analogiche, infatti il segnale non deve essere convertito in un segnale standard per poi essere riconvertito in forma digitale all'interno del computer.
Una parte essenziale e molto importante per la telecamera è il sensore ottico, usato per la generazione dell'immagine, trasformandola in un segnale elettrico. I sensori CCD (*Charge Coupled Device*, dispositivi ad accoppiamento di carica) sono i più comunemente utilizzati nelle telecamere. Il principio operativo si basa su quantità finite di cariche elettriche, detti 'pacchetti', generate in posizioni specifiche nel semiconduttore al silicio (*fotoelementi*). Tali 'pacchetti' interagiscono con gli elettroni liberi generati dell'effetto fotoelettrico del semiconduttore al silicio: mediante tale effetto vengono generate quantità di elettroni liberi nelle zone illuminate in funzione del flusso di fotoni incidenti l'area. Pertanto, nella singola cella, il numero di elettroni liberi generati è direttamente proporzionale all'area ed al tempo di esposizione. Premesso questo, il processo di formazione dell'immagine comprende le seguenti fasi:

1. Il sensore resta esposto alla luce per un certo periodo. Ogni cellula accumula tante più cariche quanto maggiore è l'intensità della luce che la colpisce. L'intervallo di tempo nel quale il sensore accumula cariche viene definito **tempo di integrazione**.

2. Si blocca il processo di accumulo di cariche e si trasferiscono quelle accumulate in precedenza nell'elettronica di lettura.

Questi elementi sono accoppiati (coupled) in modo che ognuno di essi, sollecitato da un impulso elettrico, possa trasferire la propria carica ad un altro elemento adiacente, come si può vedere nella sequenza di Figura 44. Inviando al dispositivo (device) una sequenza temporizzata d'im-

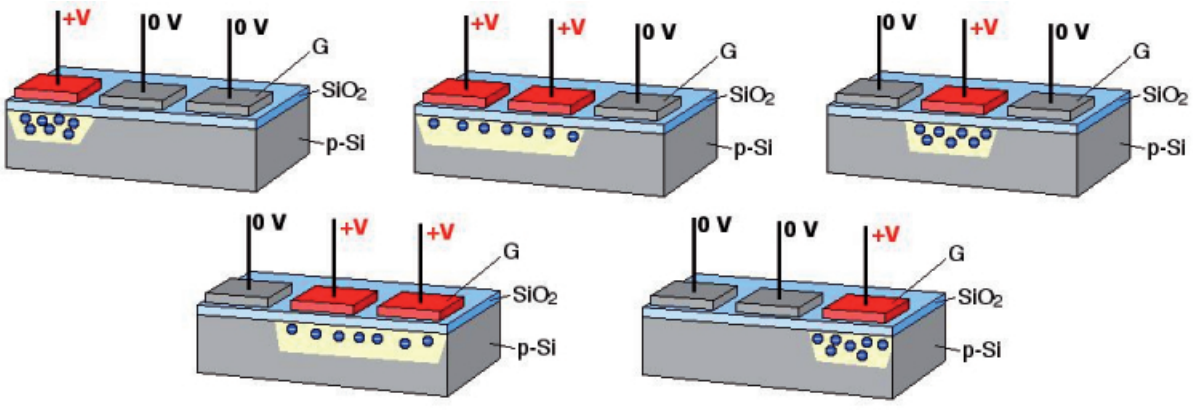

Figura 44: Sequenza di eventi, spostamento di carica in tre pixel consecutivi

pulsi, si ottiene in uscita un segnale elettrico grazie al quale è possibile ricostruire la matrice dei

pixel che compongono l'immagine proiettata sulla superficie del CCD stesso. Questa informazione può essere utilizzata direttamente nella sua forma analogica, per riprodurre l'immagine su di un monitor o per registrarla su supporti magnetici, oppure può essere convertita in formato digitale per l'immagazzinamento in file che ne garantiscano il riutilizzo futuro.
Nonostante le ottime caratteristiche dei sensori CCD essi stanno per essere soppiantati dagli oramai molto diffusi sensori CMOS (Complementary Metal Oxide Semiconductor). Infatti quest'ultima è una tecnologia emergente che si sta pian piano imponendo sul mercato per i suoi numerosi vantaggi. I sensori CMOS, che utilizzano lo stesso principio di trasformazione dell'immagine dei sensori CCD, sono molto economici da produrre essendo una tecnologia dominate nel mercato dei processori e consumano molto meno dei sensori CCD. Inoltre possono essere agevolmente incorporati circuiti per il condizionamento del segnale. I difetti di quest'ultimi sono la maggior sensibilità al rumore e alle interferenze, quindi producono un'immagine qualitativamente inferiore rispetto ai sensori CCD. La loro differenza può risultare trascurabile dipendendo dalle varie applicazioni cui si devono utilizzare. Per esempio in campo fotografico, la qualità è un elemento molto importante, quindi di prediligono i sensori CCD.
Nella maggior parte delle telecamere industriali è possibile cambiare il tempo di integrazione, utilizzando un comando di *Shutter*. Normalmente lo shutter è regolabile tramite degli switch rotativi posti all'interno della telecamera oppure, nei modelli più sofisticati direttamente tramite software. Esso è misurato in secondi (per esempio 1/50, 1/100, 1/1000 di secondo). Diminuendo il tempo di integrazione si ottine un'immagine più scura, in quanto ogni cella riesce a raccogliere meno luce, ma in caso di scene in movimento l'immagine risulta più ferma.
I sensori si possono suddividere in due categorie: **sensori lineari** e **sensori matriciali**. I sensori lineari sono utilizzati per acquisire un'immagine mono o bidimensionale. Un sensore lineare

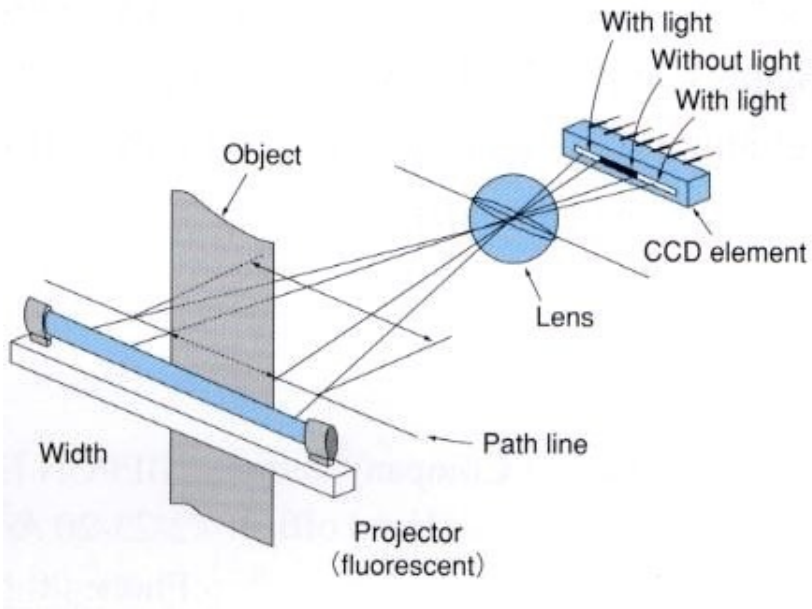

Figura 45: Sensore lineare.

è formato da una singola riga di elementi fotosensibili affiancati (pixels) pertanto, a differenza dei sensori di area (o matriciali) che acquisiscono quadri (frames), l'acquisizione avviene linea

per linea. Una singola linea di scansione può essere considerata come una mappatura monodimensionale della luminosità dei singoli punti della linea di osservazione. Una scansione lineare genera una linea che mostra sull'asse Y la luminosità di ogni punto in livelli di grigio (da 0 a 255 livelli). Una variazione improvvisa del livello di grigio di un singolo punto corrisponde ad un punto del contorno di un oggetto o alla presenza comunque di una variazione di colore o di aspetto. Il rilevamento di tale variazione consente pertanto di avere una misura di precisione consentita anche dalla risoluzione elevata del sensore lineare assai più alta di quella di un sensore di area. La lunghezza di un sensore lineare dipende dalla dimensione del singolo pixel e dalla risoluzione. Maggiore è la lunghezza del sensore, maggiori sono le dimensioni dell'obiettivo.Vedi Figura 16.

I sensori matriciali sono simili ai sensori lineari, con la differenza che i fotoelementi sono disposti in forma matriciale e che esiste un registro di trasferimento tra le colonne di fotoelementi.
Le risoluzioni dei sensori di area vanno da 32×32 a 256×256 elementi per sensori a media risoluzione. I dispositivi a più alta defizionione hanno una risoluzione di 1024×1024 elementi o più.

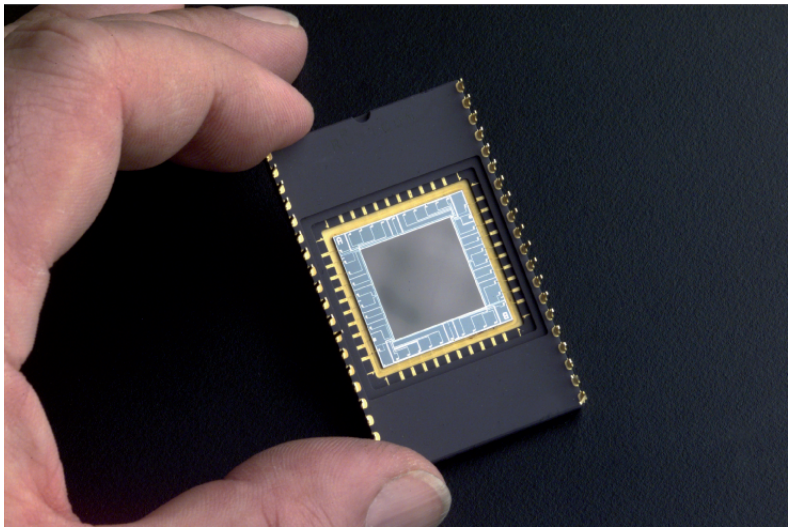

Figura 46: Sensore matriciale.

Le *dimensioni del sensore* sono al più in frazioni di pollici oppure di mm, come si può vedere in Figura 34.

Un'importante metodo per l'acquisizione delle immagini è la scansione progressiva, cioè l'informazione relativa all'immagine viene accumulata simultaneamente e successivamente scaricata linea per linea o in modo sequenziale, non interlacciato. Il risultato è un immagine che, qualora venga acquisita con un otturatore veloce, presenta una risoluzione piena sia verticalmente sia orizzontalmente.
Una telecamera tradizionale interlacciata acquisisce ogni semiquadro (1 quadro = 2 semiquadri) con una risoluzione verticale dimezzata; ciò in quanto l'acquisizione avviene tramite due semiquadri sequenziali (nel primo semiquadro vengono scandite le linee dispari, nel secondo le linee pari). Nel caso di un 'acquisizione dinamica dell'immagine, nel momento in cui il secondo

semiquadro viene immagazzinato e scandito il soggetto si è già mosso. Il risultato è un'immagine mossa una volta che i due semiquadri vengono combinati insieme per formare un'immagine interlacciata. Ciò viene eliminato con l'adozione della scansione progressiva. La scansione progressiva viene generalmente accoppiata con un otturatore elettronico. In sostanza la velocità di acquisizione viene selezionata variando il tempo di accumulo della carica su un singolo pixel. Le telecamere a scansione progressiva liberano l'utilizzatore dai vincoli dello standard televisivo permettendo il funzionamento dell'otturatore elettronico ad elevata risoluzione e un'acquisizione dinamica con una piena risoluzione verticale. Lo standard televisivo non è applicabile ad esempio a risoluzioni di 1024 x 1024 pixel o superiori.
Il numero e la dimensione fisica del pixel determina la risoluzione effettiva della telecamera, cioè la dimensione del più piccolo particolare distinguibile. Per avere una elevata risoluzione sono necessari molti pixel, ma riducendo la loro dimensione si riduce anche la loro capacità di raccogliere la luce e quindi la sensibilità.

I pixel possono essere sensibili a tutta la banda visibile e in questo caso la telecamera si dice *monocromatica*. Questo tipo di telecamera si basa sul pricipio dei livelli di grigio, cioè la cella costituisce il livello di grigio dell'immagine in quel punto, di conseguenza un punto molto scuro avrà un basso livello di grigio, viceversa per i punti molto chiari.Vedi Figura 47.
Nelle telecamere a colori il procedimento è leggermente differente, esse non sono altro che delle telecamere monocromatiche a cui sono stati posti dei filtri, detti *filtri di Bayer*, di colore rosso verde e blu di fronte al sensore. Il filtro di Bayer è costituito da matrici 2×2 di filtri per la luce con i tre *colori primari* ponendo il rosso e il blu in angoli opposti e nei rimanenti due spazi il colore verde, in accordo con la *Tristimulus Theory of color Perception*. In questo modo si rispecchia il sistema visivo umano, che determina la luminosità basandosi principalmente sulla luce verde essendo più sensibile ad essa.
Come possiamo vedere dalla Figura 48, all'uscita della telecamera l'elettronica deve fornire per ogni singolo pixel un determinato valore di colore. Per poter ricostruire l'immagine in modo esatto, l'elettronica della telecamera deve effettuare un'*interpolazione spaziale del colore*. Prendiamo come esempio il pixel rosso nell'angolo in basso a sinistra. Ci manca quindi il valore di verde e quello di blu. L'algoritmo valuta questi due valori mancanti tramite un'analisi dei pixel vicini a quello rosso. Nel nostro esempio troverà i pixel verdi con molto carico, ma quelli blu completamente scarico. Quindi il nostro pixel rosso è in realtà giallo.
Il vantaggio di questo modo di procedere sono i costi bassi. La qualità delle attuali telecamere a un sensore è sorprendentemente elevata. Perciò la maggior parte delle telecamere a colori si basa su questa tecnica.
Un'altra soluzione si basa sull'utilizzo di 3 sensori, in modo che ciascun sensore riceverà soltanto i fotoni filtrati: un sensore vedrà il colore rosso, il secondo il verde e il terzo blu. I tre canali di fotoni vengono separati da un prisma creato appositamente, vedi Figura 49. Questa soluzione porta nella pratica ad ottimi risultati, ma un considerevole svantaggio è però il costo. Se confrontiamo le 3 immagini digitali originali della telecamera a 3 sensori e quelle della telecamera a 1 sensore, queste ci sembreranno identiche. Ma questo vale effettivamente soltanto per i nostri semplici esempi. In pratica anche i migliori metodi per l'interpolazione del colore causano un effetto passa basso. Infatti le telecamere a 1 sensore forniscono delle immagini nettamente più sfocate rispetto alle telecamere a 3 sensori o alle telecamere monocromatiche. Questo effetto risulta particolarmente fastidioso su strutture delle immagini sottili e filiformi. Per applicazioni al di fuori della banda visibile esistono sensori, e quindi telecamere, sensibili alla radiazione

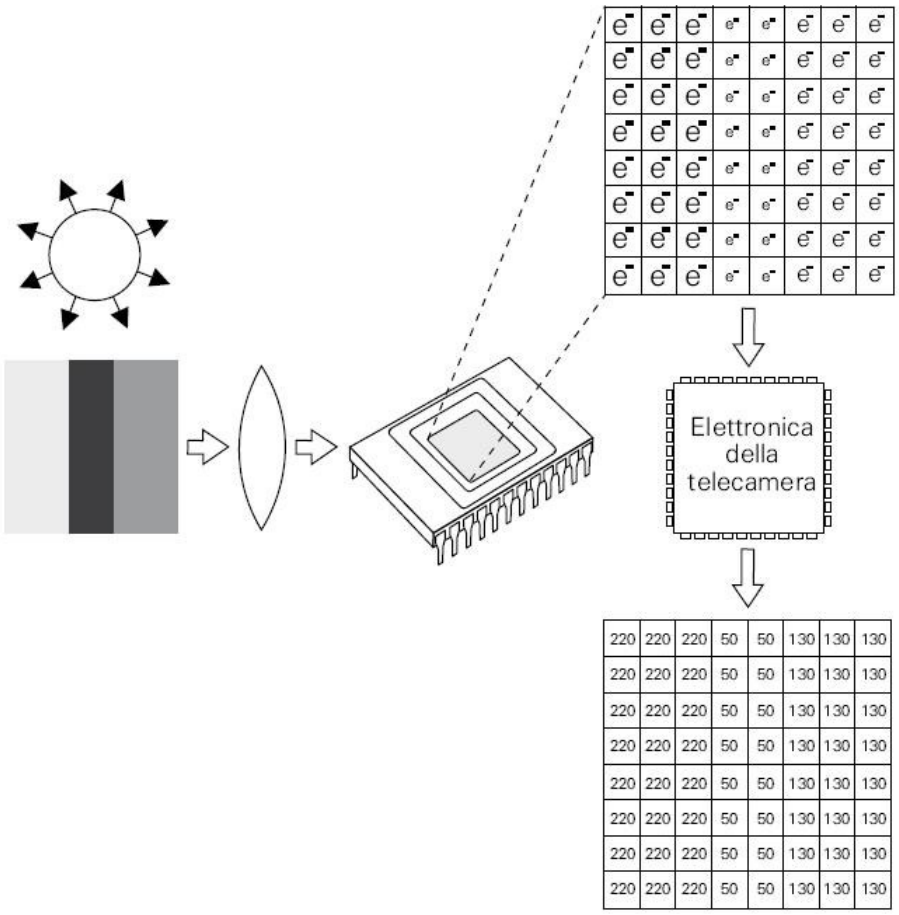

Figura 47: Funzionamento telecamera a scala di grigio.

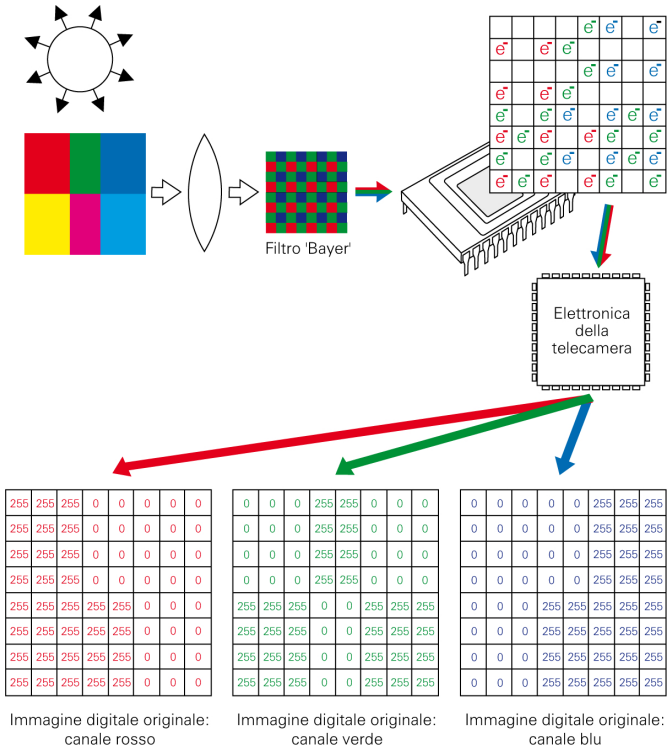

Figura 48: Funzionamento telecamera a colori con 1 sensore.

ultravioletta, a quella infrarossa o ai raggi X. In questi casi saranno necessarie ottiche adatte alla lunghezza d'onda impiegata.

A.2 Frame Grabber

Il Frame Grabber è un dispositivo in grado di acquisire e memorizzare l'immagine che viene trasmessa dal sensore della telecamera, sia analogica che digitale. Consente l'acquisizione di segnali singol-tap e multi-tap, permettendo di ricostruire l'immagine senza dover utilizzare molta memoria e CPU del computer, quindi senza perdere performance del sistema e velocizzando in questo modo l'elaborazione. La scheda di acquisizione permette di eseguire aggiustamenti delle immagini acquisite al fine di migliorarne l'apparenza, come la possibilità di modificare il gain, la temporizzazione o l'offset, cioè i livelli di bianco e nero.
Le caratteristiche comuni dei frame grabber sono:

- possibilità di memorizzare le immagini acquisite utilizzando un frame buffer;
- possibilità di generare segnali I/O di triggering per acquisizione dell'immagine o controllo di apparecchiature esterne;
- un interfaccia bus con il quale il processore può controllare e accedere ai dati.

La scelta del modello della telecamera influisce nella scelta del frame grabber e viceversa. Ovviamento la loro interfaccia deve essere compatibile. Bisogna considerare:

- acquisizione con telecamere B/N o colore

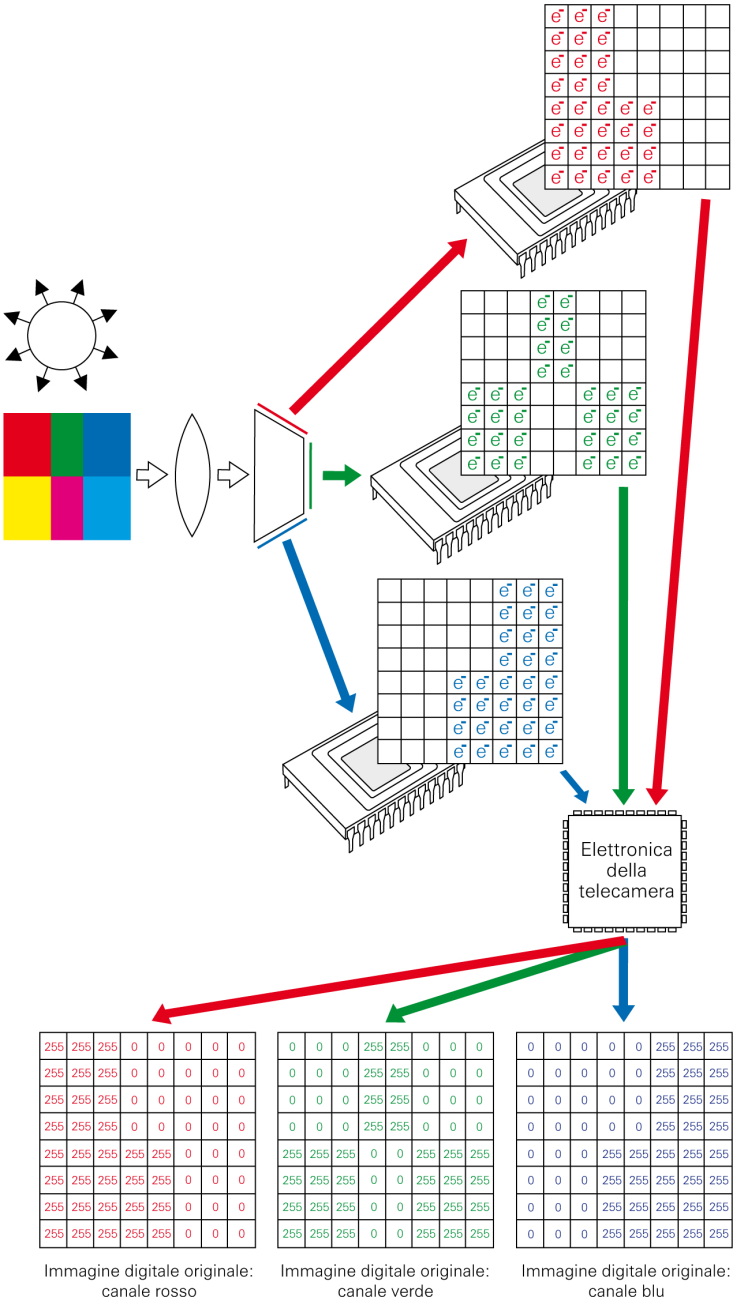

Figura 49: Funzionamento telecamera a colori con 3 sensori.

- compatibilità con il pixel rate
- compatibilità con la libreria software
- accuratezza nella digitalizzazione
- numero di telecamere che possono essere indirizzate
- utilità di controllo della camera tramite frame grabber
- sincronizzazione e triggering della telecamera
- disponibilità di processo on-board della scheda
- disponibilità di generare segnali I/O
- prezzo affidabilità e caratteristiche

Molti sistemi di visione utilizzano un segnale di trigger per controllare e sincronizzare le varie parti del sistema. Il trigger provvede a generare il segnale di scatto della telecamera e a generare il tempo di acquisizione del sensore. Nei più comuni sistemi di visione il segnale di trigger proviene da coordinate riprese dalla telecamera in azione oppure da segnali esterni dal computer, come un impulso spedito da una lampada strobo oppure da un segnale ricevuto da un sensore di posizione in una linea di produzione. In un sistema con una semplice configurazione il trigger non viene utilizzato e la telecamera acquisisce immagini in modo continuo, ma se nel sistema di visione è richiesta una certa immagine si può utilizzare il trigger per poter acquisire un singolo frame oppure una linea, facendo partire e fermare l'acquisizione.
In Figura 50 è illustrato l'acquisizione dei dati tramite il frame grabber. Generalmente si

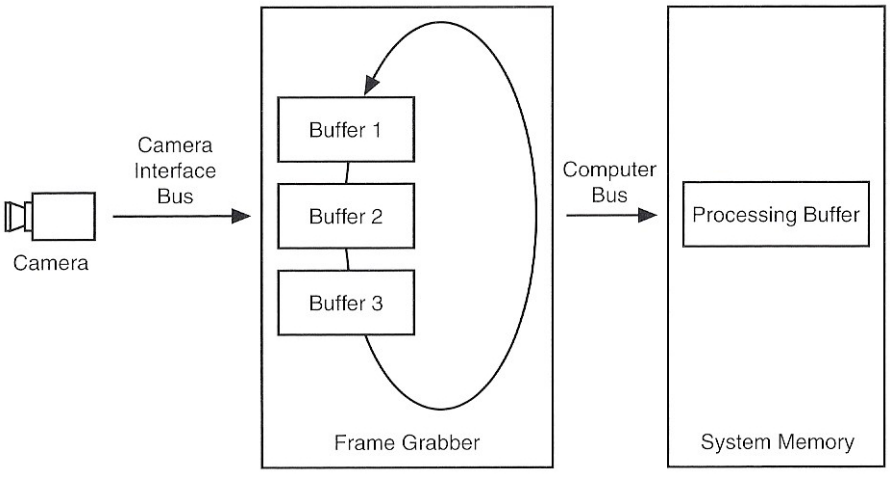

Figura 50: Acquisizione dell'immagine all'interno della memoria del frame grabber.

possono acquisire immagini con una velocità molto alta all'interno della memoria del frame grabber per poi trasferirli nella memoria del computer o nella memoria del sistema attraverso bus molto lenti come il PCI. Se c'è un ritardo associato con il bus PCI, il buffer nella scheda continua ad acquisire finchè le risorse della memoria sono disponibili. Le memoria utilizzano la

forma FIFO con la possibilità di memorizzare poche linee video, con 4 KB, oppure memorizzare parecchie immagini con 16-80 MB. Con schede dedicate si possono raggiunge elevate velocità di frame rate in periodi di tempo molto piccoli. Il bus PCI può sostenere solamente 100 MB/s e il frame grabber può essere scelto per un maggiore velocità per esempio 32 bits/pixel con 50 Hz risulta 200 MB/s di dati. In questo modo si possono superare le limitazioni del bus PCI senza la perdita di dati finchè la memoria del frame grabber è disponibile. Con questo dispositivo sono possibili molte interessanti pre-elaborazioni dell'immagini.
Per esempio il frame grabber può essere programmato per individuare una determinata area dell'immagine e eliminare l'area al di fuori di essa. Questa tecnica è definita come ROI (Region of Interest). In questo modo si può acquisire una finestra d'immagine specifica in modo tale da diminuire il frame rate poichè l'immagine è più piccola, di conseguenza ci sono meno dati da trasferire ed elaborare per ogni acquisizione, migliorando le performance dell'algoritmo.
Interessante è la possibilità di programmarlo a piacimento, quindi alcune funzioni viste in precedenza possono essere direttamente implementate nel frame grabber senza dover sovraccaricare la CPU del computer.