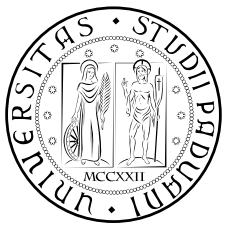


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCATRONICA

TESI DI LAUREA MAGISTRALE

Analisi dei metodi di stima della posa basati su marker ArUco e sistema di visione OptiTrack

Relatore: Prof. Roberto Oboe

Correlatore: Prof. Angel Valera

Laureando: Davide Corrata
2054079

ANNO ACCADEMICO: 2023-24

SOMMARIO

L'accurata stima della posizione di un corpo é cruciale in vari campi e spesso coinvolge l'uso di sistemi di Motion Capture. Recentemente, si é assistito ad un forte sviluppo di questa tecnologia,in particolare nei sistemi basati su telecamere, grazie ai progressi nella visione artificiale, ma, sebbene queste tecniche forniscano risultati accurati nella stima della posa, comportano costi elevati e richiedono significative risorse computazionali. Come alternativa economica, i fiducial-marker come ArUco hanno guadagnato popolaritá per la loro semplicitá, le basse esigenze computazionali e l'adattabilitá, tuttavia, in letteratura sono presenti dati sperimentali limitati sulla precisione dei sistemi di Motion Capture basati su di essi. Questo studio mira quindi a colmare la lacuna evidenziata, conducendo un'analisi sperimentale delle prestazioni di un sistema di Motion Capture basato su marker ArUco, confrontandolo con un moderno sistema commerciale sviluppato da Optitrack. L'apparato basato su ArUco in questo studio é realizzato utilizzando inizialmente una webcam ELP e successivamente una telecamera con visione stereo ZED 2i. La ricerca valuta gli errori di stima della posa in scenari sia statici che dinamici, analizzando diverse regioni della zona di lavoro della telecamera, esplorando anche la relazione tra l'errore di stima della posa e l'utilizzo di un numero crescente di marker ArUco.

"Well, here at last, dear friends, on the shores of the Sea comes the end of our fellowship in Middle-earth. Go in peace!"
J. R. R. Tolkien

RINGRAZIAMENTI

Ringrazio tutta la mia famiglia, che mi ha sempre sostenuto nelle mie scelte e mi ha dato la possibilitá di partire da casa piú volte, con la sicurezza di avere sempre un posto in cui poter ritornare.

Ringrazio le cittá di Riga e Valencia, per avermi permesso di vivere esperienze bellissime e di conoscere persone altrettanto preziose.

Ringrazio i miei amici di Albaredo ed i colleghi del Targa, per tutte le risate e momenti passati insieme, che fanno vivere la vita con un po' piú di leggerezza.

Ringrazio la TAF Yankees ed il motto "Soffri, Patissi, Triboea", per avermi trasmesso la voglia di lottare per raggiungere ogni obiettivo.

Infine, dei ringraziamenti speciali vanno al mio relatore e a José , senza i quali questo lavoro non sarebbe stato possibile.

INDICE

ELENCO DELLE FIGURE

1 INTRODUZIONE

1.1 DEFINIZIONE DEL PROBLEMA

Nella societá moderna, la raccolta di dati gioca un ruolo fondamentale nel mondo della ricerca e per riuscire ad agevolare l'essere umano nella quotidianeitá. Nel dettaglio, informazioni riguardo la posizione ed il movimento del corpo umano o di oggetti possono essere impiegate per lo sviluppo di robot e sistemi autonomi, i quali risultano essere in grado di sostituire un individuo in un lavoro pesante o migliorare la cura di pazienti. La soluzione migliore per la raccolta di tali dati é utilizzare la tecnologia *Motion Capture*, solitamente abbreviato come *MoCap*, la quale, basandosi su strumenti e approcci differenti, permette di rilevare la posizione di oggetti o persone all'interno di una specifica area di lavoro, con elevata precisione ed accuratezza. Di seguito sono presentati le principali tecnologie attualmente piú in uso in molti ambiti, da quello della ricerca fino all'ambiente medico.

Motion Capture inerziale

Il primo approccio descritto impiega la tecnologia inerziale, dove dei sensori inerziali sono applicati sul soggetto tramite appositi indumenti o supporti, e permettono di analizzarne i movimenti [52]. I sensori inerziali solitamente impiegati sono quelli MEMS, *Micro Eelectro Mechanical Systems* [30], i quali sono costituiti dall'insieme di:

- **Accelerometro**: é costituito da una massa di silicio solidale a delle molle, anch'esse in silicio, i cui spostamenti dovuti alle accelerazioni subite avvengono tramite la lettura di variazione della capacitá presente tra due elettrodi, uno fisso e l'altro connesso alla massa in moto.
- **Giroscopio**: ha una struttura simile al sensore di accelerazione, ovvero una massa collegata a delle molle, tuttavia ha lo scopo di misurare la velocitá angolare e, quindi, l'orientamento, sfruttando le informazioni sulla forza di Coriolis.

Nel mercato sono presenti modelli vari di sensori inerziali MEMS, come mostrato in Figura 1, adattati per essere posizionati in specifiche zone del corpo, in modo da aumentare il livello di comfort del soggetto e diminuire i tempi di applicazione. Tale metodo consente di ottenere dati con alta accuratezza, tuttavia la sua applicabilitá risulta limitata a causa della fragilitá dei componenti utilizzati e del complesso processo di preparazione.

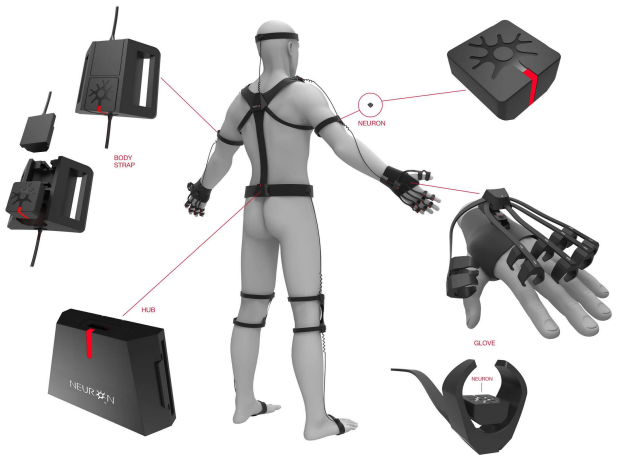


Figura 1: Esempio del posizionamento di sensori inerziali sul soggetto, utilizzando diverse tipologie di supporto [4]

Motion Capture basata su marker

Una soluzione alternativa é utilizzare la tecnologia Motion Capture basata su marker, i quali sono applicati sul soggetto e vengono rilevati da apposite telecamere, fornendo informazioni riguardo posizioni e movimenti in tutti e sei i gradi di libertá, composti da tre di traslazione e tre di rotazione.

Una delle aziende leader nella produzione di tali sistemi é OptiTrack, la quale offre prodotti che garantiscono alta accuratezza di misura, bassa latenza e una robusta integrazione software. In particolare, come si nota in Figura 2, é presente un supporto che, oltre a delimitare la zona di lavoro, sostiene e mantiene in posizione fissa le telecamere, le quali comunicando tra loro forniscono la posizione di tutti i marker rilevati, permettendo infine, tramite l'apposito software fornito, di ricavare la posa del soggetto. Le principali tecnologie

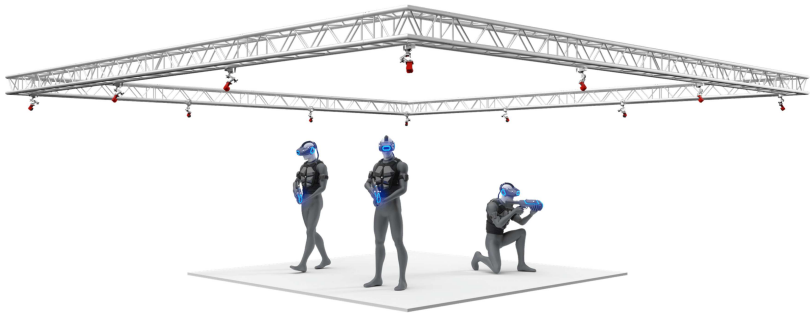


Figura 2: Esempio di sistema MoCap realizzato da OptiTrack, in questo caso utilizzato in un'applicazione di realtá virtuale [39]

che utilizzano i sistemi di Motion Capture basati su questo tipo di approccio sono [12]:

- **Optical-Passive:** prevede l'applicazione di marker riflettenti dalla forma sferica sul soggetto, i quali vengono rilevati per mezzo di un gruppo di telecamere ad infrarossi.
- **Optical-Active:** il soggetto indossa marker LED a infrarossi, che vengono rilevati tramite telecamere con medesima tecnologia. Si noti che é necessaria una fonte di alimentazione elettrica solidale al soggetto.

Le applicazioni dove questi sistemi sono utilizzati sono molte e spaziano in diversi campi, come la robotica [29], la riabilitazione clinica [27], la produzione cinematografica [57] e di videogiochi [48].

Motion Capture markerless

In tale metodologia, il soggetto viene registrato da una videocamera e contemporaneamente un algoritmo di tracciamento viene implementato in modo da rilevare la figura di uno o piú persone in ogni fotogramma del video [18]. In letteratura vi sono molti algoritmi sviluppati per ottimizzare diversi aspetti del processo, per esempio la velocitá di esecuzione, l'accuratezza delle misure raccolte oppure la robustezza [10]. I vantaggi e le potenzialitá offerte da tale approccio sono molte e, una tra tutte la quasi totale assenza di tempo di preparazione del soggetto, che permettono di favorire la tecnologia markerless piuttosto di altre. Un ulteriore aspetto che negli ultimi anni ne ha incentivato lo studio, é il progresso riscontrato nelle prestazioni delle CNN, acronimo di *Convutional Neural Network*, su cui si basano molti algoritmi, e la presenza in larga scala di dataset a disposizione, grazie ai quali si possono ottenere modelli sempre piú accurati. Attualmente tuttavia, la precisione delle misure di posizione ed orientamento acquisite tramite tale tecnologia rimangono comunque di qualitá minore rispetto ai dati raccolti con sistemi di Motion Capture basati su marker, caratteristica che ne compromette l'utilizzo in specifiche applicazioni che richiedono accuratezze elevate.

Come appena descritto, vi sono molte tecnologie diverse che permettono l'acquisizione e analisi del movimento di oggetti e persone nello spazio tramite sistemi che utilizzano la visione artificiale. In particolare, i sistemi basati su marker Optical-Passive sono largamente preferiti rispetto gli altri nelle applicazioni di laboratorio [13], grazie alla loro praticitá ed efficienza. I prodotti realizzati da OptiTrack per MoCap basati su marker sono accessibili con uno sforzo economico relativamente contenuto e permettono di ottenere dati precisi ed affidabili [55, 16]. A discapito delle prestazioni eccellenti tuttavia, i sistemi di Motion Capture basati su telecamere rimangono comunque piuttosto costosi e necessitano di molte risorse computazionali, il che li rende difficilmente applicabili nelle situazioni quotidiane.

1.2 OBIETTIVO DEL LAVORO

L'integrazione di fiducial-marker nei sistemi di MoCap permette di abbattere i costi e di ridurre la complessitá computazionale, andando peró a diminuire la precisione delle misure. Esistono diversi tipi di marker, contenuti in dizionari differenti, tra cui i principali sono ARTag, AprilTag, ArUco e STag [43]. Tra tutti, quelli ArUco emergono grazie alla loro semplicitá, facilitá di elaborazione e capacitá di adattamento a illuminazione non uniforme [47]. Inoltre, sono stati analizzati in molti studi, dove vengono approfondite l'accuratezza di misura della posa tramite simulazione [2], oppure al variare dell'orientamento [33]. In letteratura tuttavia, non sono presenti informazioni riguardo il funzionamento di tali marker nelle diverse zone di lavoro della telecamera dal punto di vista sperimentale.

Questo studio propone quindi una analisi sperimentale di un sistema di MoCap basato su marker ArUco, paragonandolo ad un sistema commerciale realizzato da OptiTrack in modo da valutarne l'accuratezza all'interno dell'area di lavoro. Di conseguenza, gli obiettivi fissati sono:

- Quantificare l'errore di stima della posa di un oggetto, dato da un sistema MoCap basato su marker ArUco nel caso statico e dinamico, utilizzando come misura di riferimento quella fornita da un sistema MoCap OptiTrack con tecnologia Optical-Passive
- Analizzare l'effetto dovuto all'impiego di un gruppo di marker nel decremento dell'errore di stima della posa.

Prima di iniziare la descrizione del sistema sviluppato al fine di raggiungere gli obiettivi posti, si riportano di seguito i concetti fondamentali riguardo i riferimenti cartesiani e la definizione della posa, indispensabili per la comprensione dei ragionamenti effettuati nei capitoli successivi, nonché utili alla definizione della simbologia utilizzata in questo documento.

1.3 SISTEMI DI RIFERIMENTO E POSA: ASPETTI GENERALI

Un sistema di riferimento cartesiano, o *frame*, é l'insieme formato da un punto O, detto *origine*, ed una terna di versori **i,j,k**, i quali hanno modulo unitario, sono ortogonali tra loro ed identificano direzione e verso degli assi. Un generico frame puó quindi essere indicato come $\mathscr{F} = \{O, (\mathbf{i}, \mathbf{j}, \mathbf{k})\}$ oppure, in alternativa, $\mathscr{F} = \{O, (\mathbf{x}, \mathbf{y}, \mathbf{z})\}$, utilizzando i nomi canonici per indicarne gli assi.

1.3.1 *Posizione ed Orientamento*

Una volta fissato un sistema di riferimento, é possibile utilizzarlo per esprimere la posizione nello spazio di un generico punto A tramite

delle *coordinate cartesiane* (x_A, y_A, z_A), le quali sono definite a partire dal vettore che congiunge l'origine del sistema di riferimento al punto stesso, ovvero $\overrightarrow{OA}$, ottenuto come combinazione lineare dei versori sopra definiti:

$$\overrightarrow{OA} = x_A\mathbf{i} + y_A\mathbf{j} + z_A\mathbf{k}$$

Considerando ora due frame:

- $\mathscr{F}_W = \{O_W, (\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W)\}$: sistema di riferimento globale e fisso
- $\mathscr{F}_A = \{O_A, (\mathbf{x}_A, \mathbf{y}_A, \mathbf{z}_A)\}$: sistema di riferimento centrato nel punto A

é possibile definire un'*orientamento* di $\mathscr{F}_A$ rispetto $\mathscr{F}_W$, il quale puó essere descritto in diversi modi [24, 54].

Il metodo piú semplice ed immediato é utilizzare gli *angoli di Eulero*, secondo cui l'orientamento viene completamente descritto tramite il vettore $\boldsymbol{\delta} = [\phi \quad \theta \quad \psi]$, dove la prima componente indica l'angolo della rotazione dell'intero frame A intorno all'asse $\mathbf{x}_W$, la seconda l'entitá della rotazione intorno all'asse $\mathbf{y}_W$ ed infine la terza é l'angolo corrispondente alla rotazione intorno all'asse $\mathbf{z}_W$, Figura 3a.

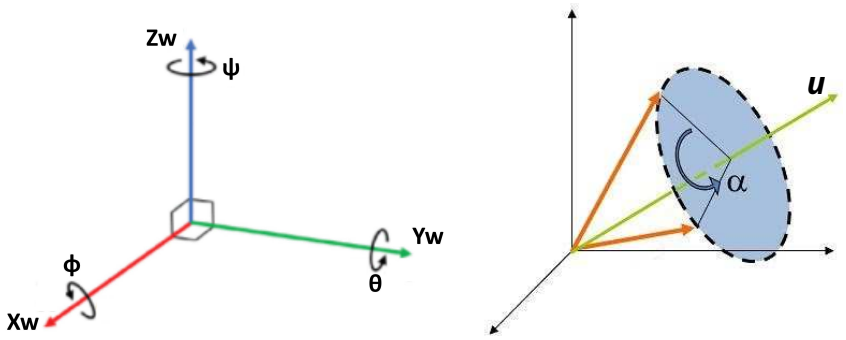
Una seconda rappresentazione prevede l'impiego di una *matrice di rotazione* $\mathbf{R}(\boldsymbol{\delta}) = {}^W\mathbf{R}_A \in \mathbb{R}^{3x3}$. Tale matrice é data dalla combinazione delle tre rotazione elementari descritte in precedenza, ciascuna delle quali indica lo spostamento di un certo angolo attorno un singolo asse del sistema di riferimento $\mathscr{F}_W$ ed é anch'essa rappresentabile tramite una matrice quadrata di dimensione 3. Dalla letteratura sono note queste matrici elementari, le quali vengono riportate di seguito:

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{1}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Per poter ricavare la matrice di rotazione complessiva, occorre definire un ordine secondo cui effettuare le rotazioni e, di conseguenza, l'ordine con cui effettuare le moltiplicazioni tra matrici. Una delle



(a) (b)

Figura 3: Orientamento rappresentato tramite: (a) angoli di Eulero (b) rappresentazione geometrica [5]

convenzioni piú utilizzate é ZYX, da cui, sostituendo le opportune definizioni precedenti, si ottiene:

$$
{}^{W}\mathbf{R}_{A} = \mathbf{R}(\boldsymbol{\delta}) = \mathbf{R}_{z}(\psi) \cdot \mathbf{R}_{y}(\theta) \cdot \mathbf{R}_{x}(\phi) =
$$

$$
\begin{bmatrix}
c(\psi)c(\theta) & c(\psi)s(\theta)s(\phi) - s(\psi)c(\phi) & c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \\
s(\psi)c(\theta) & s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi) & s(\psi)s(\theta)c(\phi) - c(\psi)s(\phi) \\
-s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi)
\end{bmatrix} \tag{4}
$$

dove sono state usate le abbreviazioni $c()$ ed $s()$ rispettivamente per le funzioni trigonometriche seno e coseno. Molto diffusa é anche la convenzione XYZ, utilizzata principalmente per descrivere l'orientamento di un oggetto che si trova in volo ed equivalente a quella appena descritta in termine di rotazione complessiva. L'unica differenza, oltre l'ordine delle moltiplicazioni effettuate tra le matrici elementari, sono i nomi che vengono attribuiti agli angoli stessi, i quali sono, con riferimento ai simboli sopra definiti:

- ϕ prende il nome di *angolo di Roll*
- θ prende il nome di *angolo di Pitch*
- ψ prende il nome di *angolo di Yaw*

Infine, una terza modalitá per descrivere una rotazione nello spazio é quello che applica la *rappresentazione geometrica*, tramite la quale essa viene definita come una rotazione di un certo angolo α attorno ad uno specifico vettore $\mathbf{u} \in \mathbb{R}^3$, come mostrato in Figura 3b. In particolare:

- $\hat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|} \in \mathbb{R}^3$: indica il vettore di modulo unitario attorno al quale avviene la rotazione dell'interno sistema di riferimento
- $\alpha = \|\mathbf{u}\| \in \mathbb{R}$: rappresenta il valore dell'angolo di rotazione

Si nota tuttavia che non vi é un legame immediato con le rappresentazioni precedenti, rendendo quindi necessaria una trasformazione in modo da poter effettuare operazioni ed analisi degli angoli.

La relazione tra rappresentazione geometrica ($\hat{\mathbf{u}} = [\hat{u}_1 \quad \hat{u}_2 \quad \hat{u}_3], \alpha$) e matrice di rotazione ${}^{W}\mathbf{R}_A$ viene fornito dalla *Formula di Rodrigues*, secondo la quale vale:

$$ {}^{W}\mathbf{R}_A = \mathbf{I}_3 + \mathbf{U}\sin(\alpha) + \mathbf{U}^2(1 - \cos(\alpha)) \tag{5} $$

dove

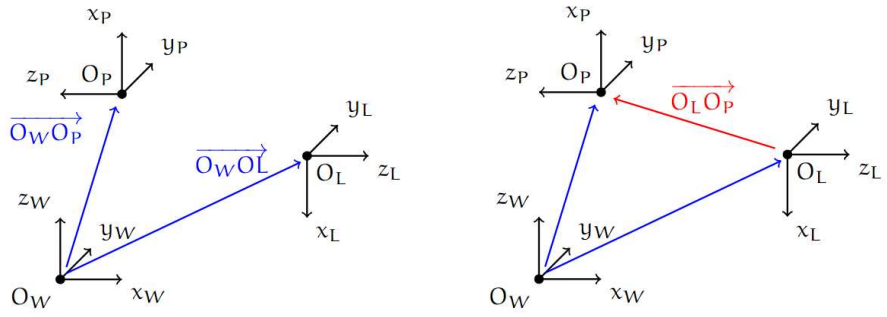
$$ \mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} 0 & -\hat{u}_3 & \hat{u}_2 \\ \hat{u}_3 & 0 & -\hat{u}_1 \\ -\hat{u}_2 & \hat{u}_1 & 0 \end{bmatrix} \tag{6} $$

1.3.2 *Definizione della Posa*

Ora che sono stati descritti i concetti fondamentali riguardo posizione ed orientamento e la simbologia utilizzata risulta chiara, é possibile passare alla loro applicazione. Con riferimento alla Figura 4a, vengono definiti tre sistemi di riferimento: un *World frame* $\mathscr{F}_W = \{O_W, (\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W)\}$, un *Local frame* $\mathscr{F}_L = \{O_L, (\mathbf{x}_L, \mathbf{y}_L, \mathbf{z}_L)\}$ e l'ultimo frame, centrato in un generico punto P $\mathscr{F}_P = \{O_P, (\mathbf{x}_P, \mathbf{y}_P, \mathbf{z}_P)\}$. La *posa* del sistema di riferimento $\mathscr{F}_P$, rispetto a $\mathscr{F}_W$ é definita come l'insieme di:

- $\mathbf{p}_P \in \mathbb{R}^3$: posizione del punto O_P in $\mathscr{F}_W$, indicata tramite il vettore $\overrightarrow{O_W O_P}$
- $\boldsymbol{\delta}_P \in \mathbb{R}^3$: orientamento di $\mathscr{F}_P$ rispetto a $\mathscr{F}_W$, descritta tramite angoli di Eulero

mentre, in modo analogo, la posa di $\mathscr{F}_L$ rispetto a $\mathscr{F}_W$ é data da $\mathbf{p}_L$ e $\boldsymbol{\delta}_L$, con medesimo significato dei simboli.



(a) (b)

Figura 4: Definizione della posa di: (a) $\mathscr{F}_L$ e $\mathscr{F}_P$ rispetto a $\mathscr{F}_W$ (b) $\mathscr{F}_P$ rispetto a $\mathscr{F}_L$

Come evidenziato nella sezione precedente, é possibile descrivere gli orientamenti impiegando delle matrici quadrate 3×3, ovvero utilizzando $\mathbf{R}(\boldsymbol{\delta}_P) = {}^{W}\mathbf{R}_P$ e $\mathbf{R}(\boldsymbol{\delta}_L) = {}^{W}\mathbf{R}_L$. Tali elementi, in aggiunta

a fornire una rappresentazione alternativa degli angoli di rotazione, permettono di effettuare un cambio del sistema di riferimento in cui sono espresse le coordinate di un punto.

Ad esempio, un vettore $\mathbf{v}_P$ che esprime la posizione di un generico punto nel frame $\mathscr{F}_P$, puó essere descritto ugualmente come un vettore $\mathbf{v}_W$, il quale utilizza come sistema di riferimento $\mathscr{F}_W$ ed é legato al precedente dalla seguente relazione:

$$\mathbf{v}_W = {}^W\mathbf{R}_P \cdot \mathbf{v}_P \tag{7}$$

Seguendo il medesimo ragionamento, con riferimento alla Figura 4b, note posizione e orientamento di $\mathscr{F}_P$ e $\mathscr{F}_L$ in $\mathscr{F}_W$ é possibile distaccarsi dal world frame e calcolare la posa di $\mathscr{F}_P$ in $\mathscr{F}_L$. La posizione viene facilmente calcolata partendo da un'analisi vettoriale:

$$\overrightarrow{O_L O_P} - \overrightarrow{O_W O_P} + \overrightarrow{O_W O_L} = 0 \tag{8}$$

da cui si ricava:

$$\overrightarrow{O_L O_P} = \overrightarrow{O_W O_P} - \overrightarrow{O_W O_L} \tag{9}$$

Per quanto riguarda l'orientamento invece, occorre effettuare la seguente operazione tra matrici:

$${}^L\mathbf{R}_P = {}^L\mathbf{R}_W \cdot {}^W\mathbf{R}_P \tag{10}$$

dove il primo elemento dell'operazione é pari all'inversa della matrice ${}^W\mathbf{R}_L$, che esprime il cambio di coordinate dal sistema di riferimento locale al world frame.

1.4 STRUTTURA DEL DOCUMENTO

Il presente documento si pone come obiettivo quello di fornire tutte le informazioni e conoscenze utili alla completa comprensione del lavoro svolto e riportato di seguito, prima di addentrarsi nel funzionamento vero a proprio del sistema, approfondendo le strumentazioni utilizzate ed i ragionamenti effettuati.

In particolare il lavoro é suddiviso come segue:

- **Capitolo 2: Strumentazione utilizzata**: in questo capitolo vengono descritti tutti gli strumenti impiegati nello sviluppo del sistema implementato, partendo dalle videocamere. Si affronta poi il sistema di visione OptiTrack presente in laboratorio, con il relativo funzionamento, ed infine le librerie ArUco ed OpenCV, indispensabili nell'acquisizione dei dati.

- **Capitolo 3: Procedura ed algoritmo**: si descrive in questa parte la procedura seguita durante le prove sperimentali eseguite, dividendola in quattro fasi: Fase di Set up, Fase di Detection, Fase

di Output e Fase di Post-Processing. Si osserva quindi l'intero processo, dalle operazioni iniziali di preparazione, all'elaborazione finale dei dati, passando per l'acquisizione delle immagini ed il formato di salvataggio dei dati raccolti.

- **Capitolo 4: Descrizione del software**: in questo capitolo si riportano e descrivono le principali funzioni implementate a livello software, utilizzando diversi linguaggi di programmazione. Anche in questo caso, per facilitarne la comprensione, si é suddivisa la spiegazione nelle medesime quattro fasi del punto precedente, con l'obiettivo di creare una analogia tra i contenuti.

- **Capitolo 5: Esperimenti e risultati**: si descrivono in questa sede le prove sperimentali condotte, nel caso statico e dinamico, con i relativi risultati. Si distinguono inoltre in base al sensore utilizzato per l'acquisizione delle immagini elaborate, in quanto sfruttano tecnologie differenti.

- **Capitolo 6: Conclusione e sviluppi futuri**: sono riportate le principali osservazioni effettuate durante lo studio condotto, seguite dagli studi futuri che, partendo da questo lavoro, avranno come obiettivo migliorare il sistema sviluppato, integrandolo in una specifica applicazione.

2

STRUMENTAZIONE UTILIZZATA

Per raggiungere gli obiettivi descritti nel capitolo precedente, ovvero di studiare le prestazioni di un sistema di Motion Capture basato su marker ArUco, si é realizzato il sistema mostrato in Figura 5, il quale prevede il posizionamento di una telecamera, prima una singola webcam e successivamente una telecamera con visione stereo, parallelamente ad un piano, sul quale vengono poi disposti i marker ArUco. Durante le prove sperimentali, le quali mirano a simulare condizioni sia statiche che dinamiche, vengono acquisite delle immagini, all'interno delle quali, utilizzando specifici software ed applicazioni, vengono rilevate posizione ed orientamento dei marcatori.

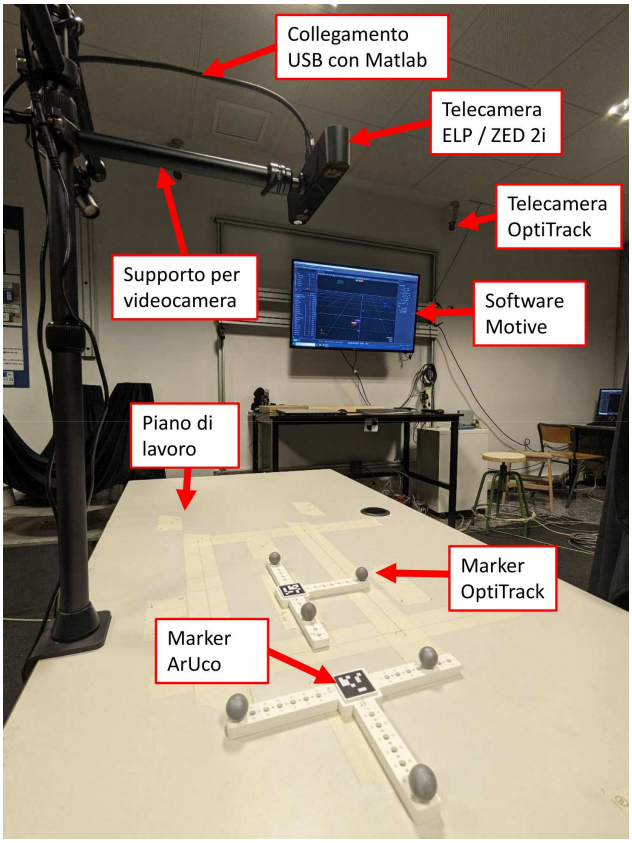


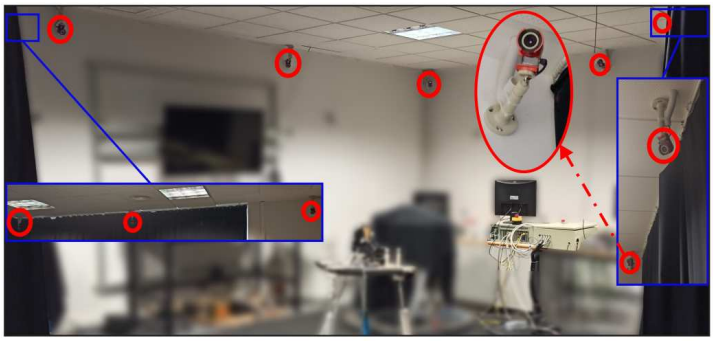
Figura 5: Immagine che riporta per intero il sistema sviluppato, nella quale si distinguono gli elementi che lo compongono.

Il presente capitolo si concentra sulla strumentazione impiegata, partendo inizialmente dall'approfondire il funzionamento del sistema OptiTrack utilizzato come riferimento per le misure effettuate. Successivamente verranno descritte le principali caratteristiche e differenze delle telecamere impiegate, soffermandosi prima su una bre-

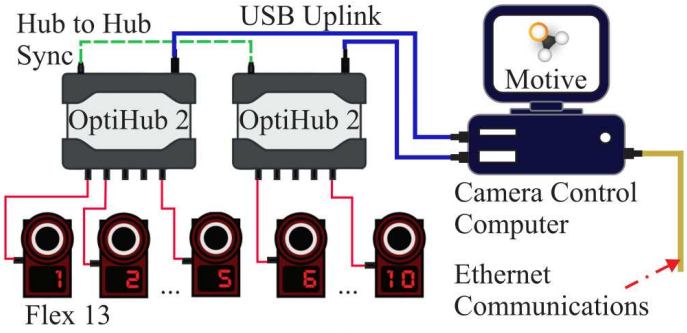
ve introduzione riguardo le terminologie concetti basilari nell'ambito delle immagini digitali. Infine si introdurranno i marker ArUco ed i due principali software sul quale si basa il funzionamento del sistema sviluppato.

2.1 SISTEMA DI MOTION CAPTURE OPTITRACK

Come descritto in precedenza, i sistemi di Motion Cappture possono essere basati su diverse tecnologie, andando a variare il tipo di sensori e marker. Il sistema a disposizione per questo lavoro [44], sfrutta la tecnologia optical-passive ed é composto da 10 telecamere ad infrarossi del modello Flex 13, Figura 6a, mentre i marker sono costituiti da sfere riflettenti di diametro 12.7mm. Alla frequenza di 120Hz, i sensori risultano avere una risoluzione pari a 1.3 megapixel e raggiungono un'accuratezza media dell'ordine del decimo di millimetro.



(a)



(b)

Figura 6: Sistema di MoCap OptiTrack utilizzato: (a) posizionamento delle telecamere, modello Flex 13 (b) architettura [44]

In Figura 6b invece, viene descritto il funzionamento di tale sistema, in cui sono presenti due OpiHub 2 per registrare le immagine catturate dai sensori stessi[1] ed inviare i dati raccolti al computer che li controlla, tramite USB ad alta velocitá. Qua, tramite il software *Mo-*

1 Si noti che il numero massimo di telecamere connesse ad un singolo device OptiHub é pari a 5

tive, anch'esso fornito da Optitrack, le immagini in due dimensioni provenienti dalle telecamere vengono processate, ricavando cosí la posa dei marker all'intero del workspace. In particolare, é possibile utilizzare Motive per creare oggetti virtuali, detti *rigid bodies*, partendo da un set di marker in numero compreso tra un minimo di 3 fino ad un massimo di 20. Infine, il programma presentato, offre la possibilitá di condividere via Ethernet la posa dei marker in tempo reale, il ché si é rivelato molto utile durante la conduzione degli esperimenti

Tale sistema, essendo presente da diversi anni all'interno del laboratorio dove é stato svolto questo studio, risulta giá collaudato e testato, permettendo quindi di conoscerne con sicurezza la qualitá delle misure acquisite.

2.1.1 *Software Motive*

Motive é il software fornito insieme al sistema OptiTrack ed é ottimizzato per acquisire ed elaborare dati provenienti da sistemi di Motion Capture in real time. Tale programma, permette di tracciare e raggruppare un elevato numero di marker contemporaneamente, fornendo la possibilitá di riconoscere e seguire il movimento di oggetti, persone o animali. Inoltre, l'algoritmo in esso contenuto é particolarmente preciso e robusto, garantendo misure molto accurate anche nelle situazioni in cui parte dei marker risultano non visibili dalle telecamere oppure parzialmente coperti. In Figura 7 viene mostrato un esempio di feedback dato dal software Motive, il quale, rilevando un set di 50 marker opportunamente posizionati sul soggetto all'interno dell'area di lavoro del sistema di visione OptiTrack, ricostruisce la sagoma ed i movimenti del corpo della persona rilevata, in tempo reale.

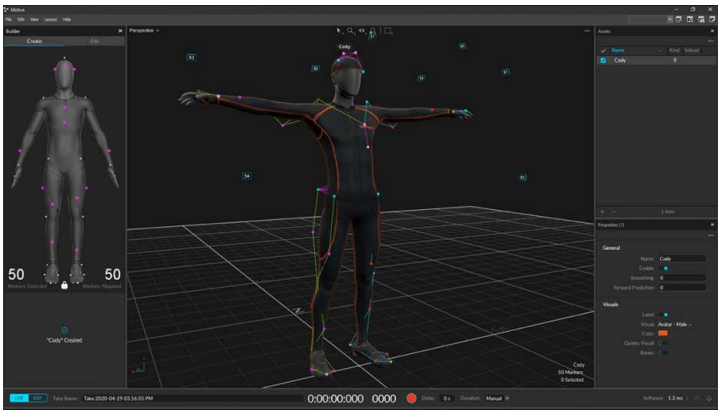


Figura 7: Esempio di utilizzo del software Motive, utilizzato per ricostruire i movimenti di un corpo umano partendo dalla rilevazione da parte di un sistema di visione OptiTrack di un set composto da 50 marker riflettenti [40]

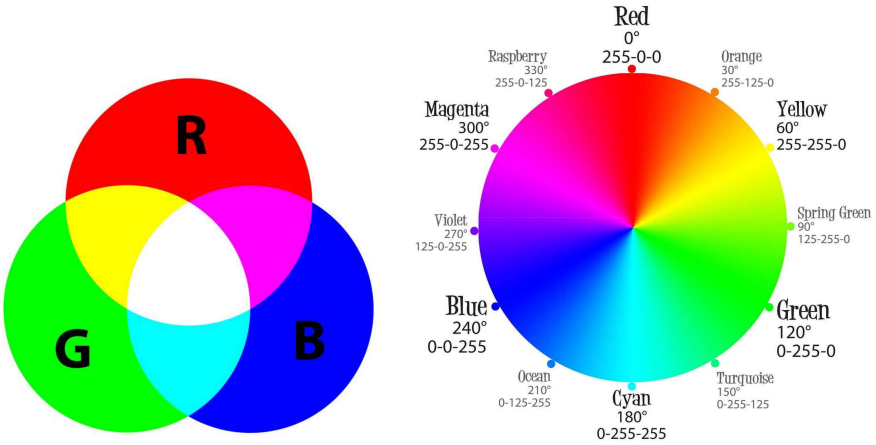
2.2 VIDEOCAMERE DIGITALI

Per descrivere le tipologie di sensori impiegate nelle prove sperimentali, occorre soffermarsi su alcuni concetti fondamentali nel mondo delle immagini digitali. La principale distinzione tra grandezze provenienti dal mondo reale e dati digitali é data dalla loro diversa continuitá: i dati analogici si muovono nel tempo continuo e hanno un infinito numero di possibili valori in ampiezza, mentre i dati digitali sono espressi in tempo discreto e possono assumere un numero limitato di valori in ampiezza. Per questa ragione, analizzare grandezze reali risulta piuttosto complesso e si rende necessaria un loro approssimazione digitale, che sia il piú accurata possibile.

Per quanto riguarda le immagini digitali, esse sono definite da una matrice $M \times N$, dove ogni elemento é detto *pixel*. Ogni pixel ha il ruolo di discretizzare una porzione dello spazio, rappresentandola tramite un livello di intensitá, il quale viene indicato da un valore compreso tra 0 e $2^n - 1$, dove n é il numero di bit disponibili per la quantizzazione. Il valore standard é $n = 8$ bit, secondo cui ciascun pixel puó variare la sua intensitá nell'intervallo $[0; 255]$, in cui l'estremo inferiore rappresenta la totale assenza di intensitá, ovvero corrisponde ad un pixel di colore nero nell'immagine, e l'estremo superiore rappresenta il valore massimo di intensitá, corrispondente nella matrice ad un pixel bianco. Nel contesto di una scala di grigi, i valori compresi tra i limiti fissati sono assegnati a diversi gradi di intensitá, la quale aumenta proporzionalmente al valore stesso.

Una telecamera digitale utilizza un sensore per misurare la quantitá di luce che ogni pixel riceve tramite la lente, assegnandogli di conseguenza il valore appropriato. Per catturare un'immagine a colori, uno degli standard piú utilizzati é il modello RGB (Red-Green-Blue), Figura 8a, che si basa sulla teoria di Thomas Young, secondo cui i coni che compongono la retina dell'occhio umano sono preferibilmente sensibili a tali tonalitá, le quali, opportunamente combinate, permettono di ottenere tutti i colori possibili. L'implementazione del modello RGB prevede la scomposizione di ogni pixel in tre *sub-pixel*, ciascuno assegnato ad uno dei tre colori primari, con la possibilitá di variarne l'intensitá. Come mostrato in Figura 8b, assegnando un determinato valore di intensitá ad ogni sub-pixel, utilizzando 8 bit per ciascuno di essi e combinandoli opportunamente, un singolo pixel puó assumere un'ampia sfumatura di tonalitá e colori diversi.

In questo studio, due diverse tecnologie di sensori digitali sono stati utilizzati: una singola videocamera e successivamente un sensore con visione stereo, le cui differenze verranno ora delineate.



(a) (b)

Figura 8: Standard RGB: (a) Modello (b) possibili colori assunti da un pixel utilizzando 8 bit [6]

2.2.1 *Videocamera digitale HD ELP*

La videocamera ELP utilizzata é il modello USBFHD01M-MFV, Figura 9a, progettato per catturare immagini e video dettagliati, che offre risoluzioni da 320×240 a 1920×1080 pixel, fornendo diversi livelli di accuratezza al variare della frequenza di acquisizione delle immagini. In particolare, alla frequenza di 100Hz, la risoluzione del sensore risulta pari a 640×480 pixel, e presenta un obiettivo varifocale da 2.8 a 12mm, che consente lunghezze focali regolabili offrendo flessibilitá nella cattura di diverse scene e dettagli[2]. É inoltre dotata di interfaccia USB che la rende compatibile con vari dispositivi, ed infine i suoi parametri possono essere facilmente impostati tramite poche righe di codice, motivo per cui é stata scelta per questo progetto.

(a) (b)

Figura 9: Telecamere digitali utilizzate durante le prove sperimentali: (a) telecamera HD ELP (b) telecamera stereo ZED 2i [9, 20]

2.2.2 *Stereo camera ZED 2i*

Per visione stereo si intende l'utilizzo di due sensori, entrambi puntati sulla stessa area di lavoro ma da due angolazioni differenti, le cui informazioni raccolte vengono successivamente condivise al fi-

2 Si veda la sezione 3.1.1 per un approfondimento sui parametri delle videocamere

ne di ottenere dati con accuratezza maggiore. Andando a combinare opportunamente le due immagini infatti, é possibile ottenere maggiori informazioni e precisione sulla misura della profonditá rispetto a quella ricavata utilizzando una singola telecamera.

Il sensore utilizzato in questo lavoro é la telecamera ZED 2i, prodotto da Stereolabs e parte della serie ZED. Le principali caratteristiche sono:

- **Visione stereo:** la presenza di due telecamere permette di catturare l'area di lavoro da due posizioni differenti, facilitando il calcolo delle informazioni nelle tre dimensioni
- **Sensori integrati:** a bordo sono presenti un barometro ed un magnetometro, che permettono allo strumento di conoscere l'ambiente circostante e la sua posizione in esso
- **Software Development Kit (SDK):** Stereolabs mette a disposizione una serie di funzioni per aiutare l'utente nelle fasi iniziali dell'utilizzo della telecamera e offre esempi di applicazione a diversi livelli di complessitá

Infine, la compagnia produttrice fornisce le informazioni e tutte le procedure necessarie per adattare il funzionamento dello strumento all'interno di vari software, tra i quali anche in Matlab [19].

In Tabella 1 sono riassunte le principali specifiche e differenze tra le videocamere impiegate.

Modello	ELP	ZED 2i
Risoluzione	640×480 a 100Hz	2x(662×376) a 100Hz
Interfaccia	USB a 5V	USB tipo C a 5V
Sensore	CMOS 2 megapixel	doppio CMOS 4 megapixel
Zoom ottico	da 2.8 a 12mm	2.1mm

Tabella 1: Caratteristiche principali delle telecamere utilizzate

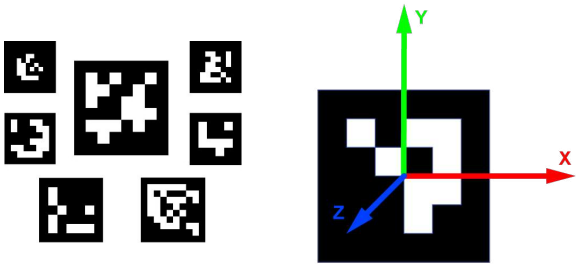
2.3 ARUCO

ArUco é una libreria open source che permette il rilevamento di marker appositamente realizzati all'interno di immagini catturate da telecamere digitali. Viene fornita in linguaggio di programmazione C++, ma sono presenti molte risorse e articoli per utilizzarla al meglio anche in altri ambienti [45, 14, 15].

2.3.1 *Marker*

In Figura 10a sono riportati degli esempi di marker sviluppati da Aruco, dove si nota che ciascuno di essi é caratterizzato da un bordo

esterno nero, il quale contiene un codice binario che viene univocamente associato ad un numero intero identificativo. Inoltre, grazie alla struttura unica, ad ogni marker puó essere associato un proprio orientamento, che viene indicato tramite un sistema di riferimento cartesiano, Figura 10b, oppure evidenziando l'angolo superiore sinistro. Sono disponibili marker di diverse dimensioni in termini di bit, partendo da 4×4 a 7×7, in quanto una maggior quantitá di codici puó essere prodotta utilizzando un numero piú alto di bit, a patto peró di avere a disposizione un'alta risoluzione della telecamera in modo da poterli distinguere. La variante scelta per questo progetto é quella di dimensione 6×6 bit, la quale permette di distinguere 250 codici diversi.



(a) (b)

Figura 10: Marker ArUco: (a) esempio di dizionari (b) orientamento di un marker [38, 8]

2.3.2 *Processo di rilevamento*

Il processo di rilevamento, o detection process, previsto dall'algoritmo sviluppato da ArUco consiste di pochi passi, una volta a disposizione l'immagine contenente uno o piú marker, Figura 11a:

- **Segmentazione:** l'immagine viene sottoposta ad un'analisi del valore di intensitá di ogni pixel, utilizzando una finestra di grandezza specifica ed una soglia specifica, Figura 11b.
- **Estrazione dei contorni:** tutti i contorni vengono estratti a partire dall'immagine segmentata, tramite uno specifico algoritmo [53], Figura 11c
- **Filtraggio dei contorni:** in questo passaggio, vengono scartati i contorni che non approssimano la forma ricercata oppure risultano troppo piccoli, Figura 11d. Inoltre, tramite regressione lineare vengono ricercate le posizioni degli angoli dei quadrati rilevati.
- **Estrazione del codice:** una volta rilevato il marker, ne viene analizzata la parte interna, andando a distinguere il codice che

lo caratterizza, Figura 11e. Per farlo, occorre prima rimuovere eventuali rotazioni oblique dell'immagine, raddrizzandola.

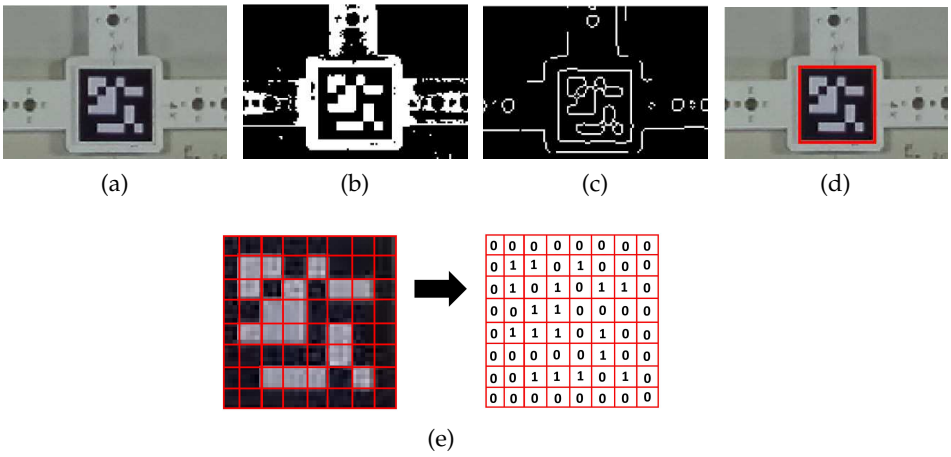


Figura 11: Processo di rilevamento ArUco: (a) immagine originale (b) immagine segmentata (c) estrazione dei contorni (d) contorni filtrati (e) estrazione del codice

Una volta terminata la procedura di rilevamento, per ogni marker vengono restituiti il suo numero identificativo, legato al codice binario, e la posizione degli angoli, da cui é successivamente possibile ricavare la posa del marker stesso, andando a risolvere un problema PnP (Perpective-n-Point). Tale problema, partendo dalla posizione nota di n punti, restituisce posizione ed orientamento dell'oggetto rispetto al sistema di riferimento solidale alla telecamera da cui é stata catturata l'immagine.

Si sottolinea che nel presente lavoro é stata utilizzata l'innovativa libreria di funzioni *aruconano*, piú veloce della versione precedente e che permette di rilevare marker tramite una semplice riga di codice.

2.3.3 *Applicazioni*

Attualmente, i marker realizzati da ArUco sono molto impiegati nelle applicazioni piú disparate, sia nell'ambito della ricerca che in sistemi commerciali. Uno degli utilizzi che risulta particolarmente utile é la possibilitá di effettuare la calibrazione di una telecamera, partendo dalla rilevazione di un set di marcatori [1], in particolare disposti a formare delle tabelle dette *ChArUco Boards*, di cui si riporta un esempio in Figura 12.

Un secondo ambito di applicazione é quello della robotica, dove, ad esempio, attraverso una rete di marker posizionati sulle pareti della stanza un robot mobile é in grado di orientarsi e muoversi con facilitá al suo interno [46], oppure per facilitare la manovra di atterraggio di un velivolo senza pilota a bordo, detto UAV, sigla di *unmanned aerial vehicle* [31].

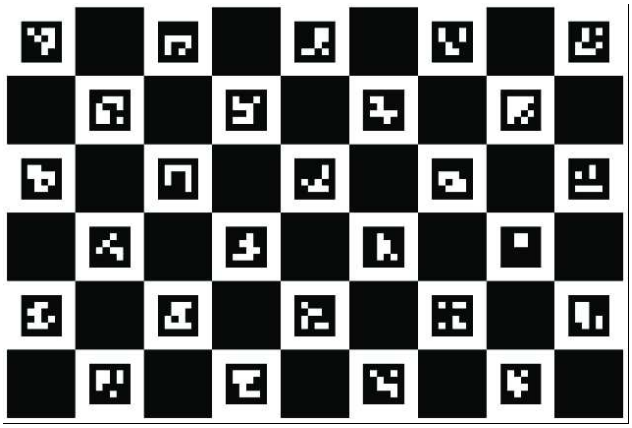
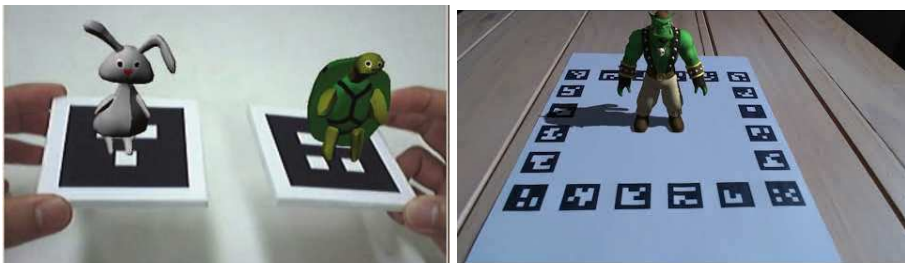


Figura 12: Esempio di una ChArUco Board [11]

Infine, i marker ArUco risultano ottimali nelle applicazioni di realtá aumentata, per sostituire al marker rilevato un'immagine a scelta [3], oppure sovrapporci degli elementi tridimensionali, come in Figura 13.



(a) (b)

Figura 13: Esempio di utilizzo dei marker ArUco per applicazioni di realtá aumentata, utilizzando (a) un singolo elemento [28] (b) una ChArUco Board [34]

2.4 OPENCV

OpenCV (Open Source Computer Vision Library) é una libreria software open source che mette a disposizione funzioni per visione artificiale e machine learning, in modo da creare uno standard in questo tipo di applicazioni e per agevolarne l'utilizzo in ampia scala. La prima versione fu presentata nel 2000, in occasione della "IEEE Conference on Computer Vision and Pattern Recognition" ma fu rilasciata ufficialmente solo sei anni piú tardi. Al momento la libreria include algoritmi classici e all'avanguardia, per applicazioni che vanno dall'identificazione di oggetti alla possibilitá di estrarre informazione 3D da un'immagine. Uno dei maggiori vantaggi offerti é la vasta comunitá presente alle sue spalle, che comprende utilizzatori individuali insieme a grosse aziende e che costantemente ne aggiorna i contenuti

[35].

OpenCV é principalmente progettato per essere utilizzato in C++, offrendo tuttavia supporto per altri linguaggi di programmazione, come Python e Java, tramite specifiche estensioni. La libreria é organizzata in moduli [37], tra cui i piú importanti sono:

- **core:** é il modulo basilare di OpenCV e contiene le strutture dati e le funzioni fondamentali, utilizzate anche nei moduli ausiliari.
- **imgproc:** contiene funzioni che permettono di elaborare immagini, effettuare trasformazioni geometriche e rilevarne le caratteristiche.
- **video:** sviluppato per analizzare video e tracciare oggetti all'interno di essi.
- **objdetect:** serve per rilevare codici a barre e QR, nonché classi predefinite dall'utilizzatore.

In aggiunta, sono presenti moduli progettati per applicazioni specifiche, tra cui il modulo *aruco*, che contiene funzioni appositamente realizzate per lavorare immagini con questo tipo di marker [36]. L'unica problematica riscontrata é che tale modulo utilizza i file sviluppati prima dell'innovativa libreria *aruconano*, di conseguenza, non tutte le funzioni in esso contenute sono state applicate in questo lavoro, richiedendo quindi un adattamento dei file non inclusi[3].

2.4.1 *Estensione Matlab per OpenCV*

Il software principale utilizzato per realizzare questo lavoro é stato MATLAB (MATrix LABoratory), il quale é un ambiente di programmazione specifico per effettuare operazioni su matrici e analizzare strutture di dati. Dato che richiede un linguaggio di programmazione proprio, per poter utilizzare le funzioni contenute nella libreria sopra descritta, si é reso necessario un passo intermedio passando per l'opportuna estensione [23], la quale permette di creare una connessione tra funzioni OpenCV, o generiche funzioni scritte in C++, e applicazioni in MATLAB, creando file di tipo *mex*. L'estensione richiede la presenza del pacchetto *Computer Vision Toolbox* e offre:

- File binari di OpenCV pre-compilati, i quali rimuovono la dipendenza dai file sorgente.
- Script MATLAB per creare file di tipo mex a partire da funzioni proprie di OpenCV.
- Conversioni dei tipi di dati tra gli ambienti OpenCV e MATLAB.

3 Si veda il capitolo 4

L'utilizzo di tale pacchetto di supporto ha inoltre facilitato l'utilizzo della stereo camera ZED all'interno di MATLAB [19], in quanto anche in questo caso si é dovuti creare un file mex in modo da poter applicare le funzionalitá incluse nello Stereolabs Software Development Kit, descritto nella sezione 2.2.2. Infine, lo stesso metodo é stato applicato per abilitare le funzioni di *aruconano* nell'ambiente utilizzato.

3

PROCEDURA DELLE PROVE SPERIMENTALI

In questo capitolo si discute la procedura seguita per eseguire l'acquisizione di dati durante le prove sperimentali, partendo da una descrizione generale del processo, affrontando poi singolarmente le fasi che lo compongono. Si ricorda che l'obiettivo degli esperimenti condotti é quello di acquisire informazioni riguardo la posa dei medesimi soggetti tramite due sistemi di Motion Capture, funzionanti in contemporanea: un sistema OptiTrack con tecnologia optical-passive ed il sistema sviluppato basato su marker ArUco, in modo da confrontarli e studiarne le prestazioni.

In Figura 14 viene mostrato quella che é stata la procedura seguita per condurre le prove sperimentali ed analizzarne i risultati. In particolare il lavoro é stato suddiviso in quattro fasi:

- Fase di *Set up*: fase iniziale che consiste nel preparare il set up per gli esperimenti, in modo da garantirne il corretto funzionamento.
- Fase di *Detection*: costituisce la parte centrale del processo, durante la quale vengono acquisite le informazioni che saranno successivamente elaborate.
- Fase di *Output*: durante questa fase i dati raccolti in precedenza vengono riportati in un formato standard, in modo da semplificarne l'esportazione e lo stoccaggio.
- Fase di *Post-Processing*: prevede i calcoli eseguiti per combinare la posa dei singoli marker rilevati, in modo da poter confrontare le prestazioni dei due sistemi di visioni utilizzati.

3.1 FASE DI SET UP

Questa fase comprende tutte le operazioni iniziali e fondamentali per predisporre il sistema a quelle successive. In particolare, inizialmente é necessario calibrare entrambi i sistemi di visione, OptiTrack e videocamera, in modo da ricavarne i parametri ed assicurarsi misure accurate e coerenti tra loro. Successivamente vengono stabiliti i dati della prova sperimentale, in termini di durata e numero di marker utilizzati.

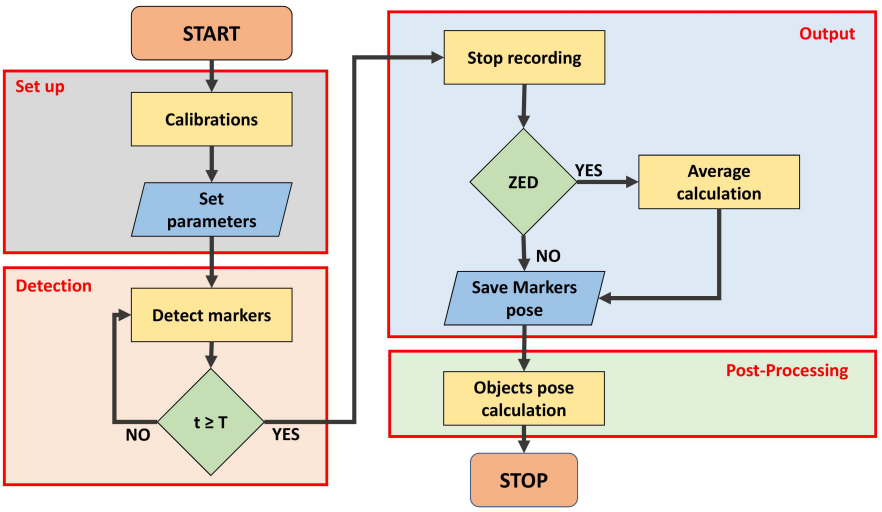


Figura 14: Diagramma di flusso seguito nella conduzione delle prove sperimentali realizzate, sia statiche che dinamiche

3.1.1 *Calibrazione dei sistemi*

Per calibrazione di una videocamera, si intende il processo di determinazione delle caratteristiche geometriche e ottiche interne della fotocamera (parametri intrinseci o *intrinsic parameters*) e della posizione e orientamento 3D del sistema di riferimento della fotocamera rispetto a un certo sistema di coordinate $\mathscr{F}_W$ (parametri estrinseci o *extrinsic parameters*) [17], il cui ruolo viene meglio descritto in Figura 15.

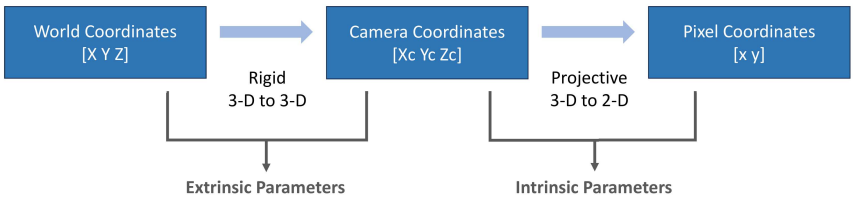


Figura 15: Ruolo dei parametri intrinseci ed estrinseci nel processo di rilevamento della posa di un punto [22]

In particolare, dato un punto di coordinate $[X, Y, Z]$ nel sistema di riferimento globale, tramite i parametri estrinseci della telecamera é possibile effettuare un cambio di coordinate, esprimendo il punto nel frame centrato sulla telecamera stessa [58]. Per fare questo si segue una procedura uguale a quella approfondita alla sezione 1.3, dove partendo dalla conoscenza della posa del generico punto e della telecamera in $\mathscr{F}_W$, si puó esprimere il tutto nel sistema $\mathscr{F}_L$, che in

questo caso corrisponde a quello posizionato sulla telecamera stessa, ottenendo:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t} \tag{11}$$

dove **R** é la matrice di rotazione che esprime l'orientamento della videocamera rispetto il world frame e **t** ne esprime la posizione. Successivamente, le coordinate in tre dimensioni che identificano la posizione del punto vengono mappate sul piano bidimensionale, tramite delle coordinate in pixel $[x, y]$, sfruttando i parametri intrinseci composti da:

- **Lunghezza focale** f: indica la distanza tra il piano dell'immagine ed il centro dell'obiettivo, tipicamente in millimetri, misurata sugli assi orizzontale (f_x) e verticale (f_y).
- **Coordinate del punto principale** (c_x, c_y): utilizzati come offset del centro dell'immagine, indicano la posizione sul piano in cui avviene l'intersezione con l'asse ottico della telecamera.
- **Coefficiente di inclinazione** s: indica la presenza di un'inclinazione tra l'obiettivo della telecamera ed il piano dell'immagine.

Infine, occorre considerare delle eventuali distorsioni, radiale o tangenziale, dell'immagine provocate dalla lente della telecamera, che possono essere compensate tramite equazioni di secondo grado, ovvero utilizzando due coefficienti ognuna:

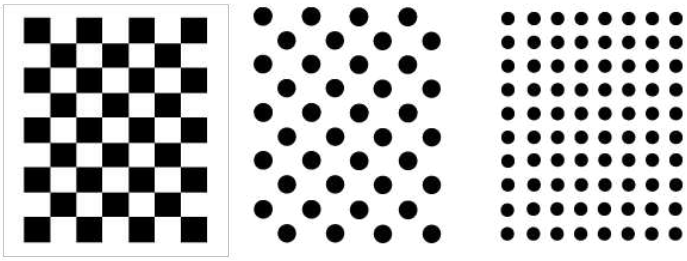
- (k_1, k_2): coefficienti di distorsione radiale della lente.
- (p_1, p_2): coefficienti di distorsione tangenziale della lente.

Per semplificarne la notazione, in questo lavoro é stata applicata la forma matriciale adottata da OpenCV [56], che prevede una matrice per i parametri intrinseci ed un vettore che contiene i coefficienti di distorsione:

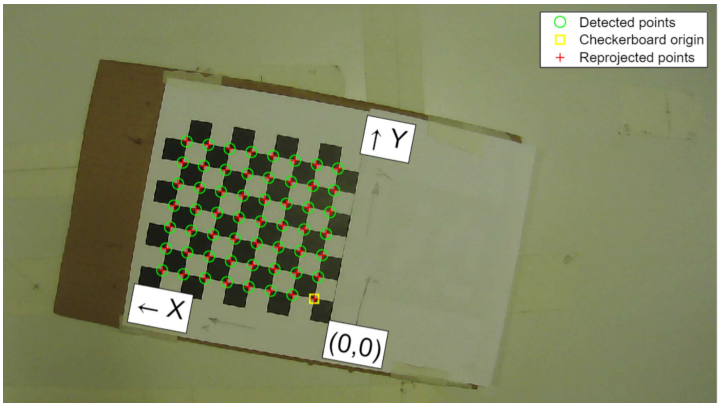
$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad d = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 \end{bmatrix} \tag{12}$$

dove s é stato ipotizzato nullo.

Di seguito vengono riportate le procedure ed i risultati delle calibrazioni dei sistemi utilizzati nelle prove sperimentali.



(a)



(b)

Figura 16: Matlab Camera Calibrator: (a) possibili pattern per la calibrazione [32] (b) punti rilevati dall'analisi delle immagini

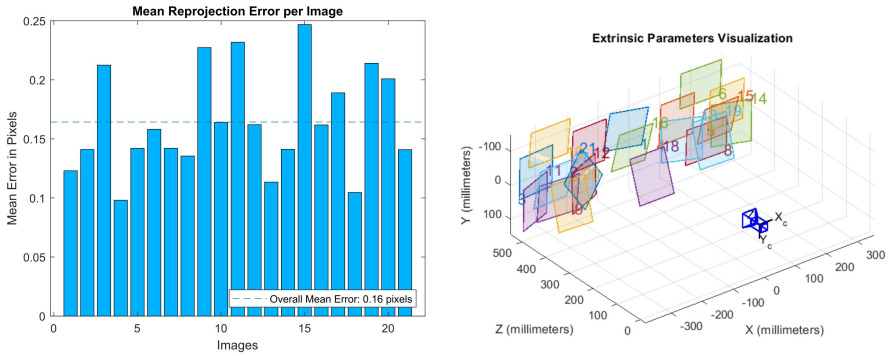
Calibrazione videocamera ELP

Per la calibrazione della videocamera digitale ELP, é stata utilizzata l'applicazione *Camera Calibrator*, all'interno di Matlab, la quale permette di ricavare i parametri geometrici di un singolo sensore a partire da un gruppo di immagini di un pattern scelto tra dei modelli predefiniti, Figura 16a, scattate dalla telecamera che si vuole calibrare [21]. Successivamente, l'applicazione analizza le foto e fornisce le stesse sovrapponendovi i punti rilevati, come mostrato in Figura 16b.

Dall'analisi delle immagini si ricavano i risultati della calibrazione, ovvero l'errore medio dei riproduzione dei punti di riferimento presenti nel pattern selezionato, Figura 17a, e la visualizzazione dei parametri estrinseci, i quali consistono nella la posizione della videocamera rispetto alla figura di riferimento per ogni immagine scattata, Figura 17b. A partire da questi dati, vengono poi estrapolati i parametri intrinseci del sensore, in modo da creare la matrice ed il vettore definiti nella (12).

Calibrazione stereo camera ZED

A differenza del primo, il sensore ZED utilizzato in questo progetto é in grado di implementare una auto calibrazione, calcolando all'accensione tutti i parametri necessari al suo impiego, forniti tramite la



(a) (b)

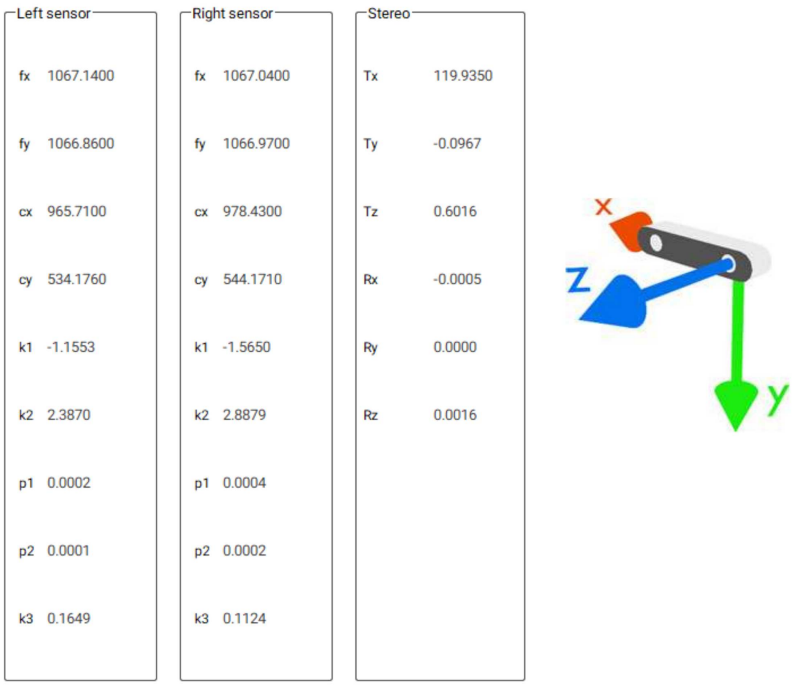
Figura 17: Risultati forniti dall'applicazione Matlab Camera Calibrator: (a) errore medio in pixel sulla riproduzione dei punti del pattern (b) visualizzazione dei parametri estrinseci

schermata mostrata in Figura 18a. Si riconoscono in particolare, per ogni sensore, i parametri relativi alla lunghezza focale, alle coordinate del punto principale ed i coefficienti di distorsione, ai quali ne viene aggiunto uno nel caso in cui si necessiti di una approssimazione del terzo grado.

Nella sezione *Stereo* vengono invece forniti i dati geometrici su posizione ed orientamento della telecamera destra rispetto al frame fissato in corrispondenza del sensore sinistro, orientato come mostrato in Figura 18b, seguendo lo standard consigliato da OpenCV. Tali misure, che comprendono un vettore per la traslazione ed un vettore per gli angoli, sono essenziali per poter permettere alla videocamera di restituire un solo dato riguardo la posa di un oggetto, eseguendo un cambio di coordinate tra le uscite ottenute dai due sensori in modo da renderle confrontabili.

Calibrazione OptiTrack

L'ultimo sistema che necessita un processo di calibrazione é il sistema di visione OptiTrack, il quale si serve di due specifici strumenti forniti dall'azienda stessa, Figura 19, realizzati appositamente per posizionare con precisione tre marker retro riflettenti ad una distanza nota. Nel dettaglio, il processo di calibrazione prevede inizialmente di orientare tutte le videocamere nella direzione dell'area di lavoro aggiustando la luminositá e l'illuminazione del workspace in modo da non rilevare oggetti indesiderati, quali materiali metallici o simili, che potrebbero disturbare il sistema. Successivamente si procede al calcolo della posizione relativa tra le videocamere andando a muovere il primo strumento all'interno dell'area di lavoro e rilevando i marker fissati su di esso ed, infine, utilizzando il secondo strumento si imposta il piano di riferimento.



(a) (b)

Figura 18: Stereo camera ZED 2i: (a) Parametri forniti dal processo di auto calibrazione (b) sistema di riferimento standard [50]

3.1.2 *Impostazione dei parametri di simulazione*

In questo passaggio, una volta a disposizione i dati dei sensori, vengono impostati diversi parametri necessari al rilevamento dei marker, tra cui la loro dimensione in centimetri. Inoltre, come specificato tra gli obiettivi del lavoro, si é testata la possibilitá di utilizzare piú marker per calcolare un'unica posa. Per fare questo é necessario inserire in questa fase il numero di marker da raggruppare ed il loro identificativo, i quali avranno un ruolo cruciale nel calcolo della posa. Infine si é impostata la durata della prova ed i parametri per la comunicazione tra i dispositivi connessi ai due sistemi di Motion capture, OptiTrack e quello basato sui marker ArUco, in modo da garantirne la contemporaneitá alla partenza ed il sincronismo durante la fase successiva.

3.2 FASE DI DETECTION

Una volta definiti tutti i parametri della prova, si procede alla rivelazione dei marker da parte dei due sistemi in contemporanea, i quali tuttavia sono progettati per rilevarne tipologie differenti: il sistema di

(a) (b)

Figura 19: Strumenti per la calibrazione del sistema di visione OptiTrack [41]

visione OptiTrack rileva gruppi di marker retro riflettenti a forma sferica, mentre il sistema sviluppato risulta programmato per rilevare la posa di marker ArUco, tramite le apposite funzioni. Per ovviare a tale questione, é stato creato un sostegno tramite stampaggio 3D, sul quale posizionare entrambe le tipologia di marcatori con una posizione nota, secondo le misure riportate in figura 20. Il principale scopo del corpo rigido creato é quindi quello di eseguire lo stesso movimento dei marker rilevati dai due sistemi, in modo da poterne confrontare le prestazioni a paritá di posizione ed orientamento, tramite la creazione di un elemento detto *object*, la cui posa viene calcolata nella fase di post-processing in due modi differenti, a partire dai dati dei diversi marker per utilizzarne il maggior numero di informazioni.

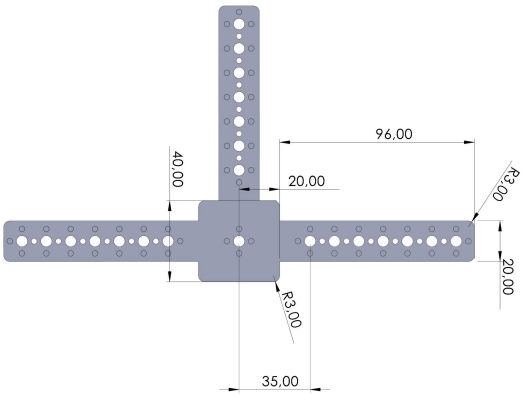


Figura 20: Corpo rigido realizzato per posizionare i marker di entrambi i sistemi di visione utilizzati

Nelle prove sperimentali eseguite sono stati impiegati due elementi object, in quanto i sistemi di Motion Capture analizzati utilizzano dei riferimenti differenti, non fornendo quindi misure confrontabili. Uno degli oggetti é stato quindi denominato *Reference Object*, la cui posa costituisce il frame di riferimento per tutte le misure effettuate, mentre il secondo é stato definito *Mobile Object* ed é l'elemento del quale vengono modificate posizione e orientamento nel corso delle prove sperimentali.

3.2.1 *Rilevamento dei marker*

Ad ogni ciclo, quindi, viene rilevata la posa di ogni singolo marker, sia ArUco che OptiTrack, e salvata temporaneamente in apposite strutture dati. In particolare, per ognuno dei marcatori ArUco, vengono rilevati:

- **Posizione**: viene fornita tramite un vettore in $\mathbb{R}^3$ che identifica la posizione del centro del marker rispetto al frame centrato sulla videocamera.
- **Orientamento**: anch'esso fornito tramite un vettore in $\mathbb{R}^3$, in quanto utilizza la rappresentazione geometrica, descritta alla sezione 1.3.1, e riferita al frame della videocamera.
- **Identificativo**: numero intero che é univocamente associato al codice contenuto all'interno del marker.

Si noti che il tempo impiegato nella rivelazione e stoccaggio dei dati da parte dei due sistemi é differente, permettendo al sistema OptiTrack di accumulare dati molto accurati ed in numero molto maggiore a paritá di tempo di lavoro, in quanto utilizza componenti ottimizzati per l'esecuzione di tale funzione. In particolare, il tempo medio presente tra due acquisizioni da parte del sistema OptiTrack é di 8.3ms, mentre per il sistema basato su ArUco risulta non minore di 40ms. Per questo motivo, d'ora in avanti il sistema OptiTrack sará considerato con un approccio "*black box*", ovvero senza addentrarsi nelle operazioni da esso effettuate ma limitandosi ai dati forniti, ipotizzandoli come misura esatta, in quanto lo scopo principale di questo lavoro é quello di sviluppare e validare un sistema di Motion Capture basato su marker ArUco, che verrá invece discusso approfonditamente in ogni suo aspetto.

3.2.2 *Verifica della durata*

Al termine di ogni iterazione, l'algoritmo verifica che il tempo trascorso dall'inizio della fase di detection, t, sia minore o uguale rispetto alla durata della prova stabilita nella fase precedete, T. Se ció si verifica, allora si prosegue con una ulteriore iterazione in cui vengono raccolti nuovi dati relativi alla posa dei marker, mentre nel caso contrario, si procede alla fase successiva, in cui entrambi i sistemi vengono bloccati al medesimo istante.

3.3 FASE DI OUTUP

Dopo aver terminato la raccolta dei dati da parte dei due sistemi, si procede alla creazione degli output che saranno successivamente elaborati ed analizzati.

3.3.1 *Strutture dati in uscita*

Il sistema fornisce in uscita due matrici, una per la posizione ed una per l'orientamento, in cui ogni colonna contiene un'informazione relativa ad un singolo marker per ogni istante di acquisizione, ed un vettore, il quale contiene informazioni sul tempo trascorso rispetto all'inizio della prova. Tale vettore é fondamentale per poter paragonare i due sistemi offline, in quanto, come detto precedentemente, funzionano a velocitá diverse. Le matrici sono inizialmente vuote e vengono aumentate di una riga ad ogni iterazione, ciascuna delle quali é proporzionale al numero di marker da rilevare nell'immagine.

Si consideri, ad esempio, il caso in cui occorra rilevare la posa di n marker con identificativo da 0 a $n-1$. Ciascuno di essi, ad ogni istante, sará caratterizzato da una posizione, T, ed un orientamento, R, entrambi rappresentati tramite un vettore di dimensione 3×1 e riferiti al frame locale, situato sulla videocamera, $\mathscr{F}_C = \{O_C, (\mathbf{x}_C, \mathbf{y}_C, \mathbf{z}_C)\}$. Si avranno quindi, dopo ogni acquisizione, n vettori che descrivono la posizione dei marker:

$$T_0 = \begin{bmatrix} T_{x,0} \\ T_{y,0} \\ T_{z,0} \end{bmatrix}, \quad T_1 = \begin{bmatrix} T_{x,1} \\ T_{y,1} \\ T_{z,1} \end{bmatrix}, \quad \dots \quad T_{n-1} = \begin{bmatrix} T_{x,n-1} \\ T_{y,n-1} \\ T_{z,n-1} \end{bmatrix} \tag{13}$$

ed altrettanti per descriverne l'orientamento in forma geometrica:

$$R_0 = \begin{bmatrix} R_{x,0} \\ R_{y,0} \\ R_{z,0} \end{bmatrix}, \quad R_1 = \begin{bmatrix} R_{x,1} \\ R_{y,1} \\ R_{1,0} \end{bmatrix}, \quad \dots \quad R_{n-1} = \begin{bmatrix} R_{x,n-1} \\ R_{y,n-1} \\ R_{z,n-1} \end{bmatrix} \tag{14}$$

La matrice che raccoglie le informazioni relative alla posizione di tutti i marker, successivamente alla prima acquisizione, viene aumentata di una riga contenente le informazioni presenti nella (13), riordinati all'interno di un singolo vettore:

$$\begin{bmatrix} T_{x,0} & T_{y,0} & T_{z,0} & \dots & T_{x,n-1} & T_{y,n-1} & T_{z,n-1} \end{bmatrix} \tag{15}$$

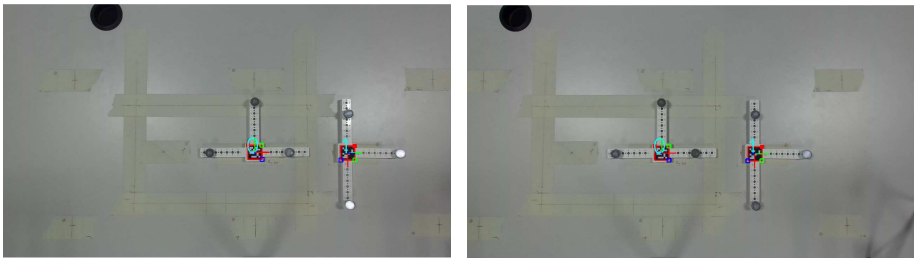
il quale contiene sulle colonne dalla 1 alla 3 la posizione lungo gli assi $\mathbf{x}_C$, $\mathbf{y}_C$ e $\mathbf{z}_C$ del marker 0, sulle colonne dalla 4 alla 6 la posizione del marker 1, e cosí via. Il procedimento risulta analogo per la matrice contenente le informazioni che riguardano l'orientamento dei marcatori, alla quale, ad ogni iterazione, viene aggiunto il vettore:

$$\begin{bmatrix} R_{x,0} & R_{y,0} & R_{z,0} & \dots & R_{x,n-1} & R_{y,n-1} & R_{z,n-1} \end{bmatrix} \tag{16}$$

Si deduce quindi che, dopo t istanti di acquisizione, la matrice delle posizioni dei marker, cosí come quella degli orientamenti, risulta essere di dimensione t righe per $3n$ colonne.

3.3.2 *Verifica sull'utilizzo della stereo camera ZED*

Nell'algoritmo occorre suddividere il caso in cui sia stata impiegato il sensore ZED oppure la videocamera ELP. Questo passaggio risulta necessario in quanto utilizzando una videocamera con visione stereo, a differenza di una singola, si hanno a disposizione due immagini della stessa area di lavoro, acquisite da due prospettive differenti, come mostrato in Figura 21, ottenendo quindi in uscita il doppio delle informazioni. Per riportare i dati di output in un formato standard, in modo da facilitare la successiva fase di elaborazione, é quindi necessario effettuare una operazione aggiuntiva di media.



(a) (b)

Figura 21: Funzionamento stereo camera ZED: (a) immagine acquisita dal sensore sinistro (b) immagine acquisita dal sensore destro

3.3.3 *Media dei dati*

Come é stato appena descritto, utilizzando una sensore con visione stereo, si hanno a disposizione due pose per ogni singolo marker, tuttavia, tali misure non possono essere confrontate, in quanto fanno riferimento ciascuna al frame solidale alla videocamera che ha acquisito l'immagine analizzata. Per ottenere una singola misura si effettua quindi un cambio di coordinate delle pose rilevate tramite il sensore destro, le quali fanno quindi riferimento al frame $\mathscr{F}_D = \{O_D, (\mathbf{x}_D, \mathbf{y}_D, \mathbf{z}_D)\}$, riportando tutto sul frame solidale alla videocamera sinistra, $\mathscr{F}_S = \{O_S, (\mathbf{x}_S, \mathbf{y}_S, \mathbf{z}_S)\}$, come definito precedentemente in Figura 18b. Le informazioni necessarie per effettuare tale operazione, ovvero posizione ed orientamento di $\mathscr{F}_D$ rispetto a $\mathscr{F}_S$, vengono fornite tramite il processo di auto calibrazione descritto alla sezione 3.1.1.

Dato quindi un marker rilevato tramite stereo camera, possiamo definire:

- $\mathbf{p}_S, \boldsymbol{\delta}_S$: posizione ed orientamento in angoli di Eulero del marker rilevato tramite sensore sinistro, che fa quindi riferimento al frame $\mathscr{F}_S$.

- $\mathbf{p}_D, \delta_D$: posizione ed orientamento in angoli di Eulero del marker rilevato tramite sensore destro, ovvero rispetto al frame $\mathscr{F}_D$.
- $\mathbf{T} = (T_x, T_y, T_z)$: posizione del frame $\mathscr{F}_D$ rispetto il frame $\mathscr{F}_S$, in millimetri.
- $\mathbf{R} = (R_x, R_y, R_z)$: orientamento del frame $\mathscr{F}_D$ rispetto il frame $\mathscr{F}_S$, espresso in forma geometrica.

Come si nota dalla Figura 18a, i due frame non presentano differenze apprezzabili per quanto riguarda l'orientamento, e per questo motivo si é posto $\mathbf{R} = 0_{3\times1}$, permettendo di calcolare l'orientamento del marker solamente tramite un'operazione di media aritmetica tra gli angoli rilevati da entrambi i sensori. Per quanto riguarda la posizione invece, si definisce il vettore:

$$\mathbf{p}'_D = \mathbf{p}_D + \mathbf{T} \tag{17}$$

che esprime la posizione del marker rilevata dal sensore destro riportata al frame di sinistra. A questo punto, entrambe le misure a disposizione sono legate allo stesso sistema di riferimento, permettendo di calcolarne la media. Si noti che l'unico elemento del vettore $\mathbf{T}$ che assume un valore significativamente diverso da zero é T_x, dovendo effettuare quindi solamente una traslazione orizzontale delle coordinate.

Si ottiene cosí, per ogni marker, una coppia di vettori in $\mathbb{R}^3$ che esprimono completamente la posa del marker stesso, i quali possono essere gestiti come descritto in 3.3.1.

3.4 FASE DI POST-PROCESSING

Nella fase finale dell'algoritmo, vengono elaborati i dati raccolti dai due sistemi di visione differenti, in modo da poterli confrontare istante per istante.

3.4.1 *Creazione degli Objects ArUco*

Come accennato in precedenza, si é reso necessario fissare un sistema di riferimento comune per entrambi i sistemi di Motion capture tramite un Reference Object, la cui posa costituirá quindi il frame di riferimento $\mathscr{F}_R = \{O_R, (\mathbf{x}_R, \mathbf{y}_R, \mathbf{z}_R)\}$, per le misure della posa del frame solidale al Mobile Object, $\mathscr{F}_M = \{O_M, (\mathbf{x}_M, \mathbf{y}_M, \mathbf{z}_M)\}$

Per quanto riguarda il sistema di visione OtiTrack, la posizione e l'orientamento del corpo rigido vengono ricavate a partire da un gruppo composto da tre sfere retro riflettenti con posizione fissa, creando un rigid body all'interno del software Motive ed estrapolando la posa del punto centrale.

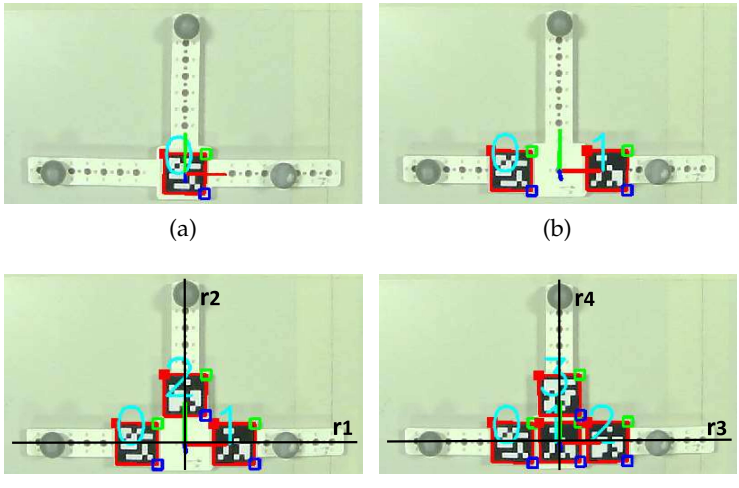
Considerando ora il sistema basato su ArUco, sono stati utilizzati un numero crescente di marker, da uno a quattro, in modo da studiare l'effetto dell'aumento di marcatori impiegati sulla diminuzione dell'errore di stima della posa del corpo rigido, andando a disporli in maniera predefinita. Partendo quindi dalla posa dei singoli marker, rilevati e messi a disposizione dalle fasi precedenti, si é dovuta ricavare la posa del singolo object applicando i procedimenti che seguono, con l'obiettivo di sfruttare il maggior numero di informazioni all'interno del procedimento sviluppato.

Object composto da 1 marker

Utilizzando un singolo marcatore ArUco, come in Figura 22a, la posa dell'elemento object corrisponde con la posa del marker stesso, ovvero la posizione e l'orientamento del corpo rigido equivalgono alle corrispettive identificate tramite l'algoritmo proprio di ArUco nella fase precedente.

Object composto da 2 marker

Quando un object é costituito da una coppia di marker, la sua posizione viene calcolata come media delle posizioni dei due marcatori lungo gli assi x, y e z, Figura 22b. In modo analogo, si é calcolato l'orientamento del corpo rigido come media degli angoli di Eulero associati ai singoli marker, corrispondenti alle rotazioni elementari attorni ai tre assi.



(a) (b)

(c) (d)

Figura 22: Disposizione dei marker all'interno del corpo rigido composto da: (a) 1 marker (b) 2 marker (c) 3 marker (d) 4 marker

Object composto da 3 marker

Nel caso in cui tre marker compongano l'object di cui si vuole rilevare la posa, si é proceduto, per quanto riguarda la posizione, calcolando l'intersezione tra le due rette r_1 ed r_2 mostrate in Figura 22c, dove:

- r_1 é la retta passante per il centro dei marker con identificativo 0 ed 1.
- r_2 é la retta passante per il centro del marker numero 2 e perpendicolare a r_1.

Per quanto riguarda l'orientamento, come prima, viene calcolato dalla media degli angoli legati all'orientamento dei singoli marker rilevati.

Object composto da 4 marker

Infine, se il corpo rigido é costituito da quattro marker, la posizione dell'object viene calcolata in modo simile al caso precedente, ovvero coincidente con l'intersezione tra le rette r_3 ed r_4 mostrate in Figura 22d, dove:

- r_3 é la retta passante per il centro dei marker 0 e 2.
- r_4 é la retta passante per il centro del marker 1 e 3.

L'orientamento, anche in questo caso é pari alla media degli orientamenti dei marker che lo compongono.

3.4.2 *Confronto dei sistemi*

Come passaggio finale, al fine di calcolare precisione ed accuratezza del sistema sviluppato, sono state confrontate le misure effettuate, considerando come esatti i dati provenienti dal sistema OptiTrack, in quanto aventi accuratezza nota dalla letteratura dell'ordine del decimo di millimetro [7]. Data la diversa velocitá di acquisizione, tuttavia, la quantitá di dati raccolta dal sistema OptiTrack rispetto a quello sviluppato risulta essere notevolmente maggiore, rendendo necessario una interpolazione delle misure basate sui marker ArUco in modo da confrontarle in ogni istante.

In particolare, si é analizzata la posizione lungo tutti e tre gli assi del Mobile object, rilevata da entrambi i sistemi, rispetto a $\mathscr{F}_R$, ed i singoli angoli delle rotazioni elementari presenti tra i due sistemi di riferimento.

4

DESCRIZIONE DEL SOFTWARE

Analogamente alla procedura esposta nel capitolo precedente, si é suddiviso la descrizione del software sviluppato nelle medesime quattro fasi, in modo da semplificarne la spiegazione. Si noti che esso presenta delle differenze in base al tipo di videocamera utilizzata, nella fase iniziale e durante il rilevamento dei marker, a causa delle diverse tecnologie e modalitá di funzionamento. Per quanto riguarda l'elaborazione dei dati, invece, il software é il medesimo in quanto si é standardizzato il formato delle informazioni in uscita dalle fasi di detection.

4.1 FASE DI SET UP

Con riferimento al codice riportato nelle appendici A e B, si nota che il primo passaggio, indipendentemente dal sensore impiegato, é quello di inserire i dati relativi alla prova sperimentale, quali la grandezza fisica dei marker a disposizione, necessaria per la rivelazione degli stessi nella fase successiva, e la matrice `IdObj`, contenente sulla prima riga, gli identificativi dei marker che compongono il Reference object, mentre sulla seconda quelli che costituiscono il Mobile object. Tale elemento risulta utile per definire sia il numero di marker utilizzati, che per permettere il salvataggio e l'analisi dei dati. Infine, vengono create le strutture dove saranno salvati i dati, inizialmente vuote, e la variabile globale `im1`, che servirá per mostrare l'immagine elaborata ad ogni istante[1] tramite la funzione `ShowFrameFcn`, richiamata ad intervalli costanti dal timer `FrameTimer`, e basata principalmente sula funzione `imshow`.

Per effettuare il caricamento dei parametri delle videocamere, occorre invece differenziare due procedimenti differenti. La videocamera ELP, il cui codice é contenuto nell'appendice A, permette di ricavare facilmente le informazioni necessarie andando semplicemente ad utilizzare `camMatrix` e `distCoeff`, che rappresentano rispettivamente la matrice composta dai parametri intrinseci della videocamera e il vettore contenente i coefficienti di distorsione, come definiti nella (12), e vengono ricavati tramite un secondo script Matlab, il quale, a partire dai risultati dell'applicazione "Camera Calibrator" di Matlab, crea tali variabili.

Al contrario, per poter utilizzare la stereo camera ZED, occorre effettuare una inizializzazione piú complessa, come mostrato nell'ap-

1 Si noti che, nel caso di utilizzo della stereo camera ZED, sono necessarie due variabili globali in quanto ad ogni istante si ha una coppia di immagini acquisite

pendice B. In particolare i passaggi da eseguire sono:

- Configurare i parametri utili ad impostare la modalitá di funzionamento della stereo camera, quali risoluzione, frequenza e area di lavoro, assegnati secondo le guide fornite [49], tramite `InitParameters`.
- Apertura del file `mexZED`, il quale ha lo scopo di integrare le funzionalitá ZED, sviluppate in linguaggio C++, all'interno dell'ambiente Matlab [51] utilizzando l'estensione *mex*.
- Se l'apertura del file ha esito positivo, si ricavano i parametri intrinseci di entrambi i sensori della stereo camera, creando quindi le rispettive matrici, `camMatrix`, e vettori di distorsione, `distCoeff`, per ogni videocamera.

É presente infine una sezione dedicata all'input della durata della prova sperimentale, uguale per le prove sperimentali che utilizzano sensori differenti, dove si fornisce la possibilitá di definire il tempo di registrazione dei due dispositivi, in base alle esigenze richieste dalla tipologia di esperimento in corso, nel formato *ore : minuti : secondi*, che viene successivamente convertito in secondi per semplicitá di utilizzo e salvata nella variabile `rec_end`. Nel caso in cui l'input sia nullo, si é definita una durata di default, in modo da avere sempre una prova sperimentale valida, pari ad un minuto.

4.2 FASE DI DETECTION

Il passaggio iniziale di questa fase é quello di far partire il timer dedicato alla gestione delle immagini ed il conteggio della durata della prova sperimentale, tramite la variabile `rec_time`, che verrá verificata al termine di ogni ciclo. A tale scopo, si é utilizzata la funzione `tic-toc`, che permette di calcolare il tempo in secondi trascorso tra i due comandi. In questo caso l'istante di partenza per il conteggio corrisponde con l'esecuzione del comando `start_rec=tic`, mentre al termine di ogni ciclo di rilevamento é presente il comando `rec_time=toc`, che quindi salva nella variabile il tempo trascorso dall'istante salvato in `start_rec`.

Successivamente si procede all'avvio vero e proprio del rilevamento dei marker, da parte dei due sistemi di visione, con delle leggere differenze. Si analizza quindi inizialmente il processo di rilevamento nel caso di utilizzo della videocamera singola, e successivamente verranno approfondite le differenze tra i due software sviluppati illustrando il codice implementato per l'utilizzo della stereo camera.

4.2.1 *Rilevamento con videocamera singola*

Il software sviluppato per il rilevamento dei marker utilizzando una singola videocamera, si veda Appendice A, inizia con l'avviamento del sistema OptiTrack, tramite la apposita rete `NatNet` creata per sincronizzare i dispositivi che utilizzano i due sistemi di visione da confrontare [42].

Si entra successivamente all'interno del ciclo `while`, il quale viene ripetuto fino al momento in cui non risulta piú verificata la condizione `rec_time` $<$ `rec_time_end`, ovvero finché la durata della prova non supera il tempo stabilito nella fase iniziale.

All'interno del loop si trova il processo di rilevamento dei marker ArUco, che inizia con l'acquisizione di un'immagine, `frame`, tramite la videocamera utilizzata, a cui segue la chiamata alla funzione `aruco_nano_detector` tramite il comando:

```
% MARKER DETECTION
[frame, T, R, IdDetected] =
aruco_nano_detector(frame, camMatrix, distCoeff, markerSize)
```

da cui si nota che la funzione riceve in ingresso l'immagine acquisita, i parametri della videocamera ed infine la misura dei marker utilizzati. A partire da queste informazioni, eseguendo le azioni descritte alla sezione 2.3.2, vengono rilevati e restituiti:

- `frame`: alla variabile inizialmente contenente l'immagine scattata, viene sovrascritta la medesima foto modificata in modo da evidenziare il contorno ed il sistema di riferimento dei marker rilevati.
- `T`: vettore di dimensione $3 \times n$, che contiene sulle righe rispettivamente le coordinate x, y e z degli n marker rilevati.
- `R`: vettore della stessa dimensione del precedente, che contiene le informazioni relative all'orientamento degli n marker, espresse in forma geometrica.
- `IdDetected:` contiene l'identificativo dei marcatori rilevati nel frame analizzato.

Alla sezione D, si é riportato per intero il codice contenuto all'interno del file
`aruco_nano_detector.cpp`, il quale permette l'utilizzo delle funzioni contenute nelle librerie *aruconano* e *OpenCV*, creando un file di tipo *mex* utilizzando specifici vettori, detti *mxArray*, e puntatori, i quali permettono di effettuare la conversione dei dati tra i due linguaggi di programmazione.

Una volta nota la posa di tutti i marker, si procede raggruppando i dati in base all'oggetto di cui fanno parte. In particolare, la funzione

`GetMarkers`, riportata in D, riordina i dati raccolti mettendo sulla prime colonne i marker che costituiscono il Reference object, mentre a seguire quelli che costituiscono il mobile object. Si noti che, nel caso in cui un marker non venga rilevato, la posa dell'intero oggetto non risulta calcolabile seguendo la procedura descritta in 3.4.1, ponendo quindi i vettori di posizione ed orientamento ad un valore nullo definito tramite la variabile propria di Matlab `NaN`[2].

Infine, i dati completi vengono salvati ad ogni istante all'interno delle strutture apposite, seguendo le modalitá descritte in precedenza, in modo da ottenere in uscita le informazioni in un formato standardizzato. In Tabella 2 si é riportato un esempio della struttura della matrice contenente le posizioni di due marker, con identificativo 0 e 1, ottenuta dopo aver effettuato tre misurazioni. Sulla prima riga sono posizionate le coordinate dei marker dopo il primo istante di acquisizione, sulla seconda riga le coordinate misurate all'istante successivo, e cosí via fino al termine della prova. La matrice relativa all'orientamento risulta analoga, suddivisa quindi in righe per ogni ciclo e nelle colonne secondo il numero di marker impiegati. Si ricorda che tutte le misurazione sono riferite al sistema di riferimento posizionato sulla videocamera, ovvero rispetto a $\mathscr{F}_C = \{O_C, (\mathbf{x}_C, \mathbf{y}_C, \mathbf{z}_C)\}$.

	col 1	col 2	col 3	col 4	col 5	col 6
riga 1	T_{x0}	T_{y0}	T_{z0}	T_{x1}	T_{y1}	T_{z1}
riga 2	T'_{x0}	T'_{y0}	T'_{z0}	T'_{x1}	T'_{y1}	T'_{z1}
riga 3	T''_{x0}	T''_{y0}	T''_{z0}	T''_{x1}	T''_{y1}	T''_{z1}

Tabella 2: Esempio di struttura della variabile `vector_T_markers` a seguito di tre istanti di acquisizione della posa di una coppia di marker, con identificativo 0 ed 1

L'ultimo passaggio effettuato ad ogni esecuzione del ciclo, prevede il salvataggio all'interno del vettore `vector_time`, del tempo trascorso dall'esecuzione del comando `tic`, tramite la funzione `toc`. Tale operazione risulta fondamentale per poter confrontare la posa dei marker rilevata dai due sistemi di acquisizione nello stesso instante, una volta terminata la prova sperimentale.

4.2.2 *Rilevamento con stereo camera*

Analogamente al caso precedente, il software riportato in Appendice B inizia con l'avviamento del sistema di Motion Capture OptiTrack, tramite la rete appositamente creata. Successivamente il procedimento di acquisizione risulta sostanzialmente uguale a quello appena descritto, tuttavia ripetuto per due videocamere. Inizialmente vi é infatti l'acquisizione delle immagini tramite entrambi i sensori, sinistro e de-

2 La sigla `NaN` sta per "*Not a Number*"

stro, utilizzando il file `mexZED`, il quale permette all'ambiente Matlab di comunicare con lo strumento. I frame acquisiti, detti `frame_left` e `frame_right`, sono poi elaborati individualmente tramite la funzione `aruco_nano_detector`, ottenendo la posa di tutti i marker rilevati da entrambi i sensori, separando quindi posizione ed orientamento ricavate dal sensore sinistro e destro, in quanto riferiti a due sistemi distinti. Successivamente, le immagini vengono mostrate tramite le variabili globali richiamate dal timer in esecuzione ed, infine, i dati sono salvati all'interno di strutture dati uguali al caso precedente, ma in numero doppio, dovuto alla quantitá di immagini elaborate.

4.3 FASE DI OUTPUT

Come approfondito nell'algoritmo, é importante in questa sede distinguere due procedure differenti, in base al tipo di tecnologia utilizzata nell'acquisizione delle immagine per il rilevamento dei marker, in quanto, impiegando la stereo camera ZED si hanno a disposizione il doppio di informazioni per ogni marcatore. É quindi necessario effettuare un'operazione aggiuntiva, con l'obiettivo di riportare i dati raccolti in un formato standard, che possa essere facilmente elaborato ed analizzato nella fase finale.

Innanzitutto, si sottolinea che avendo origine da videocamere distinte, le pose dei marker sono rilevate a partire da due sistemi di riferimento diversi, posizionati in corrispondenza dei sensori sinistro e destro della stereo camera. Al fine di ottenere posizione ed orientamento unici, si é quindi eseguita una operazione di media tra le due misure, ma solamente dopo averle rese paragonabili sfruttando le informazioni ricavate dalla calibrazione descritta in 3.1.1. Le informazioni ottenute dal processo di rilevamento, con riferimento alle definizioni fornite in 3.3.3, sono quindi:

- `T_marker_Right`: posizione di un singolo marcatore rilevata tramite il sensore destro, espressa nel sistema di riferimento $\mathscr{F}_D$.
- `T_marker_Right_moved`: posizione di un singolo marcatore rilevata tramite il sensore destro, espressa nel sistema di riferimento $\mathscr{F}_S$.
- `T_marker_Left`: posizione di un singolo marcatore rilevata tramite sensore sinistro, espressa nel sistema di riferimento $\mathscr{F}_S$.
- `T_marker`: posizione di un singolo marcatore ricavata come media delle due misure, espressa nel sistema di riferimento $\mathscr{F}_S$.

le quali vengono impiegate nel codice come segue:

```
% MARKER POSITION CALCULATION
% Move Right Camera measure into Left Camera frame
T_marker_Right_moved = T_marker_Right + camInfo.t';
```

```
% Average of the two measured positions
T_marker = (T_marker_Left + T_marker_Right_moved)/2;
```

dove in `camInfo.t` é contenuta la posizione del sistema di riferimento destro, rispetto il frame sinistro, che permette di effettuare il cambio di coordinate richiesto. In questo caso, come si nota dalla Figura 18a, l'unica traslazione tra i due frame risulta circa pari a 12cm lungo l'asse x_S, mentre i restanti elementi possono essere considerati trascurabili.

Per quanto riguarda l'orientamento invece, si é reso necessario convertire i dati raccolti, espressi in forma geometrica, in angoli di Eulero, per poter effettuare l'operazione di media. In particolare, utilizzando una simbologia analoga al caso precedente, il codice realizzato risulta:

```
% MARKER ORIENTATION CALCULATION
% Rotation matrix
RotMat_marker_Left = Rodrigues(R_marker_Left);
RotMat_marker_Right = Rodrigues(R_marker_Right);
% Eulero angles
Angles_marker_Left = rotm2eul(RotMat_marker_Left,'ZYX');
Angles_marker_Right = rotm2eul(RotMat_marker_Right,'ZYX');
% Average of the two measured orientationa
Angles_marker = (Angles_marker_Left+Angles_marker_Right)/2;
RotMat_marker = eul2rotm(Angles_marker,'ZYX');
% Marker orientation 3x1
R_marker = Rodrigues(RotMat_marker);
```

dove la funzione `Rodrigues`, riportata nella Appendice D, permette di convertire una matrice di rotazione di dimensione 3×3 in un orientamento in forma geometrica, e viceversa, utilizzando la formula di Rodrigues, equazione (5). Una volta ricavati i valori degli angoli, tramite la funziona Matlab `rotm2eul`, essi vengono mediati e successivamente riconvertiti prima di forma matriciale, ed infine in forma geometrica, salvando l'orientamento finale all'interno di `R_marker`.

Al termine di tale processo, l'uscita finale sará quindi composta da due matrici, le quali descrivono completamente la posa di tutti i marker rilevati dalla stereo camera, a cui si aggiunge un vettore contenente le informazioni temporali della prova sperimentale, il tutto nello stesso formato impiegato nel caso di utilizzo di un singolo sensore, in modo da standardizzare la procedura di analisi ed elaborazione dei dati raccolti.

4.4 FASE DI POST-PROCESSISNG

Con riferimento alla procedura descritta in 3.4 ed al codice riportato alla Appendice C, vengono approfondite le operazioni eseguite sui dati raccolti, in modo da ricavare informazioni utili a confrontare i due sistemi di Motion Capture e paragonarne le prestazioni.

Il primo passaggio é quello di eseguire una iterazione del processo di estrazione dei dati e seguente analisi per ogni istante di acquisizione, impiegando un ciclo `for`, dove si procede inizialmente alla chiamata della funzione `GetObject`. Tale funzione, richiede in ingresso la posa dei marker che compongono l'elemento, e seguendo il procedimento descritto in 3.4.1, ne restituisce posizione ed orientamento. In particolare, vi é una distinzione iniziale sul numero di marker utilizzati, in quanto la modalitá di calcolo della posa del singolo oggetto. Date `T_markers` ed `R_markers` le matrici in ingresso, di dimensione $3 \times n$ con n pari al numero di marker impiegati, il calcolo dei vettori in $\mathbb{R}^3$, `T_object` e `R_object`, contenenti la posa dell'oggetto avviene come segue.

Object composto da 1 marker

In questo caso `T_markers` ed `R_markers` sono dei vettori che descrivono completamente la posa dell'oggetto, in quanto essa coincide esattamente col marker che lo costituisce, ottenendo quindi:

$$\texttt{T\_object} = \texttt{T\_markers}, \qquad \texttt{R\_object} = \texttt{R\_markers} \tag{18}$$

Object composto da 2 marker

Nel caso in cui un oggetto sia costituito da una coppia di marker, detti genericamente marker A e B, le matrici in ingresso sono:

$$\texttt{T\_markers} = \begin{bmatrix} T_{x,A} & T_{x,B} \\ T_{y,A} & T_{y,B} \\ T_{z,A} & T_{z,B} \end{bmatrix}, \quad \texttt{R\_markers} = \begin{bmatrix} R_{x,A} & R_{x,B} \\ R_{y,A} & R_{y,B} \\ R_{z,A} & R_{z,B} \end{bmatrix} \tag{19}$$

La posizione del punto centrale dell'oggetto, descritta dalle coordinate `T_object`$= [T_x, T_y, T_z]^T$, viene quindi calcolata come media delle coordinate di ciascun marker, su ognuno dei tre assi:

$$T_x = \frac{T_{x,A} + T_{x,B}}{2}, \quad T_y = \frac{T_{y,A} + T_{y,B}}{2}, \quad T_z = \frac{T_{z,A} + T_{z,B}}{2} \tag{20}$$

Per quanto riguarda l'orientamento invece, occorre innanzitutto convertire i dati in ingresso in angoli di Eulero, passando per la forma matriciale utilizzando la formula di Rodrigues, in quanto non é possibile calcolarne la media se non in questo formato. Si ottengono quindi gli angoli delle rotazioni elementari attorno agli assi x, y, z, detti rispettivamente `object_angles`$= [\phi, \theta, \psi]$, come:

$$\phi = \frac{\phi_A + \phi_B}{2}, \quad \theta = \frac{\theta_A + \theta_B}{2}, \quad \psi = \frac{\psi_A + \psi_B}{2} \tag{21}$$

con chiaro significato dei simboli. Da questo vettore, si ottiene facilmente `R_object`, riconvertendolo in matrice di rotazione e successivamente applicando nuovamente la formula di Rodrigues.

Object composto da 3 marker

Se un object é composto da tre marker, si puó ipotizzare di considerare il centro di ognuno di essi come un punto, detti A, B e C, la cui posizione é descritta dalle coordinate nel sistema di riferimento centrato sulla videocamera da cui sono stati rilevati. Con riferimento alla Figura 23a, dove $O = [x_O, y_O, z_O]$ é il punto che rappresenta la posizione del singolo object, le cui coordinate sono incognite ed é posizionato nel punto di intersezione tra le due rette perpendicolari tra loro. Per la retta r_1, vale la seguente equazione:

$$r_1 : \frac{x - x_A}{x_B - x_A} = \frac{y - y_A}{y_B - y_A} \tag{22}$$

da cui é possibile ricavarne l'equazione in forma esplicita:

$$r_1 : \quad y = m_1(x - x_A) + y_A, \qquad \text{con} \quad m_1 = \frac{y_B - y_A}{x_B - x_A} \tag{23}$$

dove m_1 é detto coefficiente angolare della retta e ne indica la pendenza. Andando ora ad imporre la perpendicolaritá di r_2 rispetto a r_1, ed il passaggio per il punto C, si ottiene

$$r_2 : \quad y = m_2(x - x_C) + y_C, \qquad \text{con} \quad m_2 = -\frac{1}{m_1} \tag{24}$$

Si procede ora uguagliando le due equazioni, in modo da ricavare le coordinate del punto di intersezione, ovvero O, ottenendo:

$$x_O = \left(x_A + \frac{x_C}{m_1^2} + \frac{y_C - y_A}{m_1}\right) \cdot \left(1 + \frac{1}{m_1^2}\right)^{-1} \tag{25}$$

$$y_O = m_1 (x_O - x_A) + y_A \tag{26}$$

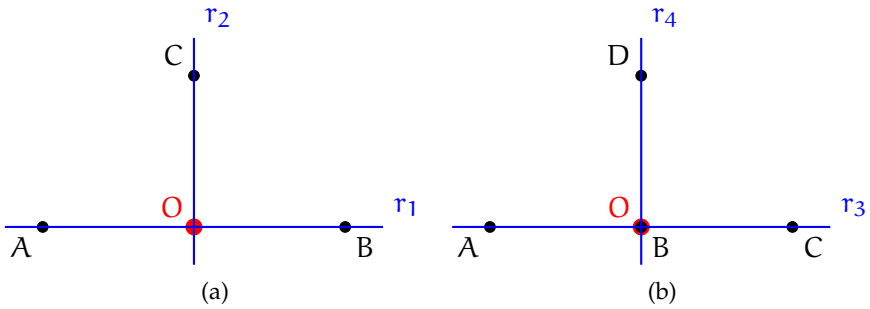
Per quanto riguarda la coordinata lungo l'asse z, data l'ipotesi che tutti i marker si trovino sullo stesso piano, viene calcolata come media delle loro posizioni:

$$z_O = \frac{z_A + z_B + z_C}{3} \tag{27}$$

da cui si ricava quindi:

$$\texttt{T\_object} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} x_O \\ y_O \\ z_O \end{bmatrix} \tag{28}$$

Infine, anche in questo caso, l'orientamento `R_object` viene calcolato come media degli angoli rilevati delle rotazioni elementari dei marker, attorno a ciascun asse.



(a) (b)

Figura 23: Metodo di calcolo della posizione degli elementi object impiegando (a) 3 marker o (b) quattro marker

Object composto da 4 marker

Analogamente al caso precedente, come si nota in Figura 23b, per calcolare il vettore `T_object`, si considera il punto centrale dei quattro marker rilevati, che compongono l'oggetto. Volendo quindi ricavare le coordinate del punto O, si scrivono le equazioni delle rette:

$$r_3: \quad y = m_3(x - x_A) + y_A, \qquad \text{con} \quad m_3 = \frac{y_C - y_A}{x_C - x_A} \tag{29}$$

$$r_4: \quad y = m_4(x - x_B) + y_B, \qquad \text{con} \quad m_4 = \frac{y_D - y_B}{x_D - x_B} \tag{30}$$

da cui si ottiene il punto di intersezione O, ovvero la posizione dell'oggetto:

$$T_x = x_O = \left(x_A + \frac{m_3}{m_4}x_B + \frac{y_B - y_A}{m_3}\right) \cdot \left(1 - \frac{m_4}{m_3}\right)^{-1} \tag{31}$$

$$T_y = y_O = m_3\,(x_O - x_A) + y_A \tag{32}$$

$$T_z = z_O = \frac{z_A + z_B + z_C + z_D}{4} \tag{33}$$

mentre l'orientamento viene calcolato, come prima, convertendo i dati in ingresso in forma angolare e successivamente eseguendo l'operazione di media.

Ad ogni istante di acquisizione, al termine dell'esecuzione della funzione `GetObject`, sono quindi noti:

- `Tref_C`: posizione del Reference Object nel sistema di riferimento centrato sula videocamera $\mathscr{F}_C$.
- `Tmob_C`: posizione del Mobile Object nel sistema di riferimento $\mathscr{F}_C$.
- `Rref_C`: orientamento del Reference Object rispetto a $\mathscr{F}_C$, in forma geometrica.

- `Tmob_C`: orientamento in forma geometrica del Mobile Object rispetto a $\mathscr{F}_C$.

dove per $\mathscr{F}_C$ si intende il frame centrato sulla videocamera, nel caso venga utilizzato un singolo sensore, oppure il sistema di riferimento posizionato sulla videocamera sinistra se la prova é stata condotta impiegando la stereo camera ZED.

L'obiettivo é ora quello di ricavare la posa del Mobile Object, $\mathscr{F}_M = \{O_M, (\mathbf{x}_M, \mathbf{y}_M, \mathbf{z}_M)\}$, rispetto al frame centrato sul Reference Object, $\mathscr{F}_R = \{O_R, (\mathbf{x}_R, \mathbf{y}_R, \mathbf{z}_R)\}$, in modo da avere una misura generale che possa essere confrontata con le rilevazioni eseguite dal sistema OptiTrack. Similmente al procedimento illustrato in 1.3.2, si ricavano inizialmente le matrici di rotazione che permettono di effettuare il cambio di coordinate tra sistema di riferimento della camera ed i frame centrati negli object, applicando la formula di Rodrigues:

```
% ROTATION MATRIX
% From Frame_R to Frame_C
RotMat_RC = Rodrigues(Rref_C);
% From Frame_M to Frame_C
RotMat_MC = Rodrigues(Rmob_C);
```

le quali, combinate, permettono di ottenere la posa finale ricercata, prima in forma matriciale e successivamente in angoli[3]:

```
% ORIENTATION
RotMat_rel = (RotMat_RC)' * RotMat_MC;
Angles_rel_R = rotm2eul(RotMat_rel,"ZYX") * 180/pi;
```

Per ottenere la posizione di O_R rispetto al sistema di riferimento fissato, si é inizialmente calcolata rispetto il frame centrato nella videocamera, e solo in seguito, si é applicato il cambio di coordinate tramite la matrice calcolata in precedenza:

```
% POSITION
% Relative position in camera system, FrameC
Trel_C = Tmob_C - Tref_C;
% Relative position in Reference object system, OR
Trel_R = (RotMat_RC)' * Trel_C;
```

Al termine di tale processo, si hanno quindi a disposizione orientamento e posizione di un object rispetto all'altro, rilevati sia dal sistema sviluppato e basato su ArUco, che dal sistema OptiTrack, il quale, come anticipato in precedenza, fornisce dati molto accurati permettendo di trascurarne un eventuale errore.

3 Si sottolinea che gli angoli sono espressi in gradi

5 ESPERIMENTI E RISULTATI

Ne presente capitolo verranno discusse le prove sperimentali eseguite al fine di ottenere informazioni utili a paragonare le prestazioni del sistema di Motion Capture implementato con il sistema OptiTrack disponile all'interno del laboratorio. Si ricorda che, dato il funzionamento ottimo del secondo, le misure da esso fornite vengono considerate come riferimento per calcolare l'errore di misura presente nelle acquisizione da parte del sistema basato su marker ArUco.

In particolare, si discuteranno di seguito il set up sperimentale che ha permesso di eseguire prove con elevata ripetibilitá e, successivamente, il procedimento seguito nelle prove stesse, con i relativi risultati ottenuti.

5.1 SET UP SPERIMENTALE

Come si nota in Figura 24, é presente un piano orizzontale nel quale sono posizionati i marker ArUco, raggruppati in modo da formare una coppia di object come descritti in precedenza. Si noti che, come inizialmente mostrato in Figura 5 all'interno del capitolo 2, il piano é posizionato all'interno dell'area di lavoro coperta dal sistema OptiTrack, la quale é stata protetta da fonti di luce esterne in modo tale che esso possa essere in grado di rilevare con precisione i marcatori collocati sugli oggetti che sostengono i marker ArUco.

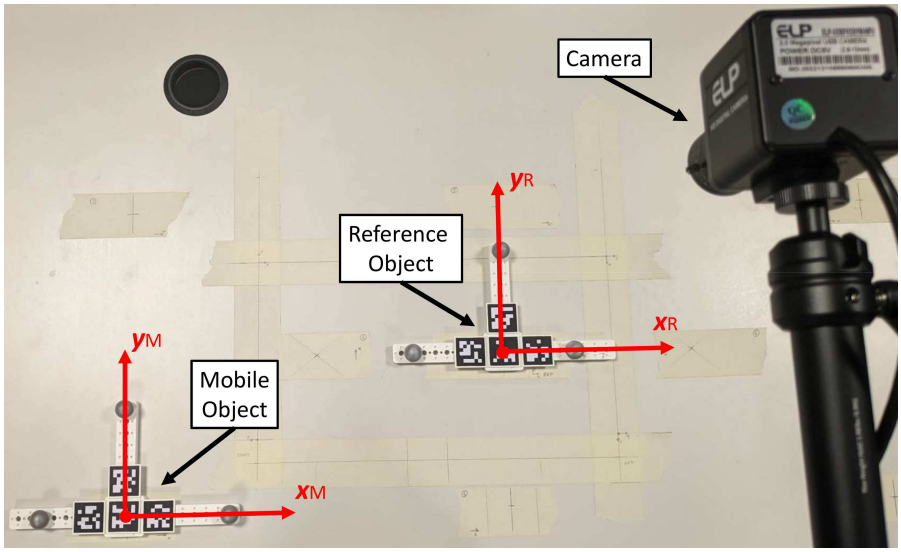


Figura 24: Posizionamento della camera e dei Reference e Mobile Object durante le prove sperimentali condotte

In particolare, si distinguono il Reference object, sul quale é fissato il frame $\mathscr{F}_R = \{O_R, (\mathbf{x}_R, \mathbf{y}_R, \mathbf{z}_R)\}$, ed il Mobile object, a cui é solidale il sistema di riferimento $\mathscr{F}_M = \{O_M, (\mathbf{x}_M, \mathbf{y}_M, \mathbf{z}_M)\}$. La camera invece, indipendentemente dal modello utilizzato, é posta parallelamente al piano ad una distanza verticale di circa 60cm, tramite un sostegno rigido che ne garantisce la stabilitá.

Partendo da tale configurazione, si é potuto studiare l'errore di misura della posa dei marker al variare della zona di lavoro coperta dalla videocamera utilizzata, spostando il Mobile object in diverse modalitá, in base alla tipologia di esperimento condotto, statico o dinamico, i quali saranno approfonditi di seguito. In particolare, in Figura 25 si riassume l'architettura ed il procedimento seguito per la raccolta dei dati in contemporanea da parte dei due sistemi di acquisizione, descritta approfonditamente nel capitolo 3. Sono infatti presenti due computer, il cui sincronismo viene dato tramite codice Matlab dalla macchina a cui é connesso il sistema basato sui marker ArUco tramite connessione ethernet. Durante lo svolgimento della prova, le dieci videocamere del sistema OptiTrack rilevano la posizione degli oggetti e, comunicando con il software Motive, permettono di identificarne la posa con accuratezza elevata. Allo stesso tempo, essi vengono rilevati anche dal software implementato, il quale sfrutta una singola videocamera o una camera con tecnologia stereo, permettendo poi un confronto dei dati raccolti.

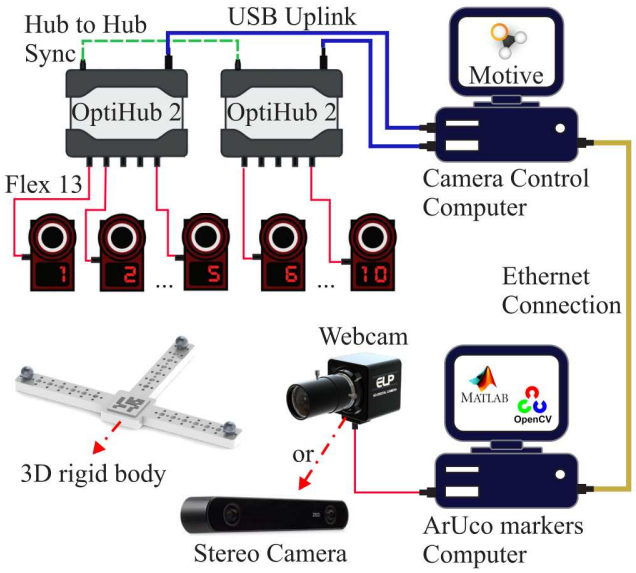


Figura 25: Architettura implementata per l'acquisizione di dati da parte dei due sistemi utilizzati, OptiTrack e ArUco-based [44]

5.2 DESCRIZIONE DELLE PROVE SPERIMENTALI

5.2.1 *Caso statico*

Il primo caso analizzato é quello statico, in cui si é posizionato sul piano orizzontale il Reference object, al centro dell'area coperta dalla videocamera, e mantenuto fermo per tutta la durata della prova, mentre il Mobile object ha assunto otto diverse pose, in termini di posizione ed orientamento come mostrate in Figura 26, e mantenuto fermo in ognuna di essa per una durata pari a 10 secondi. Durante tale intervallo, sono state acquisite il maggior numero di misure possibile da parte di entrambi i sistemi in funzione, le quali sono state successivamente mediate al fine di ottenere una misura unica confrontabile per ogni punto.

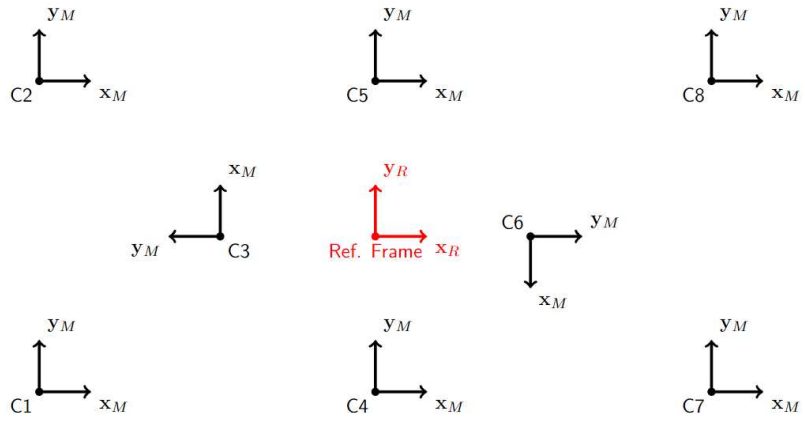


Figura 26: Posizionamento del Mobile object e Reference object durante la prova sperimentale svolta per analizzare le prestazioni del sistema sviluppato nel caso statico

Si noti che i punti da C1 a C8 sono stati posizionati in base all'area coperta dalla videocamera ELP, in quanto copre uno spazio minore rispetto alla videocamera con visione stereo ZED. Si é quindi coperto per intero il workspace, studiando il funzionamento dell'algoritmo sviluppato nella zona centrale, tramite le posizioni C3 e C6, e lungo il perimetro esterno, sfruttando i restanti punti. In termini di misure, facendo riferimento al frame $\mathscr{F}_R$, l'area coperta spazia da -40cm a 40cm lungo l'asse $\mathbf{x}_R$ e da -15cm a 15cm nella direzione verticale.

Per analizzare le prestazioni del sistema di Motion Capture sviluppato nel caso statico, si é calcolata la differenza presente tra le misure da esso raccolte e le informazioni messe a disposizione del sistema OptiTrack, di cui si riporta un esempio in Figura 27. Come detto in precedenza, le seconde sono state considerate come misure reali della

posizione del Mobile object rispetto il Reference object, permettendo di calcolare quindi l'errore di misura di posizione come:

$$\mathbf{e}_p = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = |\mathbf{P} - \mathbf{p}| \tag{34}$$

dove

- $\mathbf{P} \in \mathbb{R}^3$ indica la posizione reale di $\mathscr{F}_M$ nel sistema di riferimento fissato, fornita dal sistema OptiTrack.
- $\mathbf{p} \in \mathbb{R}^3$ rappresenta la posizione misurata dal sistema sviluppato del Mobile object rispetto al Reference object.

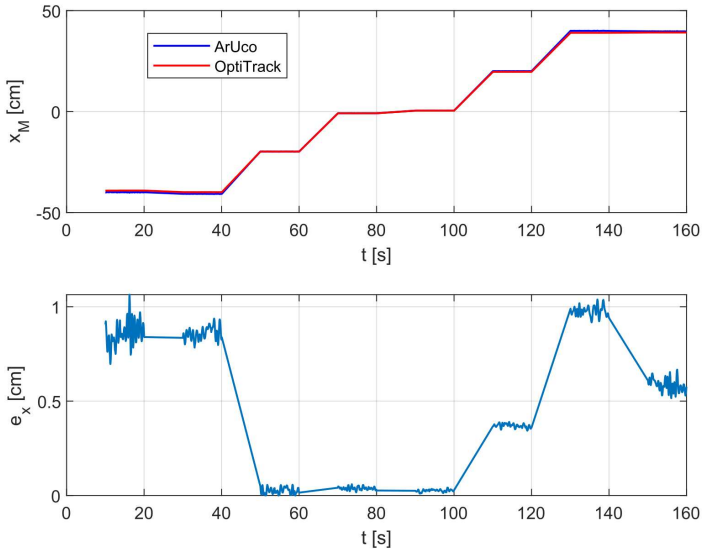


Figura 27: Posizione lungo l'asse orizzontale del Mobile Object rispetto al Reference Object, rilevata dai due sistemi di visione utilizzati, con la relativa differenza espressa in modulo, nel caso statico ed utilizzando la videocamera ELP

Per quanto riguarda l'orientamento, si é seguito il medesimo metodo, calcolando quindi la differenza tra le informazioni raccolte dai sistemi in esecuzione, utilizzando la notazione degli angoli di Eulero. Si ottiene quindi un errore di misura dell'orientamento definito come:

$$\mathbf{e}_\delta = \begin{bmatrix} e_\phi \\ e_\theta \\ e_\psi \end{bmatrix} = |\boldsymbol{\Delta} - \boldsymbol{\delta}| \tag{35}$$

dove, come nel caso precedente,

- $\boldsymbol{\Delta} \in \mathbb{R}^3$ rappresenta l'orientamento del Mobile object nel sistema di riferimento $\mathscr{F}_R$.

- $\boldsymbol{\delta} \in \mathbb{R}^3$ rappresenta l'orientamento del frame $\mathscr{F}_M$ nel frame solidale al Reference object.

Si evidenzia che, dato il fine di analizzare le prestazioni solamente in condizioni statiche, i dati sono stati raccolti esclusivamente durante gli intervalli in cui entrambi gli oggetti si trovavano in una posa fissa, in corrispondenza dei punti stabiliti. In particolare, gli intervalli di misura sono separati da intervalli altrettanto lunghi, ovvero 10 secondi, che permettono di effettuare lo spostamento del Mobile object da un punto al successivo, e durante i quali le misurazioni non vengono considerate.

5.2.2 *Caso dinamico*

Per studiare le prestazioni del sistema sviluppato nella situazione in cui uno dei due oggetti sia in moto, si é condotta una seconda tipologia di prova. In particolare, con riferimento alla Figura 28, si é mantenuto il Reference object in una posizione fissa, analoga al caso precedente, ovvero al centro dell'area coperta dalla videocamera, mentre si é spostato il Mobile object lungo un percorso prestabilito, durante il quale, oltre la posizione, si é variato solamente l'angolo di rotazione attorno all'asse $\mathbf{z}_R$ per quanto riguarda l'orientamento, come segue:

- dal punto C1 al punto C2, $\psi = 90$ gradi
- dal punto C2 al punto C5 e da C5 a CC8, $\psi = 0$ gradi
- dal punto C8 al punto C7, $\psi = -90$ gradi

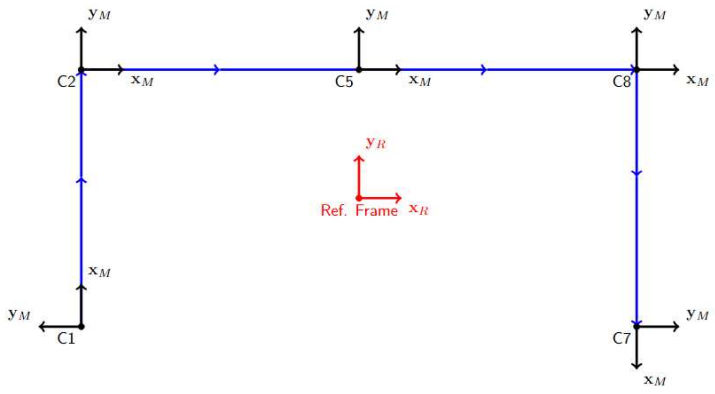


Figura 28: Posizionamento del Mobile object e Reference object durante la prova sperimentale nel caso dinamico

Si sottolinea inoltre che, essendo le prove condotte manualmente, ovvero spostando il Mobile object lungo la traiettoria stabilita a mano, non é assicurata una ripetibilitá eccellente dei dati raccolti. In particolare, nota la posizione dei punti sul piano come in Figura 29, si é

seguita la traiettoria retta che li congiunge, secondo l'ordine di cui sopra, trascinando sul piano di lavoro, costituito da una superficie liscia, il Mobile Object, mantenendo l'orientamento previsto per ciascun tratto. Per questo motivo, le prove sono state ripetute piú volte, fino a quando le misure acquisite con il sistema OptiTrack, e quindi molto accurate, fossero coerenti tra gli esperimenti condotti variando il numero di marker per ogni oggetto.

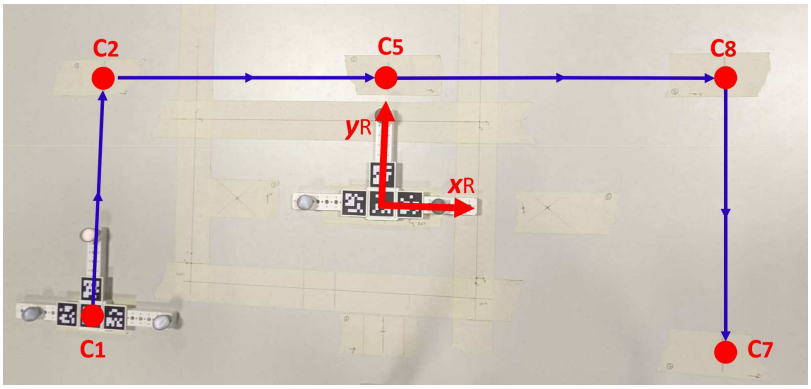


Figura 29: Traiettoria seguita trascinando manualmente sul piano il Mobile object, mantenendo il Reference object in posizione costante per tutta la durata della prova.

In Figura 30 sono riportate le coordinate nel piano del frame $\mathscr{F}_M$ rispetto al Reference object, durante le quattro prove eseguite variando il numero di marker impiegati ed utilizzando la videocamera ELP. Si nota che, nonostante non siano esattamente sovrapposte, a causa di velocitá di traslazione differente e del rumore nel posizionamento, le coordinate sul piano orizzontale sono piuttosto simili nel tempo, con una differenza dell'ordine del centimetro.

Per effettuare il paragone tra i sistemi analizzati, in modo simile al caso precedente, sono stati calcolati i valori medi di posizione ed orientamento del Mobile object, solamente nei tre tratti in cui esso si trova in movimento, mediandoli successivamente ottenendo un valore unico confrontabile.

Si riportano selle sezioni successive i risultati osservati dalle prove condotte utilizzando prima la singola videocamera digitale ELP, e in seguito sfruttando la visione stereo fornita dalle camera ZED2i.

5.3 RISULTATI

5.3.1 *Videocamera digitale HD ELP*

Confrontando i dati raccolti mediante il sistema sviluppato basato sui marker ArUco, utilizzando la videocamera ELP, con le informazioni fornite dal sistema MoCap OptiTrack, si é notato un andamento dell'errore di misura della posizione nel piano, ovvero lungo gli assi x e

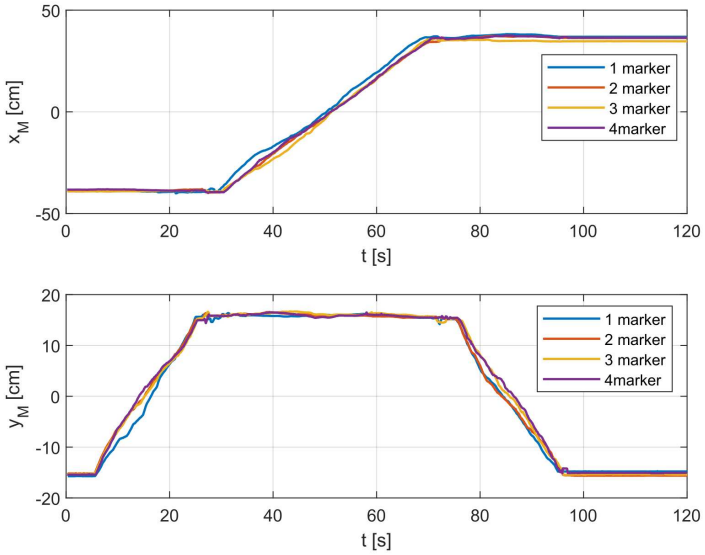


Figura 30: Coordinate x_M e y_M del Mobile object nel sistema di riferimento $\mathscr{F}_R$, rilevate tramite il sistema OptiTrack durante le prove sperimentali nel caso dinamico, al variare del numero di marker utilizzati, da 1 a 4

y, il quale partendo da un valore iniziale, presenta una diminuzione nella parte centrale per poi aumentare nella zona finale, fino a raggiungere circa il valore di origine. Come é facile intuire, ció é dovuto alla minor distorsione introdotta dalla lente del sensore nella zona di lavoro centrale, ovvero quando il Mobile object si trova in prossimitá dei punti C3, C4, C5 e C6, a differenza delle zone periferiche, in cui l'effetto ottico ha una maggiore influenza sull'accuratezza di misura. Tale fenomeno risulta piú evidente lungo la direzione orizzontale, a causa delle lunghezze coperte maggiori, del quale si riporta un esempio in Figura 31, dove é mostrato l'andamento della prima componente dell'errore nel tempo, ottenuta come differenza della posizione rilevata dai due sistemi in uso contemporaneamente, nel caso statico ed utilizzando quattro marker per ogni object.

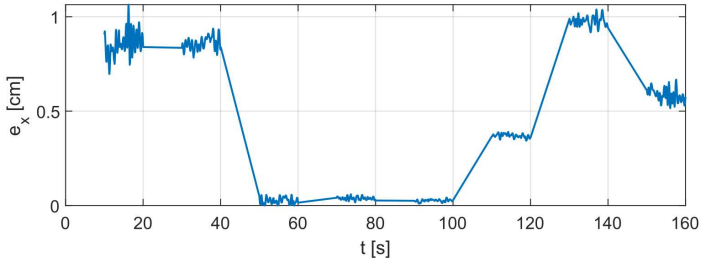
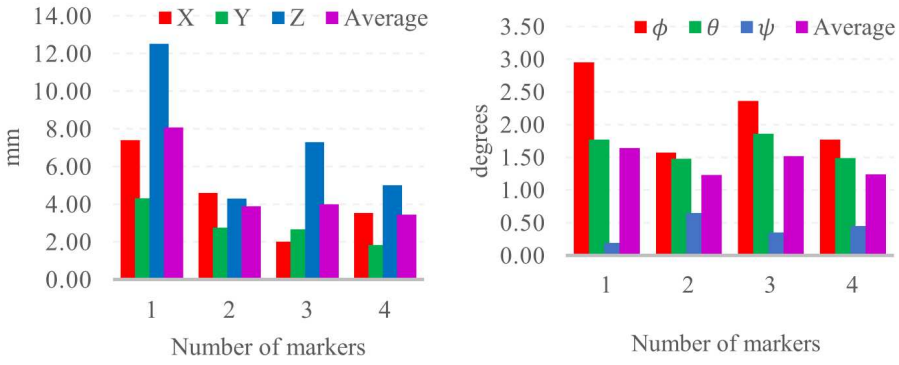


Figura 31: Andamento dell'errore lungo l'asse orizzontale nel tempo, utilizzando la videocamera ELP e 4 marker per ogni object, nel caso statico

Per ogni numero di marker utilizzati, si é poi calcolato il valore

medio presente durante gli intervalli in cui il Mobile object risulta stazionario in uno dei punti fissati, ottenendo quindi otto valori, i quali sono successivamente stati nuovamente mediati al fine di ottenere un dato unico per ogni asse. Per esempio, in ogni prova, durante la quale il numero di marker impiegati é fissato, si calcola l'errore di posizione medio per ogni intervallo di acquisizione, con riferimento ai punti indicati in Figura 26 e la definizione in (34), ottenendo i valori $\mathbf{e}_{p,C1}, \mathbf{e}_{p,C2}, \dots \mathbf{e}_{p,C8}$. Successivamente, eseguendo nuovamente la media dei risultati ricavati, si ottiene un dato unico di errore medio della prova il quale, scomposto in ogni sua componente, permette di analizzare gli andamenti lungo tutti e tre gli assi al variare del numero di marker utilizzati.

In Figura 32a sono riportati i valori medi di errore di posizione lungo gli assi $\mathbf{x}_R, \mathbf{y}_R$ e $\mathbf{z}_R$ presente nelle misurazioni effettuate dal sistema basato su marker ArUco sviluppato, con la corrispondente media, mentre in Figura 32b gli errori nell'identificazione degli angoli di rotazione attorno ai medesimi assi, in entrambi i casi usando la videocamera ELP nel caso statico. Si nota che passando da oggetti composti da un singolo marker ad oggetti costituiti da una coppia di marker, vi é un significativo decremento dell'errore medio, sia nella posizione, dove si osserva un decremento del 52%, che nell'orientamento, il quale mediamente viene ridotto di circa il 25%. Gli incrementi successivi tuttavia non hanno prodotto il risultato atteso, non influenzando positivamente l'errore di stima della posa, il quale si é notato rimanere all'interno di un range di valori abbastanza costante.



(a) (b)

Figura 32: Errore di misura medio (a) di posizione ed (b) di orientamento del sistema sviluppato basato su ArUco utilizzando una videocamera digitale ELP, al variare del numero di marker impiegati, nel caso statico [44]

Il sistema sviluppato fornisce quindi misure, in condizioni statiche, con una tolleranza di circa 9mm in posizione e 1.64 gradi nell'orientamento nel caso peggiore. Tali limiti vengono tuttavia ridotti impiegando almeno una coppia di marker per ogni oggetto, ottenendo come

risultato un errore medio di stima della posa pari a circa 4mm per la posizione e 1.3 gradi nell'orientamento.

Per quanto riguarda le misure dinamiche invece, i cui risultati sono sintetizzati in Figura 33a e 33b, gli errori medi sono maggiori, come atteso, rimanendo tuttavia minori di 9mm e 4 gradi nelle condizioni di minor accuratezza.

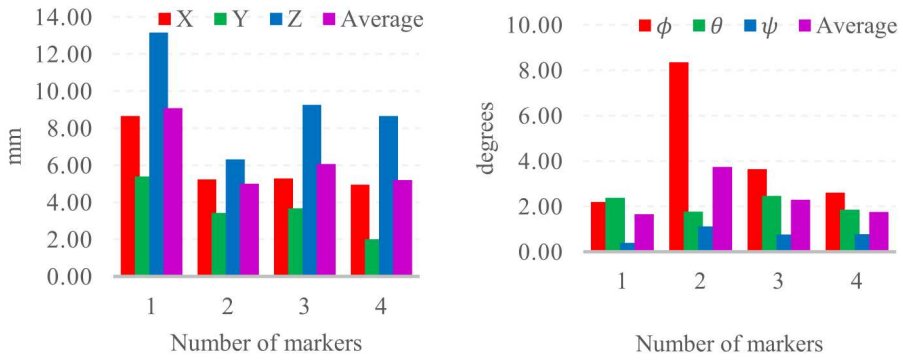


Figura 33: Errore di misura medio (a) di posizione ed (b) di orientamento del sistema sviluppato basato su ArUco utilizzando una videocamera digitale ELP, al variare del numero di marker impiegati, nel caso dinamico [44]

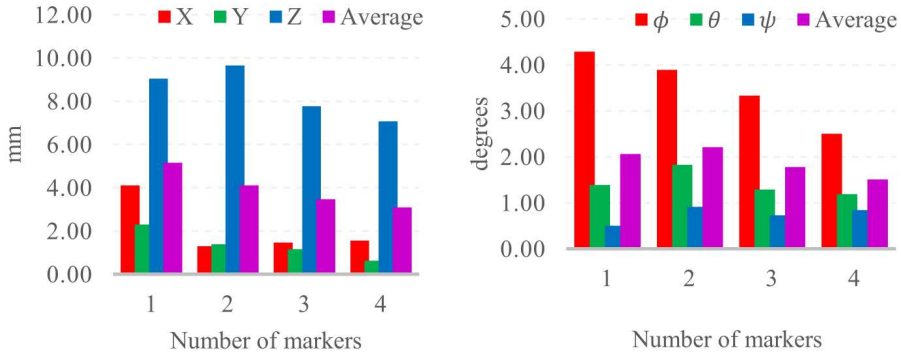
Dal punto di vista della complessitá computazionale, il tempo medio presente tra una acquisizione e la successiva da parte di tale algoritmo, impiegando una singola videocamera digitale, risulta pari a 41ms, il che perfette quindi di acquisire circa 244fps[1] a fronte degli 8.3ms presenti tra due acquisizioni eseguite dal sistema OptiTrack.

Infine, una ulteriore osservazione effettuata é che, per quanto riguarda la posizione, la terza componente dell'errore risulta essere maggiore delle altre due in tutti i casi, indipendentemente dal numero di marker impiegati. Questo é dovuto al fatto che la videocamera si trova con la lente esattamente parallela agli oggetti da rilevare, rendendo tale misura meno affidabile rispetto al rilevamento sul piano orizzontale. Per lo stesso motivo, il sistema implementato utilizzando una singola videocamera, presenta notevoli difficoltá nella rilevazione degli angoli di rotazione attorno agli assi x e y, rispettivamente detti ϕ e θ, mentre l'accuratezza delle misure aumenta sensibilmente per l'angolo ψ, legato alla rotazione attorno all'asse uscente dal piano. Una possibile soluzione proposta in questo lavoro per compensare tali aspetti, é l'utilizzo di una videocamera con visione stereo la quale, acquisendo il doppio delle informazioni da prospettive differenti, permette di ottenere una maggiore precisione nelle misure di profonditá richieste.

1 L'unitá di misura fps sta per *Frame per Second*, ed indica il numero di immagini acquisite in un secondo.

5.3.2 *Stereo camera ZED 2i*

Studiando i risultati ottenuti acquisendo le informazioni rilevanti la posa degli oggetti tramite stereo camera ZED2i nel caso statico, come riportato in Figura 34a per la posizione ed 34b per l'orientamento, si é notato che la misura della profonditá non presenta netti miglioramenti rispetto al sensore precedente. Nelle misura della posizione nel piano, al contrario, le misure risultano molto accurate, con errori al di sotto dei 2mm nella maggior parte dei casi, permettendo di avere un andamento dell'errore di posizione medio decrescente all'aumentare del numero di marcatori impiegati.



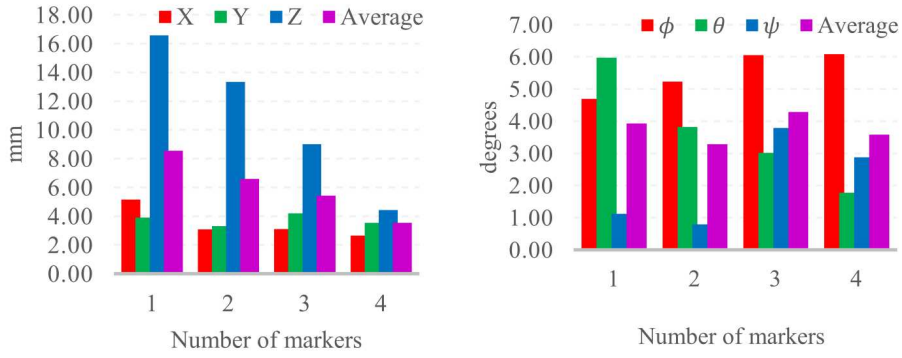
(a) (b)

Figura 34: Errore di misura medio (a) di posizione ed (b) di orientamento del sistema sviluppato basato su ArUco utilizzando una stereo camera ZED2i, al variare del numero di marker impiegati, nel caso statico [44]

Nell'orientamento invece, si nota che e_ψ risulta costantemente sotto al limite di 1 grado, a fronte di valori molto elevati nella prima componente dell'errore d orientamento, i quali ne aumentano il valore medio finale, presentando tuttavia un decremento variando il numero di marker impiegati. In definitiva, sfruttando la videocamera con visione stereo, si hanno misure statiche che presentano un errore di posizione mediamente inferiore ai 5mm e di orientamento di circa 2 gradi.

Osservando ora i dati raccolti dalla prova dinamica, riportati in Figura 35a e 35b, risulta ancora piú evidente l'effetto dell'aggiunta di marker sull'accuratezza della misura di profonditá, la quale aumenta di circa il 73% passando da oggetti composti da 1 a 4 marker. Analogamente, l'errore medio di posizione diminuisce del 60%, andando da un massimo di 8.5mm ad un minimo di 3.5mm, al contrario di quello di orientamento, che invece rimane sempre nei dintorni di 4 gradi.

Infine, il tempo medio misurato tra due frame successivi acquisiti dal sensore ZED2i, risulta pari a 110ms, notevolmente maggiore al



(a) (b)

Figura 35: Errore di misura medio (a) di posizione ed (b) di orientamento del sistema sviluppato basato su ArUco utilizzando una stereo camera ZED2i, al variare del numero di marker impiegati, nel caso dinamico [44]

caso precedente, lavorando quindi ad una frequenza di acquisizione di circa 91fps.

6

CONCLUSIONI E SVILUPPI FUTURI

Lo studio condotto e riportato in questo lavoro ha valutato con successo le prestazioni di un sistema di stima della posa low-cost basato su marker ArUco, in condizioni statiche e dinamiche, utilizzando prima una videocamera digitale ELP e successivamente una stereo camera ZED 2i, in condizioni di lavoro con illuminazione controllata e senza situazioni di occlusione. In tutte le prove condotte si é sfruttato un sistema di Motion Capture OptiTrack, con tecnologia optical-passive, come riferimento per le misurazioni effettuate, in quanto garantisce accuratezza e precisione molto elevate nelle circostanze analizzate, nel dettaglio del decimo di millimetro.

In particolare, si é riscontrato che l'accuratezza attesa del sistema sviluppato, utilizzando il sensore ELP, risulta 9mm e 4 gradi, rispettivamente per posizione ed orientamento, nel caso peggiore, ovvero impiegando oggetti composti da un singolo marker. Aumentandone tuttavia il numero, non si é notato un andamento decrescente, se non passando da uno a due marker per object.

Impiegando invece la stereo camera ZED 2i, si ottengono valori pari a 8.5mm e 4.3 gradi, considerando sempre la condizione di lavoro meno favorevole.

Con tali valori, il sistema di stima della posa sviluppato basato su marker ArUco, é adatto ad applicazioni con accuratezza richiesta al di sotto del centimetro in termini di posizionamento, e con un tempo computazionale minore di 41ms utilizzando un unico sensore ELP, il quale aumenta a 110ms impiegando la stereo camera ZED 2i.

Curiosamente, si é verificato che la stima della posa fornita dal sistema implementato non migliora utilizzando una telecamera dotata di visione stereo al posto di una singola webcam, cosí come aumentare il numero di marker non porta a miglioramenti significativi nella precisione della stessa. Tuttavia, impiegando la videocamera ZED 2i, l'andamento dell'errore medio, sia nel caso statico che in condizioni dinamiche, presenta l'andamento atteso in termini di posizione rilevata, il che lascia intuire che si possa approfondirne il funzionamento e raggiungere risultati migliori in quanto ad accuratezza della stima della posa di oggetti basata su marker ArUco.

I possibili ambiti di applicazione del sistema sviluppato, con l'accuratezza ottenuta, sono quelli dove non si richiede particolare precisione nel rilevamento dei movimenti, ma solamente una misurazione approssimata, senza doversi dotare di sistemi di Motion Capture moderni e ad elevato costo. In particolare, un esempio é il caso in cui

occorra rilevare i movimenti delle mani di un operatore durante la realizzazione di un task su un piano di lavoro, tramite l'applicazione di marker su un paio di guanti. Tali informazioni sono utili per verificare la produttivitá del soggetto, oppure l'ottimalitá della disposizione dei componenti ed attrezzature all'interno dell'area di lavoro. Il sistema sviluppato puó risultare utile anche in ambito robotico, per delimitare l'area di lavoro ed monitorare la posizione di una persona nei pressi del robot stesso. In questo caso infatti, é sufficiente rilevare la presenza di un soggetto all'interno del workspace del macchinario, avviando una procedura dedicata a tale situazione. Infine, in ambito medico, applicando dei marker sulle gambe o braccia del paziente, é possibile identificarne i movimenti durante la camminata o una determinata operazione. Ció fornisce informazioni approssimative riguardo la presenza o meno di problemi nella postura o nel funzionamento di specifiche zone del corpo.

Negli studi futuri che saranno condotti presso l'Universitá Politecnica di Valencia, il sistema di Motion Capture sviluppato, basato su marker ArUco, sará implementato su una scheda Nvidia Jetson Nano, la quale unisce a notevoli capacitá computazionali la possibilitá di comunicare con diversi dispositivi, tra cui una videocamera [25, 26]. Inizialmente si utilizzerá una stereo camera ZED 2i, alla quale ne sará aggiunta una seconda, in modo da studiarne le prestazioni e l'eventuale riduzione dell'errore di stima. Infine verrá analizzata la possibilitá di implementare un sistema di Motion Capture markerless, ovvero che non necessiti della presenza di marcatori all'interno dell'area di lavoro.

APPENDICI

A

CODICE SVILUPPATO PER VIDEOCAMERA ELP

```
% ********************************
%% SET UP
% ********************************

% Set marker size
markerSize = 3;    % [cm]

% Set objects markers
IdObj = [0  1  2  ;    % Reference Object
         4  5  6 ];    % Mobile Object

% Create data structures
vector_time = [];           % Time
vector_T_markers_obj = [];  % Markers position
vector_R_markers_obj = [];  % Markers orientation

% Create global variable for showing frames
global im1

% Configuration of the timer for showing the frames
% Timer period
ptimer = 0.1;
% Delay before starting the timer
dtimer = 0.01;
% Creation of the timer
FrameTimer = timer('ExecutionMode', 'fixedSpacing', ...
                   'Name','FrameTimer',...
                   'Period', ptimer,...
                   'StartDelay',dtimer,...
                   'BusyMode', 'drop',...
                   'TimerFcn', @ShowFrameFcn);

% Camera parameters - ELP Camera
load('camMatrix.mat')
load('distCoeff.mat')

% Setting the recording time
recording_time_input = input('Set the recording time in the
    following format: hh:mm:ss [00:01:00] ', 's');
% Check the input value
```

```
if isempty(recording_time_input)
    recording_time_input = '00:01:00'; % default value
end
rec_time_end = seconds(duration(recording_time_input)); %[s]

% Capture initial image
frame = snapshot(cam);  % capture a frame
im1 = frame;

% Start of the timer
if strcmp(FrameTimer.Running, 'off')
    start(FrameTimer);
end

% Start counting recording time
rec_time = 0;
start_rec = tic;

% ********************************
%% MARKERS DETECTION - ELP CAMERA
% ********************************

% Start OptiTrack system
NatnetClientfun.startRecord;
% Start ArUco system
while rec_time < rec_time_end

    % Capture frame from the camera
    frame = snapshot(cam);
    % Markers detection in the frame
    [frame, T, R, IdDetected] = aruco_nano_detector(frame,
        camMatrix, distCoeff, markerSize);

    % View results in the video player
    im1 = frame;

    % Group markers data for objects
    % Start column for storage data of one object
    colStart = 1;
    % Iterare for all the Objects
    for i=1:size(IdObj,1)
        % Markers Id of one object
        MarkersObj = IdObj(i,:);
        % End column for storage data of one object
        colEnd = colStart + size(MarkersObj,2) - 1;
        % Call function GetMarkers
```

```
        [T_ordered(:,colStart:colEnd), R_ordered(:,colStart:
            colEnd)] = GetMarkers(T, R, IdDetected,
            MarkersObj);
        % New end column
        colStart = colEnd + 1;
    end

    % Storage markers data
    vector_T_markers(end+1,:) = reshape(T_ordered,1,[]);
    vector_R_markers(end+1,:) = reshape(R_ordered,1,[]);

    % Save time value
    rec_time = toc(start_rec);
    vector_time(end+1,1) = rec_time;
end
```

B

CODICE SVILUPPATO PER STEREO CAMERA ZED

```
% ********************************
%% SET UP
% ********************************

% Set marker size
markerSize = 3;    % [cm]

% Set objects markers
IdObj = [0  1  2  ;    % Reference Object
         4  5  6 ];    % Mobile Object

% Create data structures
vector_time = [];           % Time
vector_T_markers_obj = [];  % Markers position
vector_R_markers_obj = [];  % Markers orientation

% Create global variables for showing frames
global im1
global im2

% Configuration of the timer for showing the frames
% Timer period
ptimer = 0.1;
% Delay before starting the timer
dtimer = 0.01;
% Creation of the timer
FrameTimer = timer('ExecutionMode', 'fixedSpacing', ...
                   'Name','FrameTimer',...
                   'Period', ptimer,...
                   'StartDelay',dtimer,...
                   'BusyMode', 'drop',...
                   'TimerFcn', @ShowFrameFcn);

% Camera parameters - ZED Camera
% Configuration of the camera
InitParameters.camera_resolution = 1;   % 1=HD1080
InitParameters.camera_fps = 30;
InitParameters.depth_mode = 0;          % NONE
InitParameters.coordinate_units = 1;    % CENTIMETER
InitParameters.coordinate_system = 0;   % IMAGE
```

```
% Define minimun and maximum depth (in CENTIMETER)
InitParameters.depth_minimum_distance = 40;
InitParameters.depth_maximum_distance = 100;
% Setup runtime parameters
RuntimeParameters.sensing_mode = 0;     % 0=STANDARD

% Open the mexZED file
result = mexZED('open', InitParameters);

% if the camera is opened with success, get parameters
if(strcmp(result,'SUCCESS'))
    % Get cameras informations
    camInfo = mexZED('getCameraInformation');
    % Left sensor parameters
    Left_Sensor_CamMatrix = [
        camInfo.left_cam.fx, 0, camInfo.left_cam.cx;...
        0, camInfo.left_cam.fy, camInfo.left_cam.cy;...
        0, 0, 1];
    Left_Sensor_distCoeff = camInfo.left_cam.disto;
    % Right sensor parameters
    Right_Sensor_CamMatrix = [
        camInfo.right_cam.fx, 0, camInfo.right_cam.cx;...
        0, camInfo.right_cam.fy, camInfo.right_cam.cy;...
        0, 0, 1];
     Right_Sensor_distCoeff = camInfo.right_cam.disto;

% Setting the recording time
recording_time_input = input('Set the recording time in the
    following format: hh:mm:ss [00:01:00] ', 's');
% Check the input value
if isempty(recording_time_input)
    recording_time_input = '00:01:00'; % default value
end
rec_time_end = seconds(duration(recording_time_input)); %[s]

% Start of the timer
if strcmp(FrameTimer.Running, 'off')
    start(FrameTimer);
end

% Start counting recording time
rec_time = 0;
start_rec = tic;

% ********************************
%% MARKERS DETECTION - ZED CAMERA
```

```
% ********************************

% Start OptiTrack system
NatnetClientfun.startRecord;
% Start ArUco system
while rec_time < rec_time_end

    % Capture images with both sensors
    frame_left = mexZED('retrieveImage', 0);
    frame_right = mexZED('retrieveImage', 1);

    % Markers detection in the frames
    [frame_left, T_Left, R_Left, Id_Left] =
        aruco_nano_detector(frame_left,
        Left_Sensor_CamMatrix, Left_Sensor_distCoeff,
        markerSize);
    [frame_right, T_Right, R_Right, Id_Right] =
        aruco_nano_detector(frame_right,
        Right_Sensor_CamMatrix, Right_Sensor_distCoeff,
        markerSize);

    % View results in the video player
    im1 = frame_left;
    im2 = frame_right;

    % Group markers data for objects
    % Start column for storage data of one object
    colStart = 1;
    % Iterare for all the Objects
    for i=1:size(IdObj,1)
        % Markers Id of one object
        MarkersObj = IdObj(i,:);
        % End column for storage data of one object
        colEnd = colStart + size(MarkersObj,2) - 1;

        % Call function GetMarkers
        % Left Cam
        [T_ordered_Left(:,colStart:colEnd), R_ordered_Left
            (:,colStart:colEnd)] = GetMarkers(T_Left, R_Left
            , Id_Left, MarkersObj);
        % Right Cam
        [T_ordered_Right(:,colStart:colEnd), R_ordered_Right
            (:,colStart:colEnd)] = GetMarkers(T_Right,
            R_Right, Id_Right, MarkersObj);

        % New end column
```

```
        colStart = colEnd + 1;
    end

    % Store markers data
    vector_T_markers_Left(end+1,:)  = reshape(T_ordered_Left
        ,1,[]);
    vector_T_markers_Right(end+1,:) = reshape(
        T_ordered_Right,1,[]);
    vector_R_markers_Left(end+1,:)  = reshape(R_ordered_Left
        ,1,[]);
    vector_R_markers_Right(end+1,:) = reshape(
        R_ordered_Right,1,[]);

    % Save time value
    rec_time = toc(start_rec);
    vector_time(end+1,1) = rec_time;
end

% ********************************
%% MARKERS POSE AVERAGE CALCULATION
% ********************************

% Filling the NaN values of Position and Orientation matrix
vector_T_markers_Left = fillmissing(vector_T_markers_Left,'
    linear');
vector_T_markers_Right = fillmissing(vector_T_markers_Right,
    'linear');
vector_R_markers_Left = fillmissing(vector_R_markers_Left,'
    linear');
vector_R_markers_Right = fillmissing(vector_R_markers_Right,
    'linear');

%Calculate the number of markers
n = size(IdObj,1) * size(IdObj,2);

% For every stepTime (row)
for j=1:size(vector_time,1)

    % Start column
    colStart = 1;

    % For every marker (column)
    for i=1:n

        % End column
        colEnd = colStart + 2;
```

```
        % Extract the marker information from the two
            cameras
        T_marker_Left = reshape(vector_T_markers_Left(j,
            colStart:colEnd),[3,1]);
        T_marker_Right = reshape(vector_T_markers_Right(j,
            colStart:colEnd),[3,1]);
        R_marker_Left = reshape(vector_R_markers_Left(j,
            colStart:colEnd),[3,1]);
        R_marker_Right = reshape(vector_R_markers_Right(j,
            colStart:colEnd),[3,1]);

        % MARKER POSITION CALCULATION
        % Move Right camera measure into Left Camera
            reference system
        T_marker_Right_moved = T_marker_Right + camInfo.t';
        % Average of the two measured positions
        T_marker = (T_marker_Left + T_marker_Right_moved)/2;

        % MARKER ORIENTATION CALCULATION
        % Rotation matrix
        RotMat_marker_Left = Rodrigues(R_marker_Left);
        RotMat_marker_Right = Rodrigues(R_marker_Right);
        % Eulero angles
        Angles_marker_Left = rotm2eul(RotMat_marker_Left,'
            ZYX');
        Angles_marker_Right = rotm2eul(RotMat_marker_Right,'
            ZYX');

        % Average of the two measured orientationa
        Angles_marker = (Angles_marker_Left +
            Angles_marker_Right)/2;
        RotMat_marker = eul2rotm(Angles_marker,'ZYX');

        % Marker orientation 3x1
        R_marker = Rodrigues(RotMat_marker);

        % Store markers data
        vector_T_markers(j,colStart:colEnd) = reshape(
            T_marker,1,[]);
        vector_R_markers(j,colStart:colEnd) = reshape(
            R_marker,1,[]);

        % New start column
        colStart = colEnd + 1;
    end
```

```
end
```

C

CODICE SVILUPPATO PER POST PROCESSING

```
% ********************************
%% POST PROCESSING
% ********************************

% RELATIVE POSITION IN GLOBAL SYSTEM OR
% Frame_R = Reference Object centred Frame
% Frame_M = Mobile Object centred Frame
% Frame_C = Camera cetred Frame
% Hp: First Object  --> Reference object
%     Second Object --> Movement object

vector_Trel = [];
vector_angles = [];
vector_d = [];

% Number of marker for every object
m = size(IdObj,2);

% For every steptime
for j=1:size(vector_time,1)
    % Reset start column
    colStart = 1;

    % Iterate for all the Objects
    for i=1:size(IdObj,1)

        % Set end column
        colEnd = colStart + 3*m - 1;

        % Create matrixes for position and orientation
        T_markers = reshape(vector_T_markers_obj(j,colStart:
            colEnd),[3,m]);
        R_markers = reshape(vector_R_markers_obj(j,colStart:
            colEnd),[3,m]);

        % Get one object pose
        [T_Obj(:,i), R_Obj(:,i)] = GetObjectOffline(
            T_markers, R_markers);
```

```
        % Move start column
        colStart = colEnd + 1;

    end

    % Reference Object pose
    Tref = T_Obj(:,1);
    Rref = R_Obj(:,1);
    % Mobile Object pose
    Tmob = T_Obj(:,2);
    Rmob = R_Obj(:,2);

    % ROTATION MATRIX
    % From Frame_R to Frame_C
    RotMat_RC = Rodrigues(Rref_C);
    % From Frame_M to Frame_C
    RotMat_MC = Rodrigues(Rmob_C);

    % POSITION
    % Relative position in camera system, FrameC
    Trel_C = Tmob_C - Tref_C;
    % Relative position in Reference object system, OR
    Trel_R = (RotMat_RC)' * Trel_C;

    % ORIENTATION
    RotMat_rel = (RotMat_RC)' * RotMat_MC;
    Angles_rel_R = rotm2eul(RotMat_rel,"ZYX") * 180/pi;  %Z(
        phi), Y(theta), X(psi)

end

% SIGNAL SOOMTHING
% Position, linear velocity and linear acceleration (
    suvizado_optimo-IBV)
[nfunc,pva]=suavizado_optimo(vector_Trel,vector_time);
% Extraction of the position
Trel_sm=pva(:,:,1);

% Orientation, angular velocity and angular acceleration (
    suvizado_optimo-IBV)
[nfunc,oaa]=suavizado_optimo(vector_angles,vector_time);
% Extraction of the orientation
angles_sm=oaa(:,:,1);
```

D

FUNZIONI AUSILIARI SVILUPPATE

FUNZIONE "ARUCO_NANO_DETECTOR"

```
%% aruco_nano_detector.cpp file

#include "opencvmex.hpp"
using namespace cv;
#include "aruco_nano.h"

% Pointer to the image
cv::Ptr<cv::Mat>img = ocvMxArrayToImage_uint8(prhs[0],true);
% Pointer to the camera parameters and marker size
cv::Ptr<cv::Mat> camMatrix = ocvMxArrayToImage_double(prhs
    [1], true);
cv::Ptr<cv::Mat> distCoeff = ocvMxArrayToImage_double(prhs
    [2], true);
double *markersSize = mxGetPr(prhs[3]);

% Markers detection
auto markers = aruconano::MarkerDetector::detect(*img);

% Creation of mxArray for the outputs
int numRows = 3;
plhs[1] = mxCreateDoubleMatrix(numRows, markers.size(),
    mxREAL); % Rotations
plhs[2] = mxCreateDoubleMatrix(numRows, markers.size(),
    mxREAL); % Transaltions
plhs[3] = mxCreateDoubleMatrix(1, markers.size(), mxREAL);
          % Markers ID

% Get the pointers to the output mxArray
double *Tout = mxGetPr(plhs[1]);
double *Rout = mxGetPr(plhs[2]);
double *Idout = mxGetPr(plhs[3]);

int col = 0;
int row = 0;
int index = 0;

% Markers pose and Id
for(const auto &m:markers){
```

```
    m.draw(*img);
    auto r_t = m.estimatePose(*camMatrix, *distCoeff, *
        markersSize);

    cv::Point3d T(r_t.second);
    cv::Point3d R(r_t.first);

    row = 0;
    index = col * numRows + row;
    Tout[index] = T.x;
    Rout[index] = R.x;

    row = 1;
    index = col * numRows + row;
    Tout[index] = T.y;
    Rout[index] = R.y;

    row = 2;
    index = col * numRows + row;
    Tout[index] = T.z;
    Rout[index] = R.z;

    Idout[col] = m.id;
    col++;
}

    plhs[0] = ocvMxArrayFromImage_uint8(*img);
```

FUNZIONE "GET_MARKERS"

```
%% Get markers function

function [T_markers, R_markers] = GetMarkers(T, R, Id,
    markersId)

    if ~all(ismember(markersId,Id))   % if there is a not-
        detected marker

        T_markers = NaN(3, size(markersId,2));
        R_markers = NaN(3, size(markersId,2));

    else

        % Look for the object markers
```

```
        for j=1:size(markersId,2)

            Col(j) = find(Id == markersId(j));

            % Object markers matrix, Position and
                Orientation
            T_markers(:,j) = T(:,Col(j));
            R_markers(:,j) = R(:,Col(j));
        end

    end

end
```

FUNZIONE "RODRIGUES"

```
%% Rodrigues.cpp file

#include "opencvmex.hpp"
#include "opencv2/calib3d.hpp"
#include <opencv2/core.hpp>
using namespace std;
using namespace cv;

% I/O pointers
cv::Ptr<cv::Mat> RotVec = ocvMxArrayToMat_double(prhs[0],
    true);
cv::Mat RotMat;
% Rodrigues function
cv::Rodrigues(*RotVec, RotMat);
plhs[0] = ocvMxArrayFromMat_double(RotMat);
```

FUNZIONE "GET_OBJECT"

```
%% Get object function

function [T_object, R_object] = GetObject(T_markers,
    R_markers)
% This function create an object, knowing the data of every
    marker

% OBJECT TRASLATION
```

```
switch size(T_markers,2)

    case 1
        Tx_object = T_markers(1,1);
        Ty_object = T_markers(2,1);
        Tz_object = T_markers(3,1);

    case 2
        Tx_object = (T_markers(1,1) + T_markers(1,2))/2;
        Ty_object = (T_markers(2,1) + T_markers(2,2))/2;
        Tz_object = (T_markers(3,1) + T_markers(3,2))/2;

    case 3
        xa = T_markers(1,1);
        ya = T_markers(2,1);
        xb = T_markers(1,2);
        yb = T_markers(2,2);
        xc = T_markers(1,3);
        yc = T_markers(2,3);

        m = (yb-ya)/(xb-xa);

        Tx_object = ( xa + (xc/m^2) + (yc-ya)/m )/(1 + 1/m
            ^2);
        Ty_object = m*(Tx_object-xa)+ya;
        Tz_object = mean(T_markers(3,:));

    case 4
        xa = T_markers(1,1);
        ya = T_markers(2,1);
        xb = T_markers(1,2);
        yb = T_markers(2,2);
        xc = T_markers(1,3);
        yc = T_markers(2,3);
        xd = T_markers(1,4);
        yd = T_markers(2,4);

        mac = (yc-ya)/(xc-xa);
        mbd = (yd-yb)/(xd-xb);

        Tx_object = ( xa - (xb*mbd/mac) + (yb-ya)/mac )/(1 -
             mbd/mac);
        Ty_object = mac*(Tx_object-xa)+ya;
        Tz_object = mean(T_markers(3,:));
end
```

```
T_object = [Tx_object; Ty_object; Tz_object];

% OBJECT ORIENTATION
for k=1:size(R_markers,2)
    marker_matrix = Rodrigues(R_markers(:,k));
    marker_angles(k,:) = rotm2eul(marker_matrix,'ZYX');
end
object_angles = [mean(marker_angles(:,1)), mean(
   marker_angles(:,2)), mean(marker_angles(:,3))];
R_object_3x3 = eul2rotm(object_angles,'ZYX');
R_object = Rodrigues(R_object_3x3);

end
```

BIBLIOGRAFIA

[1] Atle Aalerud, Joacim Dybedal, and Geir Hovland. Automatic calibration of an industrial rgb-d camera network using retroreflective fiducial markers. *Sensors*, 19(7):1561, 2019. doi: https://doi.org/10.3390/s19071561. URL https://www.mdpi.com/1424-8220/19/7/1561.

[2] Lopez-Ceron Alberto and Canas Jose Maria. Accuracy analysis of marker-based 3d visual localization. *Actas de las XXXVII Jornadas de Automatica 7, 8 y 9 de septiembre de 2016, Madrid*, pages 1120–1131, 2022. doi: https://doi.org/10.17979/spudc.9788497498081.1124. URL https://ruc.udc.es/dspace/handle/2183/29246.

[3] Danilo Avola, Luigi Cinque, Gian Luca Foresti, Cristina Mercuri, and Daniele Pannone. A practical framework for the development of augmented reality applications by using aruco markers. In *Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,*, pages 645–654. INSTICC, SciTePress, 2016. ISBN 978-989-758-173-1. doi: 10.5220/0005755806450654. URL https://www.scitepress.org/Papers/2016/57558/.

[4] Armen Barsegyan. Cgi coffee, perception neuron inertial motion capture vs optical mocap systems and the first production motion capture session experience. URL https://cgicoffee.com/blog/2017/04/first-production-motion-capture-session-report.

[5] K. E. Bisshopp. Rodrigues formula and the screw matrix. *Journal of Engineering for Industry*, 91(1):179–184, 02 1969. ISSN 0022-0817. doi: 10.1115/1.3591509. URL https://doi.org/10.1115/1.3591509.

[6] John Brugman. Coding and colors: A practical approach to hex and rgb values. URL https://medium.com/@brugmanj/coding-and-colors-a-practical-approach-to-hex-and-rgb.-values-9a6e98720b25.

[7] Qing Chen, Yu Zhou, Yong Wang, Mao Mao Zhu, Lei Guo, and Chang Xu He. Research on stability and accuracy of the OptiTrack system based on mean error. 11884:118841C, October 2021. doi: 10.1117/12.2605796. URL https://ui.adsabs.harvard.edu/abs/2021SPIE11884E..1CC.

[8] COEX Clover. Aruco marker detection. URL https://clover.coex.tech/en/aruco_marker.html.

[9] Shenzhen Ailipu Technology Co. Elp 3840x2160 4k usb camera with cs 2.8-12mm varifocal lens elp-usb4khdr01-mfv (2.8-12mm). URL https://www.elpcctv.com/elp-3840x2160-4k-usb-camera-with-cs-2812mm-varifocal.-lens-elpusb4khdr01mfv-2812mm-p-181.html.

[10] Yann Desmarais, Denis Mottet, Pierre Slangen, and Philippe Montesinos. A review of 3d human pose estimation algorithms for markerless motion capture. *Computer Vision and Image Understanding*, 212:103275, 2021. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2021.103275. URL https://www.sciencedirect.com/science/article/pii/S1077314221001193.

[11] Hao Dong and Aura Ganz. Cost efficient virtual environment generation framework using annotated panoramic videos. *IEEE Access*, PP:1–1, 08 2019. doi: 10.1109/ACCESS.2019.2937877. URL https://www.researchgate.net/publication/335435865_Cost_Efficient_Virtual_Environment_Generation_Framework_Using_Annotated_Panoramic_Videos.

[12] Matthew Field, David Stirling, Fazel Naghdy, and Zengxi Pan. Motion capture in robotics review. pages 1697–1702, 2009. doi: 10.1109/ICCA.2009.5410185. URL https://ieeexplore.ieee.org/document/5410185.

[13] Joshua S. Furtado, Hugh H. T. Liu, Gilbert Lai, Herve Lacheray, and Jason Desouza-Coelho. Comparative analysis of optitrack motion capture systems. pages 15–31, 2019. URL https://link.springer.com/chapter/10.1007/978-3-030-17369-2_2.

[14] S. Garrido-Jurado, R. Munoz-Salinas, F.J. Madrid-Cuevas, and M.J. MarÃn-JimÃ©nez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2014.01.005. URL https://www.sciencedirect.com/science/article/pii/S0031320314000235.

[15] S. Garrido-Jurado, R. Munoz-Salinas, F.J. Madrid-Cuevas, and R. Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481–491, 2016. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2015.09.023. URL https://www.sciencedirect.com/science/article/pii/S0031320315003544.

[16] Clint Hansen, David Gibas, Jean-Louis Honeine, Nasser Rezzoug, Philippe Gorce, and Brice Isableu. An inexpensive solution for motion analysis. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 228 (3):165–170, 2014. doi: 10.1177/1754337114526868. URL https://journals.sagepub.com/doi/10.1177/1754337114526868.

[17] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997. doi: 10.1109/CVPR.1997.609468. URL https://ieeexplore.ieee.org/document/609468.

[18] Weiming Hu, Tieniu Tan, Liang Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(3):334–352, 2004. doi: 10.1109/TSMCC.2004.829274. URL https://ieeexplore.ieee.org/document/1310448.

[19] StereoLabs Inc. How to use matlab with zed, . URL https://www.stereolabs.com/docs/matlab.

[20] StereoLabs Inc. Zed 2i stereo camera, . URL https://www.stereolabs.com/en-it/store/products/zed-2i.

[21] MathWorks Italia. Camera calibrator, estimate geometric parameters of a single camera, . URL https://it.mathworks.com/help/vision/ref/cameracalibrator-app.html.

[22] MathWorks Italia. What is camera calibration?, . URL https://it.mathworks.com/help/vision/ug/camera-calibration.html.

[23] MathWorks Italia. Matlab opencv interface, . URL https://it.mathworks.com/discovery/matlab-opencv.html.

[24] Soohwan Kim and Minkyoung Kim. Rotation representations and their conversions. *IEEE Access*, 11:6682–6699, 2023. doi: 10.1109/ACCESS.2023.3237864. URL https://ieeexplore.ieee.org/document/10019271/similar#similar.

[25] Agus Kurniawan. *Introduction to NVIDIA Jetson Nano*, pages 1–6. Apress, Berkeley, CA, 2021. ISBN 978-1-4842-6452-2. doi: 10.1007/978-1-4842-6452-2_1. URL https://doi.org/10.1007/978-1-4842-6452-2_1.

[26] Agus Kurniawan. *NVIDIA Jetson Nano Camera*, pages 85–105. Apress, Berkeley, CA, 2021. ISBN 978-1-4842-6452-2. doi: 10.1007/978-1-4842-6452-2_6. URL https://doi.org/10.1007/978-1-4842-6452-2_6.

[27] Fong Kenneth N. K. Lam Winnie W. T., Tang Yuk Ming. A systematic review of the applications of markerless motion capture (mmc) technology for clinical measurement in rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 2023. doi: 10.1186/s12984-023-01186-9. URL https://link.springer.com/article/10.1186/s12984-023-01186-9.

[28] Gun Lee, Gerard Kim, and Mark Billinghurst. Immersive authoring: What you experience is what you get (wyxiwyg). *Commun. ACM*, 48:76–81, 01 2005. URL https://www.researchgate.net/publication/220420015_Immersive_authoring_What_You_eXperience_Is_What_You_Get_WYXIWYG.

[29] Lian-Wang Lee, Hsin-Han Chiang, and I-Hsum Li. Development and control of a pneumatic-actuator 3-dof translational parallel manipulator with robot vision. *Sensors*, 19(6), 2019. ISSN 1424-8220. doi: 10.3390/s19061459. URL https://www.mdpi.com/1424-8220/19/6/1459.

[30] Kazusuke Maenaka. Mems inertial sensors and their applications. pages 71–73, 2008. doi: 10.1109/INSS.2008.4610859. URL https://ieeexplore.ieee.org/document/4610859.

[31] Adam Marut, Konrad Wojtowicz, and Krzysztof Falkowski. Aruco markers pose estimation in uav landing aid system. pages 261–266, 2019. doi: 10.1109/MetroAeroSpace.2019.8869572. URL https://ieeexplore.ieee.org/document/8869572.

[32] MathWorks. Calibration patterns. URL https://it.mathworks.com/help/vision/ug/calibration-patterns.html.

[33] Kalaitzakis Michail, Brennan Cain, Carroll Sabrina, Ambrosi Anand, Whitehead Camden, and Vitzilaios Nikolaos. Fiducial markers for pose estimation. *Journal of Intelligent and Robotic Systems*, 2021. doi: 10.1007/s10846-020-01307-9. URL https://doi.org/10.1007/s10846-020-01307-9.

[34] Ogre. Augmented reality made simple with ogre and opencv. URL https://www.ogre3d.org/2020/12/24/augmented-reality-made-simple-with-ogre-and-opencv.

[35] OpenCV. Open computer vision library, . URL https://opencv.org/.

[36] OpenCV. cv::aruco namespace reference, . URL https://docs.opencv.org/4.x/d4/d17/namespacecv_1_1aruco.html.

[37] OpenCV. Opencv documentation, . URL https://docs.opencv.org/2.4/index.html.

[38] OpenCV. Detection of aruco markers, . URL https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html.

[39] OptiTrack. Built your own motion capture system, . URL https://optitrack.com/systems/#virtual-reality/flex-13/12.

[40] OptiTrack. Motive, optical motion capture software, . URL https://optitrack.com/software/motive/.

[41] OptiTrack. Optitrack calibration tools, . URL https://optitrack.com/accessories/calibration-tools/.

[42] MATLAB Central File Exchange Or Hirshfeld. Natnet (motive/arena) simple sample for location data by or hirshfeld. URL https://www.mathworks.com/matlabcentral/fileexchange/48279-natnet-motive-arena-simple-sample-for-location.-data-by-or-hirshfeld.

[43] Anton Yu. Poroykov, Pavel Kalugin, Sergey Shitov, and I.A. Lapitskaya. Modeling aruco markers images for accuracy analysis of their 3d pose estimation. *Proceedings of the 30th International Conference on Computer Graphics and Machine Vision (GraphiCon 2020). Part 2*, 2020. URL https://api.semanticscholar.org/CorpusID:232286011.

[44] José L. Pulloquinga, Davide Corrata, Vicente Mata, Ángel Valera, and Marina Vallés. Experimental analysis of pose estimation based on aruco markers. In *Innovations in Industrial Engineering III*, pages 138–149, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-61582-5. URL https://link.springer.com/chapter/10.1007/978-3-031-61582-5_12.

[45] Francisco J. Romero-Ramirez, Rafael Munoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38–47, 2018. ISSN 0262-8856. doi: https://doi.org/10.1016/j.imavis.2018.05.004. URL https://www.sciencedirect.com/science/article/pii/S0262885618300799.

[46] Sara Roos-Hoefgeest, Ignacio Alvarez Garcia, and Rafael C. Gonzalez. Mobile robot localization in industrial environments using a ring of cameras and aruco markers. pages 1–6, 2021. doi: 10.1109/IECON48115.2021.9589442. URL https://ieeexplore.ieee.org/abstract/document/9589442.

[47] Mohammad Fattahi Sani and Ghader Karimian. Automatic navigation and landing of an indoor ar. drone quadrotor using aruco marker and inertial sensors. In *2017 International Conference on Computer and Drone Applications (IConDA)*, pages 102–107, 2017. doi: 10.1109/ICONDA.2017.8270408. URL https://ieeexplore.ieee.org/document/8270408.

[48] Shubham Sharma, Shubhankar Verma, Mohit Kumar, and Lavanya Sharma. Use of motion capture in 3d animation: Motion capture systems, challenges, and recent trends. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pages 289–294, 2019. doi: 10.1109/COMITCon.2019.8862448. URL https://ieeexplore.ieee.org/document/8862448.

[49] StereoLabs. Zed sdk 4 initparameters. URL https://www.stereolabs.com/docs/api/structsl_1_1InitParameters.html.

[50] Stereolabs. Coordinate frames. URL https://www.stereolabs.com/docs/positional-tracking/coordinate-frames.

[51] StereoLabs. stereolabs/zed-matlab github repository. URL https://github.com/stereolabs/zed-matlab.

[52] Kai Sun, Wenhui Chen, Cheng Guo, and Lu Sun. Interactive solution for man-machine system based on mems sensors. pages 1741–1745, 2016. doi: 10.1109/CGNCC.2016.7829053. URL https://ieeexplore.ieee.org/document/7829053.

[53] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1): 32–46, 1985. ISSN 0734-189X. doi: https://doi.org/10.1016/0734-189X(85)90016-7. URL https://www.sciencedirect.com/science/article/pii/0734189X85900167.

[54] Gabriel Taubin. 3d rotations. *IEEE Computer Graphics and Applications*, 31(6):84–89, 2011. doi: 10.1109/MCG.2011.92. URL https://ieeexplore.ieee.org/document/6056514.

[55] Dominic Thewlis, Chris Bishop, Nathan Daniell, and Gunther Paul. Next-generation low-cost motion capture systems can provide comparable spatial accuracy to high-end systems. *Journal of Applied Biomechanics*, 29(1):112 – 117, 2013. doi: 10.1123/jab.29.1.112. URL https://journals.humankinetics.com/view/journals/jab/29/1/article-p112.xml.

[56] CO3 Group UPV. Simulink czep aruco nano. URL https://github.com/CO3-UPV/Simulink_CZEP_Aruco_Nano/tree/main.

[57] Mars Caroline Wibowo, Sarwo Nugroho, and Agus Wibowo. The use of motion capture technology in 3d animation. *International Journal of Computing and Digital Systems*, 15(1):975–987, 2024. URL https://journals.uob.edu.bh/handle/123456789/5222.

[58] Yu-Jin Zhang. *Camera Calibration*, pages 37–65. Springer Nature Singapore, Singapore, 2023. ISBN 978-981-19-7580-6. doi: 10.1007/978-981-19-7580-6_2. URL https://doi.org/10.1007/978-981-19-7580-6_2.