UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI **INGEGNERIA**

Corso di Laurea triennale in **INGEGNERIA MECCANICA**

PROGETTO E SVILUPPO DI UN SISTEMA PER L'ANALISI DEL MOVIMENTO TRIDIMENSIONALE DI ATLETI BASATO SU RETI NEURALI

DESIGN AND DEVELOPMENT OF A SYSTEM FOR THE ANALYSIS OF THE THREE-DIMENSIONAL MOVEMENT OF ATHLETES BASED ON NEURAL NETWORKS

Relatore:

Prof. **Paolo Castellini**

Tesi di Laurea di:

Baldassarri Davide

A.A. **2021 /2022**

Indice

Capitolo 1

Introduzione

Il Capture Motion, ossia il processo di acquisizione dei movimenti umani, si è dimostrato negli ultimi anni sempre più importante e centrale in vari campi, dall'intrattenimento allo sportivo, dalla sorveglianza alla robotica, al punto da sviluppare numerose tecnologie con lo scopo di rilevare gesti, acquisire dati di velocità e accelerazione di parti, produrre modelli virtuali.

In questa trattazione si esporranno i metodi adottati e i risultati ottenuti per il rilevamento di un gesto atletico, in particolare di nuoto in acqua, tramite un sistema di visione stereo di camere RGB, con l'obiettivo di allestire un processo che permetta un'acquisizione tramite un numero arbitrario di camere, così da limitare i problemi di inquadratura dati dal campo visivo delle stesse e creare un modello virtuale dell'atleta per una successiva analisi dei movimenti.
La scelta del metodo di acquisizione è stata vincolata dalla natura stessa del movimento che, essendo acquatico, ha impedito l'utilizzo di vare tecniche di Capture Motion, tra cui camere a infrarossi, marcatori passivi o attivi.
Il processo di analisi software è stato affidato ad un sistema di reti neurali del programma MediaPipe, che rileva 33 punti di tracking nel video in input e fornisce le loro coordinate in output. L'elaborazione dei dati è avvenuta tramite il programma MatLab con lo sviluppo di uno script che permetta la visualizzazione del modello 3D dell'atleta e delle camere, con particolare focalizzazione al caso bicamera, dove è stata eseguita un'analisi delle velocità e accelerazioni di un generico punto di tracciamento.

Per lo sviluppo del progetto sono state utilizzate dapprima due comuni camere Reflex, che hanno permesso un iniziale approccio semplificato al problema, e solo successivamente un sistema con più camere, con cui è stata esaminata l'influenza di vari parametri nella visualizzazione finale del modello, come la posizione relativa delle macchine, la sincronizzazione, l'orientazione rispetto al soggetto, le condizioni di luce, la natura del movimento.

1.1 Cenni di Teoria

Si introduce un semplice richiamo al modello Pinhole, sistema di visione teorico in cui un generico punto dello spazio viene acquisito attraverso una camera con un unico foro infinitesimo e poi proiettato su un piano, generando l'immagine. Questo servirà per descrive quali parametri vengono ottenuti tramite la calibrazione e le equazioni di proiezione prospettica su cui si basano le acquisizioni.

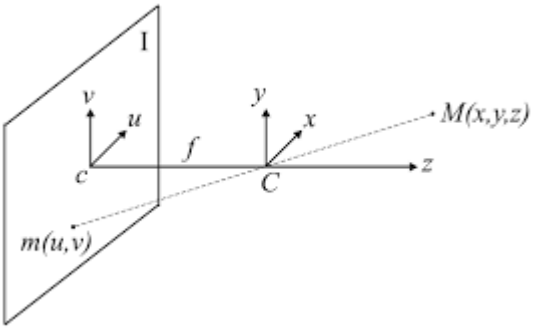


Figura 1.1: Modello pinhole

Con riferimento a *Figura 1.1* chiamiamo con ***I*** il *piano immagine* che si trova a distanza ***f***, *lunghezza focale*, dal *piano focale*, cioè quello su cui giace il *sistema di riferimento mondo* ***{x,y,z}*** di centro ***C***, *centro ottico.*

Come si può notare, un punto nel mondo 3D avrà una terna di coordinate M(x,y,z) mentre nel sistema di riferimento 2D della camera di centro ***c*** ne avrà due, *{u,v}*. Questo permette di scrivere le equazioni di proiezione prospettica:

$$\begin{cases} u = -x\,\frac{f}{z} \\ v = -y\frac{f}{z} \end{cases} \qquad (1.1)$$

Di fatto si trovano due sole equazioni, in quanto la terza equivarrebbe alla distanza focale ($w=f$) che si mantiene costante; si noti inoltre che la (1.1) a causa di $1/z$ è non lineare e non mantiene le distanze tra i punti e gli angoli tra le linee.

Per poter sfruttare questo in una applicazione di Computer Vision sono necessarie quindi equazioni che leghino il mondo 3D a quello 2D e per farlo devono essere fatte due assunzioni: la prima è che il sistema di rifermento della camera sia posizionato rispetto al sistema di riferimento mondo, e la seconda è che essendo le coordinate pixel dell'immagine le uniche direttamente rilevabili, tramite queste devono essere ottenute quelle dei punti dell'immagine nel sistema di riferimento camera. [1]

Questo implica perciò la conoscenza di due parametri, divisibili tra estrinseci ed intrinseci, ora illustrati e affrontati da un punto di vista matematico:

1. *Parametri estrinseci*: sono quei set parametri che consentono di eseguire un cambio di coordinate tra i due sistemi di riferimento descritti, definendo univocamente la posizione e l'orientamento del sistema camera a noi sconosciuto rispetto al sistema mondo a noi noto. La calibrazione permette appunto di ottenere, esclusivamente tramite le immagini catturate, dati sull'orientazione del sistema camera, che generalmente sono composti da un vettore traslazione T, avente tre dimensioni, che descrive la posizione relativa delle origini nei due sistemi di riferimento, e una matrice di rotazione R, 3x3 ortogonale, che riallinea gli assi.
2. *Parametri intrinseci*: sono quei set di parametri che dipendono dalle specifiche tecniche ed hardware della macchina, tra cui le caratteristiche ottiche, geometriche e digitali. Permettono inoltre di collegare le coordinate pixel di un'immagine con le corrispondenti nel sistema di riferimento della camera. Anche in questo caso tramite la calibrazione si ottengono questi dati, elencabili come la lunghezza focale (f [mm]), la distorsione geometrica dovuta all'ottica della camera (k1,k2 [numeri puri]), , la trasformazione tra coordinate camera e pixel (ox,oy [pixel]) e l'unità di misura del mondo 3D riportata al piano immagine (sx,sy [mm]).

Per una trattazione più approfondita consultare [1], [2], dove viene trattato il modello generale con lenti focalizzanti e la relativa distorsione geometrica (cosa che non riguarda questo studio in quanto le acquisizioni eseguite tendono a non produrre una distorsione rilevante).

Alcune delle informazioni necessarie per lo studio del video acquisito devono essere per cui ottenute tramite un preliminare processo di calibrazione delle camere, singolo nel caso di un sistema ad una sola camera ma anche stereo se abbiamo invece una applicazione stereoscopica.
Chiameremo d'ora in poi il processo di calibrazione fatto rispetto ad una sola camera *Camera Calibration*, mentre quello rispetto a più camere *Stereo Camera Calibration*.
Come già illustrato i dati che si ottengono da questo processo sono i set di parametri intrinseci ed estrinseci, alla quale si aggiungono i parametri estrinseci relativi nel caso di

un sistema di visione stereo, che comprendono la posizione e l'orientamento relativo tra le due camere (l'una rispetto l'altra).

Sono diversi i metodi con la quale si può eseguire una calibrazione, come il processo tradizionale, a visione attiva o ad auto calibrazione, ma in particolare si illustra il metodo classico di Zhang.

Questa tipologia è molto utilizzata, in particolare anche dal programma di calibrazione di Matlab, poiché è possibile ottenere dati accurati con un sistema semplice basato sull'omografia di pattern di dimensioni e forma conosciute che agisce da riferimento. In generale questo elemento planare può essere di varie geometrie, ma il più utilizzato è sicuramente il checkboard (o scacchiera) che, quando viene identificato nei vari frame acquisiti, l'algoritmo posiziona il sistema di riferimento nel vertice del quadrato in alto a sinistra, definendo così il piano di riferimento z=0 come in *Figura 1.2*. Tramite la corrispondenza dei vari punti il sistema definisce i dati estrinseci ed intrinseci, ma affinché questo sia possibile sono necessarie varie acquisizioni e che il pattern assuma diverse orientazioni e posizioni.[2][3]

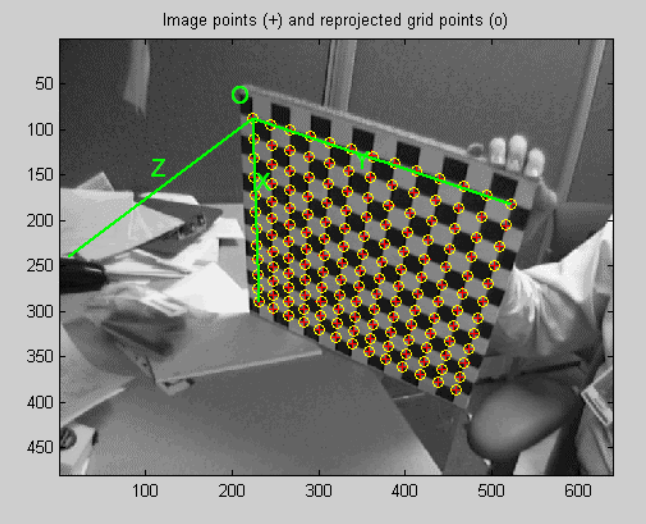


Figura 1.2: Identificazione vertice di calibrazione

Per motivare l'utilizzo di un sistema stereoscopico rispetto ad altre soluzioni (e perché in genere la sua applicazione si rende necessaria) si consideri che l'uso di una sola camera RGB per l'acquisizione del movimento non è sufficiente a rilevare una misura di profondità.

Questa deve essere per cui affiancata ad altri sistemi per il rilevamento della distanza, che possono essere divisi in due tipi: quelli che associano ad una camera monoculare

altri sensori di rilevamento (vedi *1.2.1 Microsoft Kinect System e Asus Xtion*), o appunto un sistema stereo di coppie di camere, chiamato anche sistema a "disparità".
Considerando un sistema di due camere con uguale lunghezza focale (*f*) e assi ottici paralleli (interasse *b*), questo termine è legato alla differenza di posizione di un ipotetico punto proiettato da entrambe le macchine (detto appunto disparità *d*) con la quale è anche possibile calcolare matematicamente la distanza del soggetto:

$$Z = \frac{b \cdot f}{d} \qquad (1.2)$$

L'adozione di un sistema stereoscopico è particolarmente vantaggiosa in quanto permette di utilizzare un hardware semplice ed economico senza parti mobili (aumentando quindi l'affidabilità del sistema), non necessita di molte informazioni preacquisizione e in output fornisce immagini/video a colori (o monocromatiche), che sono utilizzabili nella maggior parte delle applicazioni. [4], [5]

1.2 Stato dell'arte

Le principali alternative attualmente disponibili ai sistemi stereoscopici nel campo del Capture Motion con utilizzo di camere possono essere ricondotte a modelli con sensori TOF (time of flight) ossia a tempo di volo (vedi "*1.2.1 Microsoft Kinect System e Asus Xtion"*), e a tecnologie di realtà aumentata, che non verranno però trattate poiché essendo ancora in fase di sviluppo non sono confrontabili dal punto di vista di precisione ed affidabilità con le precedenti soluzioni.
Per completezza si presentano brevemente alcuni esempi di rilevamento del movimento non qualitativo "*1.2.2 Metodi alternativi di rilevamento"* che, non permettendo la creazione di skeleton, impediscono lo studio dei gesti e della cinematica del corpo. Sono quindi applicabili maggiormente in altri campi rispetto a quello preso in esame.

1.2.1 Microsoft Kinect System e Asus Xtion

Il Kinect è uno dispositivo RGB-D che fornisce in uscita immagini a colori e di profondità sincronizzate, che possono essere usate per la ricostruzione di punti 3D o riconoscimento del contenuto di un'immagine.
Inizialmente sviluppato da Microsoft per la console di gioco Xbox con lo scopo di rilevare movimenti umani senza l'ausilio di controller, ha trovato impiego in numerosi altri campi, aprendo la strada a nuove applicazioni nel Computer Vision essendo più economico e rapido delle tradizionali soluzioni (come telecamere stereo).[6]

La parte hardware è visualizzabile in *Figura 1.3* ed è composta da:

- una telecamera RGB con 640x480 a 30 fps;
- un sensore di profondità (IR) composto da un emettitore e una camera ad infrarossi;
- un array di quattro microfoni;
- un motore per la regolazione dell'inclinazione (tilt motor).

La camera RGB ha lo scopo di acquisire immagini a colori come principale dato di input al fine di tracciare i movimenti del soggetto inquadrato.
L'emettitore (IR) proietta un motivo a puntini (unici) nella banda dell'infrarosso sull'area di inquadratura che verranno quindi captati solamente dalla camera IR corrispondente (e non dalla camera RGB).

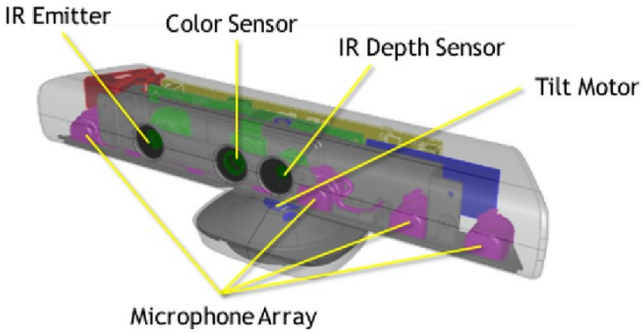


Figura 1.3: Hardware kinect

La misurazione della distanza si basa sul confronto tra la posizione relativa di coppie di punti, nel pattern di proiezione e sull'oggetto fisico. Tramite una calibrazione off-line si ottiene una relazione geometrica tra il pattern proiettato e quello acquisito dalla camera

IR, che permette quindi di mappare la profondità dello spazio inquadrato senza l'utilizzo di un sistema di camere stereo. [6]
L'array di microfoni permette di ottenere dati audio sincronizzati alle immagini attuando contemporaneamente un processo di eliminazione di rumore e riverbero, al fine di sviluppare un sistema di riconoscimento vocale. Uno studio a riguardo è trattato in [7].
Dal punto di vista software sono a disposizione del sistema varie librerie, alcune Open Source e altre sviluppate dalla stessa Microsoft, che comprendono algoritmi di gestione e processing.
Tra queste troviamo OpenNI, che per eseguire il tracker dello skeleton richiede di mantenere una calibrazione iniziale finché il sistema non riconosce abbastanza punti di ancoraggio e articolazioni; Microsoft NDK non richiede un'iniziale calibrazione ma è più incline a errori di riconoscimento, soprattutto quando la posa iniziale è particolarmente complessa.
La principale differenza tra le due sta nella velocità di esecuzione, che dipende maggiormente dalle condizioni ambientali e dalla potenza di calcolo, e nella modalità di riconoscimento, che in OpenNI si concentra sulla mano, mentre in Microsoft NDK nel riconoscere un gesto elementare.
Con la seconda soluzione, inoltre, è possibile riconoscere solo una porzione del corpo (ad esempio la parte superiore), molto utile se si sta inquadrando un soggetto seduto. [6]
Per una trattazione più approfondita riferirsi a [6]–[8].

Similarmente troviamo Asus Xtion, che sfrutta la medesima tecnologia della soluzione precedente per il rilevamento della profondità e del soggetto:
un sistema di emettitori ad infrarossi proietta un pattern sul campo che, grazie alla distorsione dello stesso, permette di risalire alle misure geometriche e di forma del soggetto.
Entrambe le tecnologie si basano sul brevetto di PrimeSense, (ora azienda affiliata di Apple.inc), e sono quindi paragonabili in termini di errore e affidabilità.[9]
Questi sistemi possono essere ricondotti alla tipologia TOF "Time of Flight", che basano la misurazione della distanza su emettitori di luce LED o infrarosso proiettati sull'oggetto e poi riacquisite da un rilevatore dopo che questa è stata rifratta dal corpo. Il tempo di volo permette una misurazione accurata della distanza e lo sviluppo di strumenti per la rilevazione della profondità a basso costo.

1.2.2 Metodi alternativi di rilevamento

Nel campo delle acquisizioni del movimento sono diversi i metodi con la quale è possibile rilevare lo stesso, alcuni dei quali con obiettivo la semplice individuazione di presenza o meno del movimento in un dato video o spazio. Queste soluzioni essendo più inclini al puro rilevamento quantitativo del movimento sono meglio applicabili quindi a sistemi di sicurezza e identificazione di oggetti.

Un esempio particolarmente significativo è il metodo con template temporale, in cui il movimento viene rilevato direttamente dal video e non dopo essere stato ricostruito tridimensionalmente. Per prima viene costruita un'immagine binaria di energia-movimento "MEI" che rappresenta dove è avvenuto il movimento in una sequenza di immagini, successivamente viene creata un'immagine che descrive la cronologia del movimento MHI. Presi insieme questi due elementi possono essere considerati come i componenti di un template temporale, in cui ogni pixel viene "riempito" (*Figura 1.4*) quando riceve dati. [10]

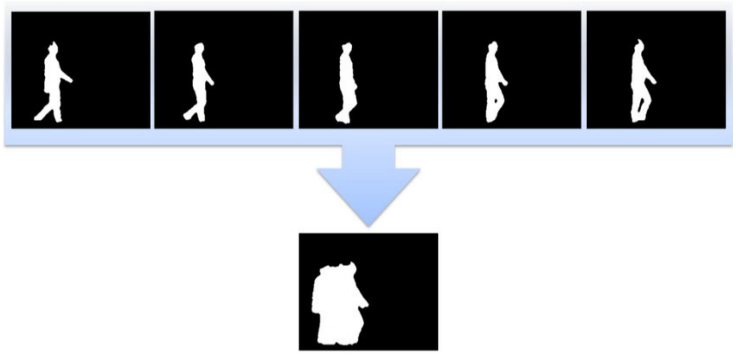


Figura 1.4: Template temporale

Capitolo 2

Material

In questo capitolo saranno discussi i sistemi hardware, in particolare camere, sostegni e cavalletti, sistema di sincronizzazione, illuminazione, checkboard di calibrazione, e quelli software, ovvero Matlab, Spider (e un accenno a DaVinci), utilizzati per l'esecuzione della prova.

2.1 Hardware

Le macchine utilizzate per l'iniziale studio bicamera sono state due comuni reflex Nikon, diverse nel modello e nelle ottiche ma comunque confrontabili in prestazioni e funzionalità. I parametri come l'apertura del diaframma e l'ISO sono stati impostati in modo da garantire una quanto maggiore chiarezza nel video e un campo di messa a fuoco tale da permettere i movimenti di prova.
Le camere sono state settate in modalità manuale, messa a fuoco compresa, così da garantire il mantenimento di questi parametri in tutti i video registrati. Essenziale è l'impostazione della frequenza di acquisizione dei frames per secondo dei video, che deve essere necessariamente uguale per tutte le camere. Nel nostro caso si è adottata una classica soluzione a 30 fps.
Due sono gli aspetti che hanno maggiormente vincolato inizialmente il progetto: la necessità di un supporto per le camere che fosse abbastanza ampio da sostenerle e quanto più piano, e l'esigenza di sincronizzare le macchine nella registrazione.
Per il supporto si è deciso di assemblare una semplice impalcatura di *Figura 2.1* composta da un profilato di alluminio con sezione standard sorretta da due aste e bloccata tramite delle staffe a cavallotto. Su questa sono state montate due piastrine con fori che per il serraggio delle camere tramite viti ottiche filettate.

La sincronizzazione di più camere può essere effettuata in vari modi, alcuni dei quali però inattuabili se si necessita di una particolare precisione: in dettaglio le tecniche di sincronizzazione con scatto wireless non sono adottabili a causa della loro natura di funzionamento basata su onde radio emesse in modo discontinuo e ad intermittenza.

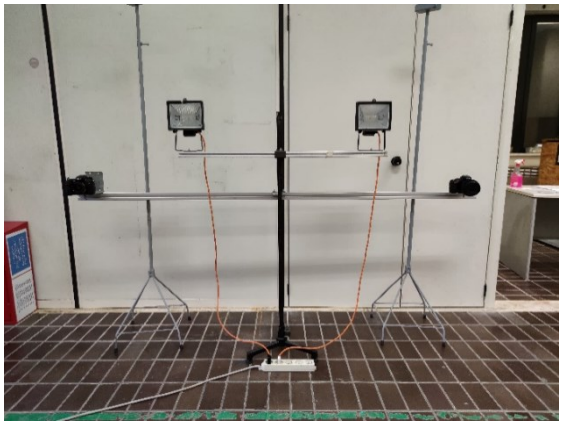


Figura 2.1: Supporto camere e comparto luci

Una valida soluzione può essere l'utilizzo di un rumore bianco durante la ripresa che, essendo registrato contemporaneamente da entrambe le camere, segna l'inizio dell'acquisizione e permette un successivo taglio del video.
In definitiva si è però scelto di adottare un sistema che sfrutti la porta del telecomando di scatto presente nelle camere, garantendo precisione nella sincronizzazione con una soluzione semplice e comoda.

Figura 2.2: Pulsante di scatto (sinistra), circuito interno (destra)

L'apparato è composto da due cavi di scatto (uno per camera) formati da un'estremità con jack da 3.5 mm e l'altra dal terminale caratteristico dei sistemi di scatto remoto Nikon.
In particolare, il jack da 3.5 mm è composto da tre canali, la massa, lo scatto e la messa a fuoco che, se messi in cortocircuito, permettono di inviare un segnale di scatto alla macchina. In una semplice scatola (*Figura 2.2*) sono state alloggiate le prese in cui inserire i jack e un pulsante collegato ad un basilare circuito che attua il cortocircuito tra tutti i cavi in entrata, garantendo quindi una sincronizzazione dei segnali in uscita.
Dall'analisi dei dati acquisiti da un semplice test si è osservato che l'assoluta sovrapposizione dello scatto non è stata raggiunta. Malgrado questo si è comunque

ritenuto accettabile il risultato poiché questo ricade in un range di errore limitato e rimuovibile tramite software (la differenza media in un video con circa 2000 frame è dell'ordine di 9-10 immagini).

La convinzione alla base dell'utilizzo di un comparto luci aggiuntivo, oltre a quelle ambientali, è che un'illuminazione abbondante avrebbe permesso di migliorare vari parametri delle macchine, quali la sensibilità ISO e l'apertura del diaframma, così da diminuire il tempo di integrazione della camera ed ampliare il campo di messa a fuoco dell'ottica, garantendo quindi una maggiore chiarezza dei dati in uscita. Sono state quindi utilizzate dapprima due lampade a neon di limitata potenza, mentre successivamente si è scelto di adottare un sistema di due lampade a incandescenza di potenza superiore, posizionate su un cavalletto.
Anche in questo caso dei semplici test di prova hanno evidenziato come l'utilizzo del comparto luci aggiuntivo non fosse particolarmente necessario con un'illuminazione ambientale già sufficiente, ma comunque vantaggioso per i motivi sopracitati.

Il sistema di calibrazione utilizzato dal sistema di MatLab è basato sull'algoritmo di Zhang quindi per l'esecuzione necessita di pattern fisico planare. Si è scelto di utilizzare un classico checkboard fissato tramite puntine (per evitare grinze della carta dovute a colla o nastro adesivo) su un piano di polistirolo, come quello in *Figura 2.3*, avente 7x10 rettangoli ognuno dei quali misura 62x62 mm.

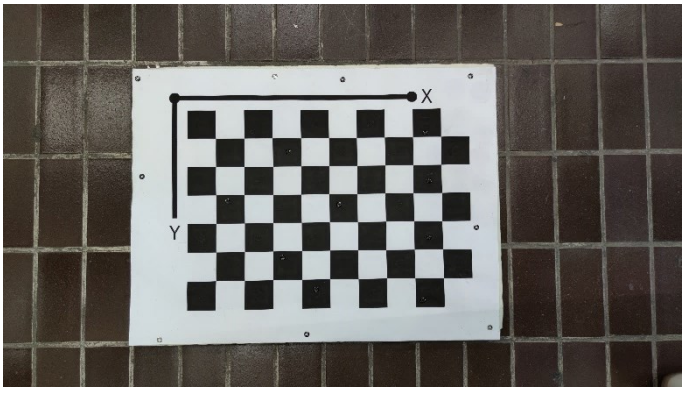


Figura 2.3: Checkboard di calibrazione

Per il successivo rilevamento a tre camere gli hardware utilizzati sono stati i medesimi, con l'implementazione di una camera Canon di specifiche confrontabili alle altre e una serie di cavalletti aggiuntivi come mostrato in *Figura 2.4*. Nella terza camera il processo

di acquisizione è avvenuto però in modo manuale, in quando non è stato possibile reperire altri cavi per il sistema di scatto sincronizzato. Questo metodo è stato giudicato accettabile solo dopo una verifica dei dati acquisiti, che non risultavano troppo compromessi e discordanti in fatto di numero di frames rispetto le altre camere.
Il comparto luci e il pattern rimangono invariati.

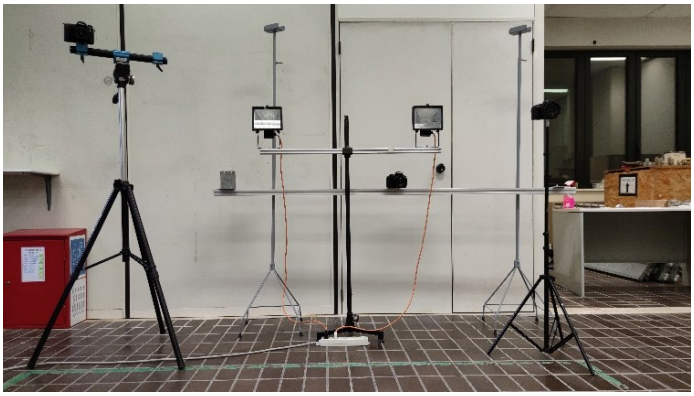


Figura 2.4: Sistema a tre camere

2.2 Software

Il software principalmente utilizzato è stato Matlab, abbreviazione di Matrix Laboratory, che lavorando con il linguaggio matematico (principalmente matriciale) permette di analizzare i dati acquisiti, eseguire calcoli numerici e visualizzare graficamente i risultati.
Tra i diversi pacchetti disponibili nella libreria del software sono presenti anche strumenti di calibrazione monocamera e stereoscopica con la quale sono stati ottenuti i parametri estrinseci ed intrinseci.
Una volta ottenuti i dati di base tramite le acquisizioni, i restati dati di input necessari per l'elaborazione dello script automatizzato che esegue la ricostruzione grafica del movimento sono stati ottenuti tramite altri script a loro volta, tra cui citiamo come esempio quello per l'ottenimento dei frames dei video, del caricamento dei dati e della triangolazione degli stessi.
Nel caso bicamera in output sono stati forniti anche i grafici di velocità e accelerazione di uno dei generici punti di ancoraggio al corpo in funzione dei frames, allo scopo di verificare i dati elaborati e valutarne il processo di filtraggio degli stessi, ma anche per una successiva analisi critica dei movimenti effettuati dal soggetto.

Un altro software fondamentale per lo studio dei dati, in particolare per l'elaborazione dei movimenti nei video, è stato Mediapipe, piattaforma OpenSource di Google (in questo caso utilizzata con sistema operativo Linux) che ha permesso di applicare la teoria del deep learning per l'acquisizione e l'assegnazione dei punti di tracking del corpo nelle registrazioni.

Il programma consiste in una rete neurale scritta nel linguaggio di programmazione Python che consente di eseguire rilevamento facciale, di posa, tracciamento di oggetti e diverse altre analisi legate al machine learning. In generale Mediapipe, così come tutte le reti neurali, basa la sua capacità di identificazione e riconoscimento di oggetti su un processo di apprendimento automatico tramite pattern di base presenti in un archivio di dati fornito in fase di sviluppo. La macchina, in questo modo, riesce autonomamente a confrontare i dati in suo possesso con quelli fornitigli in input, così da identificare eventuali riscontri ed estrarre i diversi punti necessari alla ricostruzione del modello. In *Figura 2.5* si possono notare i diversi target identificabili dal programma utilizzato, ognuno associato ad un numero e una nomenclatura.

Per una trattazione più approfondita del Deep/Machine learning consultare [11], [12]

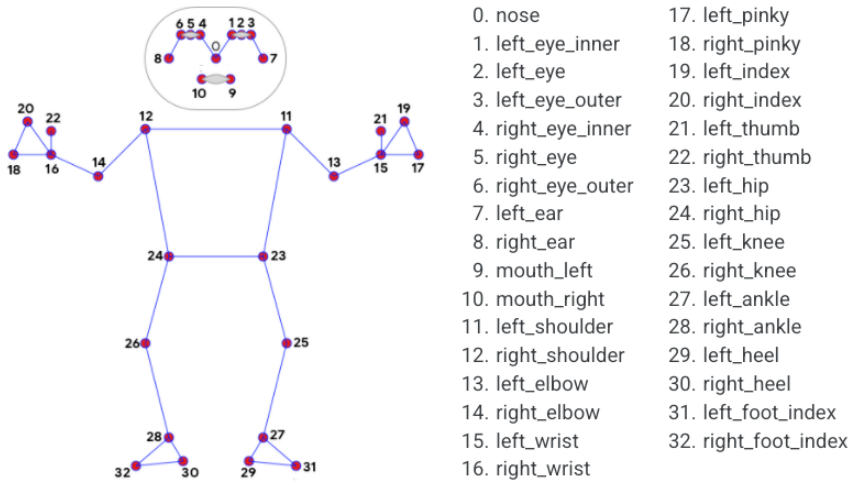


Figura 2.5: Punti di treacking di Mediapipe e i rispettivi link

Una volta acquisiti i video tramite le camere, questi vengono inseriti nel programma Spider, ossia un generico ambiente di sviluppo integrato di cui ci si avvale per l'utilizzo della rete neurale, che tramite quest'ultima identifica 33 parti del corpo, da cui estrapola le coordinate pixel {x,y,z} per ogni frame (che inserisce in un file .txt) e costruisce un video "skeleton" (*Figura 2.6)*.

Questo video consiste in una sovraimpressione nella registrazione iniziale dei diversi punti del corpo rilevati che una volta uniti tramite segmenti creano uno “scheletro” virtuale del soggetto in movimento. Durante lo studio questa tipologia di video si è rivelato molto utile in quanto permette una prima analisi critica sulla buona riuscita o meno dell’acquisizione, andando a confrontare visivamente i movimenti realmente eseguiti con quelli captati dal programma.

Figura 2.6: Skeleton rilevato sul soggetto

Infine, anche se utilizzato in minori occasioni, si cita l’utilizzo del programma DaVinci con la quale, in caso di necessità, è stata eseguita una preliminare elaborazione delle registrazioni acquisite al fine di uniformare l’estensione dei file video.

Capitolo 3

Method

In questo terzo capitolo saranno illustrati i metodi di sviluppo dei dati e le fasi di scrittura e funzionamento dello script (reso automatico) per la visualizzazione dei risultati. Diverse informazioni raccolte in questa sezione sono state ottenute tramite l'utilizzo della sezione Help di Matlab [13].

3.1 Calibrazione

In primis si riporta il processo di calibrazione nel caso base con due camere per poi generalizzare il discorso alla condizione con un numero qualsiasi di macchine.

3.1.1 Condizione stereo camere

Per prima viene eseguita l'acquisizione dei video di calibrazione, quindi le camere vengono montate sul supporto con asta all'incirca a 2,5 metri l'una dall'altra cercando di mantenere come regola generale che l'interasse tra esse misuri quanto la distanza tra la camera di riferimento e il soggetto in movimento.

Per camera di riferimento, che ora chiameremo *Camera 1*, intendiamo una delle due in modo arbitrario che sarà considerata come origine del sistema di riferimento stereoscopico. Nel momento in cui le camere disponibili fossero differenti, magari aventi obiettivi con ampiezze focali diverse, è opportuno posizionare quella con apertura minore come *Camera 1*. Infatti, disponendo la macchina con angolo di visione più grande ai lati si garantisce l'acquisizione di tutto il campo di movimento rispetto alla soluzione contraria.

La posizione relativa delle camere rispetto il soggetto non è vincolante per l'utilizzo di questo sistema: la disposizione può essere assolutamente arbitraria anche se è raccomandato direzionare tutte le camere verso il soggetto. Questo aspetto può sembrare ovvio ma il suo mancato adempimento, come l'adozione della disposizione di *Figura 3.1*, può portare a false catture da parte del programma di rete neurale.

Per un approfondimento del tema documentarsi al Paragrafo 4.1 *"Condizione stereo camere"*.

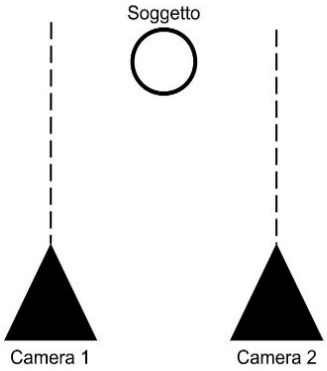


Figura 3.1: Caso di assi ottici delle camere paralleli tra loro.

Una volta preparato il setup delle camere e del sistema di sincronizzazione, il processo di acquisizione dei video di calibrazione consiste nella registrazione dello spostamento del checkboard (mosso dall'operatore), al fine di ottenere coppie di frames delle due camere con il pattern nella medesima posizione.
Affinché le immagini estrapolate dai video siano perfettamente a fuoco e non mosse, le camere devono essere impostante in modo manuale (MF) e il movimento della scacchiera deve essere lento e costante; inoltre, per poter assicurare una corrispondenza di posizione e orientamento del pattern nelle coppie di frames, deve essere garantita la sincronizzazione dei video (e banalmente il nome delle coppie di immagini).
Lo spostamento del pattern deve coprire una zona quanto più ampia possibile nello spazio inquadrato, in particolare nell'area di movimento del soggetto, e deve assicurare un certo numero minimo di coppie di immagini.

In *Figura 3.2* sono mostrate le coppie di immagini sincronizzate appartenenti alle due camere.

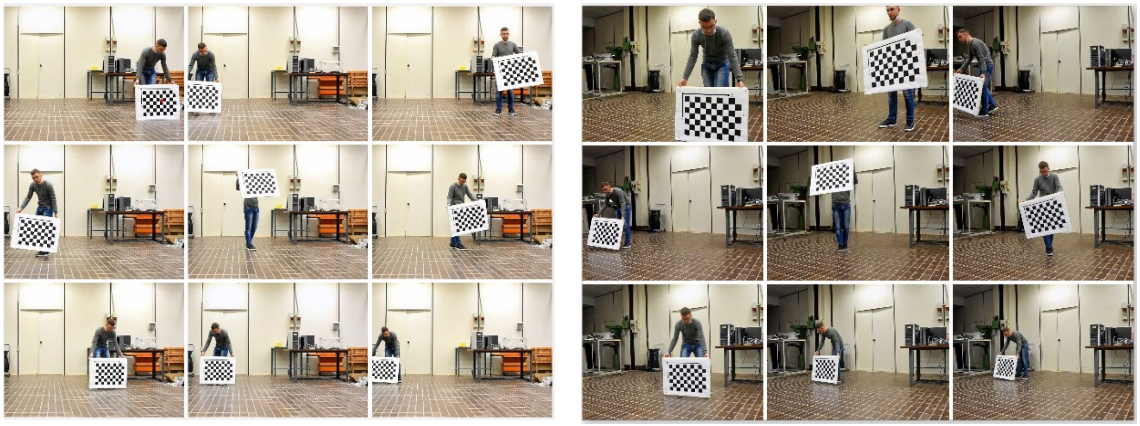


Figura 3.2: Frames della Camera 1 (sinistra) e della Camera 2 (destra) con camere convergenti

Ottenuti i video di calibrazione, per estrapolare le immagini da essi si esegue uno script costruito in Matlab che sfrutta la funzione: `imwrite(A,filename)`
Dato un certo elemento A, che equivale al video caricato, e la cartella di destinazione con l'estensione nominale e il formato, il codice recupera tutti i frame del dato input.

```
vid12= VideoReader('calib_12.MOV');
numFrames=vid12.NumFrames;
n12=numFrames;

numFrames12=min(n12,n21);

for i=1:P:numFrames12;
    frames=read(vid12,i);
    imwrite(frames,['*percorso cartella desiderata*' int2str(i),'.jpg']);
end
```

Nel primo blocco si caricano i video di calibrazione (uno ad uno) rinominati in modo analogo così da mantenere coerenza (ad esempio *"calib"* seguito da numeri dove il primo pedice indica la camera di riferimento e il secondo l'altra). Una volta estratto il numero di frames (*numFrames*) di ogni file si esegue una comparazione così da determinarne il valore minimo, che sarà quello di riferimento per tutti i video. Come già specificato, il processo di sincronizzazione anche se automatico non garantisce un identico numero di frames registrati tra le camere poiché entrano in gioco fattori quali la velocità software della macchina e di scrittura della scheda SD. Per ovviare al problema si eliminano i frame finali dei video più lunghi così da omologarne il numero.
Nel terzo blocco è inserito un ciclo *for* che va da 1 al *numFrames* minimo, con un passo *P* che andrà a recuperare una foto ogni *P* frame saltati. Questo valore può essere scelto arbitrariamente ma prove sperimentali (e la stessa pagina descrittiva del tool sul sito Web di Matlab) rivelano che per una buona calibrazione è necessario un numero di immagini all'incirca pari a 100 (quindi un video di 2 minuti con 25 fps dovrà avere passo a 30).
Il ciclo recupera le immagini che poi salva nella cartella definita dal percorso scelto e le rinomina seguendo un ordine numerico, in modo che i frames di tutti i video abbiano nomi identici ma si trovino in cartelle differenti. Questo è fondamentale poiché lo strumento di calibrazione di Matlab associa le immagini con lo stesso nome considerandole scattate allo stesso istante su entrambe le camere.

A questo punto è possibile effettuare le mono calibrazioni di ogni singola camera: una volta aperto il tool *"Camera Calibration"* di Matlab si caricano le immagini e inseriscono i dati relativi al tipo di pattern, le sue misure ed eventuali distorsioni. Terminato il processo vengono visualizzate due finestre (quelle di *Figura 3.3*), una con la rappresentazione tridimensionale della camera con i diversi pattern rilevati, e l'altra con un grafico degli errori medi (in pixel) in funzione dei frames. In questa fase è possibile eliminare alcune immagini con associato un errore elevato al fine di migliorare i dati in output (anche se questo nella mono calibrazione non è in genere necessario).
Una volta esportati e salvati i risultati si ottiene un file *.mat* contenente i parametri itrinseci ed estrinseci della camera (*RadialDistortion, TangentialDistortion, TranslationVectors, RotationVectors, ImageSize, Skew, FocalLength, PrincipalPoint* per citarne alcuni).

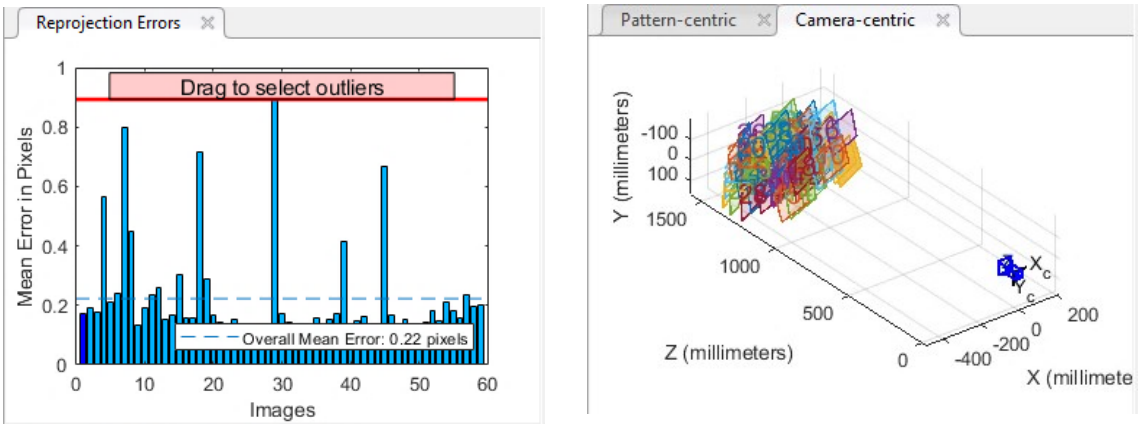


Figura 3.3: Grafico dell'errore medio (sinistra) e visualizzazione 3D dei pattern con camera (destra) nel tool di mono calibrazione

Successivamente si eseguono le stereo calibrazioni: aperto il tool *"Stereo Camera Calibration"*, si caricano le immagini di entrambe le acquisizioni, considerando che il software pone come origine del sistema di riferimento la seconda camera inserita. Nei dati in output sono presenti i termini *TranslationOfCamera2* e *RotationOfCamera2* che equivalgono ai parametri estrinseci relativi, ossia la distanza e l'orientazione della prima camera caricata rispetto la seconda.
Anche in questo caso le due finestre di *Figura 3.4* permettono di valutare la posizione relativa tra quest'ultimi e le camere e l'errore medio dei vari pattern, e di abbassarlo. Quando viene diminuito il limite massimo dell'errore il software elimina tutte le coppie di frames nella quale il riscontro del modello ha evidenziato un errore superiore a quello impostato. Questo è il principale motivo per cui è fondamentale garantire un numero

minimo di coppie di frames. Il file *.mat* in output contiene, oltre ai parametri citati prima, anche i singoli parametri intrinseci delle due camere alla quale il software riesce a risalire pur non essendo pensato per mono calibrazioni. Data la bassa precisione di questi dati si tende a scartarli e ad usare quelli ottenute tramite le calibrazioni precedenti.

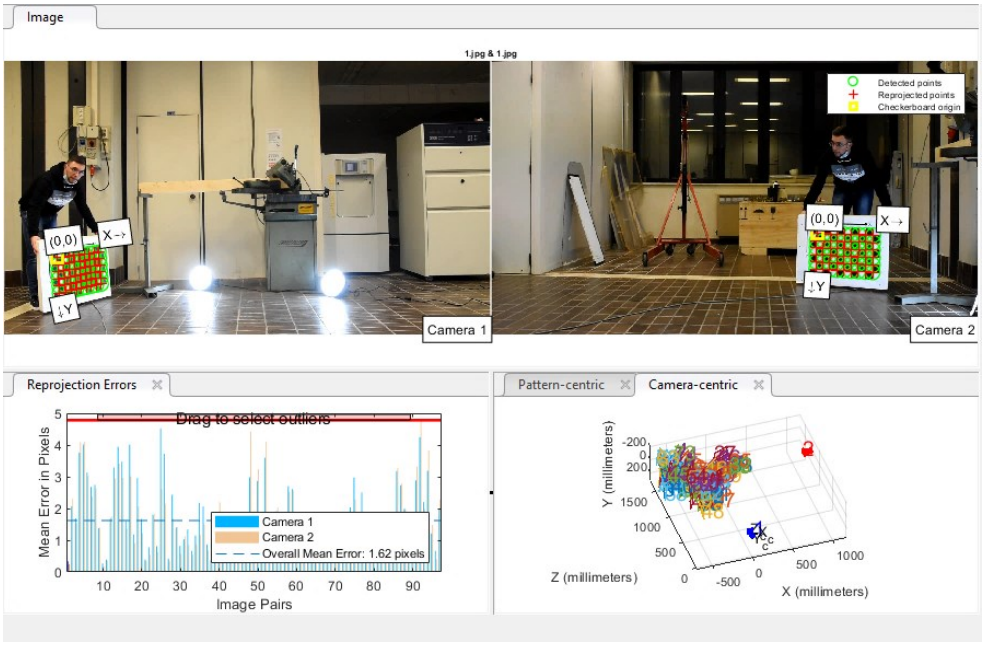


Figura 3.4: Coppia di frame identificate (sopra) e finestra dell'errore (sotto) nel tool di stereo calibrazione

3.1.2 Condizione n-camere

Il metodo di calibrazione di un numero *n* qualsiasi di camere è un'estensione del caso bicamera poiché il processo coinvolge volta per volta una coppia di camere. Essendo il tool di calibrazione di Matlab basato sul modello di Zhang, concepito per l'elaborazione dei dati di pattern inquadrati da due sole camere, un numero di macchine superiore da calibrare non può essere allora studiato con tale sistema. Per rimediare a questa limitazione è stato ideato un metodo comprendente una serie di calibrazioni multiple i cui dati vengono poi ricondotti tutti al sistema di riferimento della *Camera 1*.

Preso un sistema come quello in *Figura 3.5* si eseguono le classiche calibrazioni 21, 32, 43, n4 registrando quindi *n-1* video diversi, in cui il pattern dovrà essere inquadrato ogni volta dalle due telecamere a cui si fa riferimento.

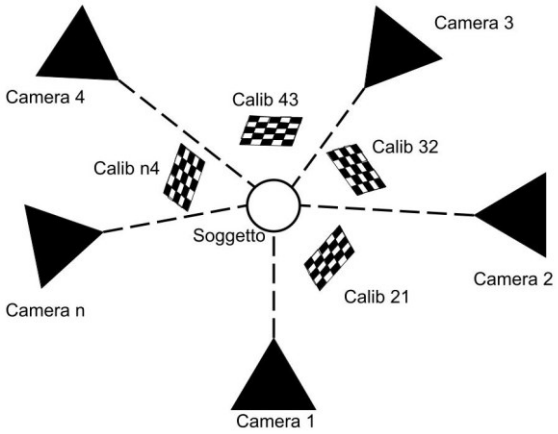


Figura 3.5: Sistema di posizione e calibrazione con n-camere

Successivamente per eseguire il processo di mutazione del sistema di riferimento si va ad ottenere la matrice di trasformazione T, che ha una struttura del tipo:

$$T = \begin{bmatrix} R & V \\ 0 & 1 \end{bmatrix}$$

Con R *matrice di rotazione* e V *vettore posizione relativi* (ciò che nelle calibrazioni viene chiamato *RotationOfCamera2* e *TraslationOfCamera2*) di una camera rispetto l'altra.

Prendendo come esempio T21, se questa viene moltiplicata scalarmente per la matrice di trasformazione della coppia di camere successive, in questo caso T32, si ottiene la matrice T31, i cui componenti saranno proprio l'orientazione e la posizione relativa tra la camera 3 e la 1.

Sullo script si estraggono innanzitutto gli *StereoParams* di tutte le *n-1* calibrazioni eseguite, dopodiché si compone la prima matrice T (T21) riempiendone una generica 4x4 di zeri con gli elementi sopracitati:

```
T.T_21=zeros(4,4);
T.T_21(1:3,1:3)=stereoParams.stereoParams_21.RotationOfCamera2';
```

```
T.T_21(1:3,4)=stereoParams.stereoParams_21.TranslationOfCamera2;
T.T_21(4,4)=1;
location(1,:)=zeros(1,3);
location(2,:)=stereoParams.stereoParams_21.TranslationOfCamera2;
orientation(:,:,1)=[1 0 0; 0 1 0; 0 0 1];
orientation(:,:,2)=stereoParams.stereoParams_21.RotationOfCamera2;

for i=3:n_cams
    Ta=T.(['T_' int2str(i-1) int2str(1)]);
    rb=stereoParams.(['stereoParams_' int2str(i) int2str(i-
1)]).RotationOfCamera2';
    tb=stereoParams.(['stereoParams_' int2str(i) int2str(i-
1)]).TranslationOfCamera2;
    Tb=zeros(4,4);
    Tb(1:3,1:3)=rb;
    Tb(1:3,4)=tb;
    Tb(4,4)=1;
    T.(['T_' int2str(i) int2str(1)])=Ta*Tb;
end
```

Costruita poi anche la seconda matrice di trasformazione necessaria, la si moltiplica per la precedente e si salva il risultato ottenuto. Questo processo è completamente automatizzato essendo inserito all'interno di un ciclo for, così che tutti i prodotti vengano eseguiti automaticamente, in modo da ottenere, in fase finale, una struttura contenente le varie matrici di trasformazione.

3.2 Sviluppo dei dati

Il passo successivo consiste nel registrare i video sincronizzati del movimento da studiare, ma affinché Matlab possa eseguire l'analisi dei dati si rende necessario un passaggio intermedio, che consiste nel caricare quest'ultimi e quelli generati dalle calibrazioni per inserirli in strutture e matrici più consone allo studio. Inizialmente i file *.txt* prodotti dalla rete neurale contengono un numero di righe pari al numero di frames per le 33 parti del corpi, alle quali sono associate quattro colonne pari alle coordinate in pixel *{x,y,z}* e la visibilità. Quest'ultimo parametro in particolare è utile per effettuare un'analisi valutativa del processo di acquisizione: più il numero è vicino ad 1 e più la visibilità è alta (banalmente la visibilità pari a 0 non dà alcuna coordinata perché il punto non è stato rilevato).

Da questo processo di ricollocamento si ottengono i file *.pos* , che contengono due sole colonne delle coordinate *{x,y}* dei punti, essendo i valori in {z} non accettabili per i motivi già illustrati.

Una volta caricati i dati di calibrazione (intrinseci, estrinseci ed estrinseci relativi) e di movimento si trasformano i valori nulli [0] in valori [*NaN*]. Questo si rende necessario a causa di un diverso approccio del significato di zero nei due programmi: per Mediapipe è un mancato rilevamento (ossia un "non valore"), per Matlab invece la coordinata nulla (l'origine). Non modificare questi valori genera quindi un errata acquisizione dei punti.

Un processo fondamentale da eseguire ogni qual volta si acquisiscono dei dati è il filtraggio, che permette di migliorare i valori ottenuti, eliminare quelli fuori scala e di "smussare" i movimenti così da generare un'animazione fluida e continua.
Il processo permette quindi di compensare i valori fuori norma dovuti ad un errata acquisizione da parte delle camere ed interpretazione della rete neurale.
Il parametro più importante che riguarda questa tipologia di processo è il passo, strettamente legato al tipo di movimento.
In genere si sceglie di adottare un sistema di filtraggio tramite medie di valori calcolate nell'intorno di ogni punto: se il passo è troppo elevato, lo sarà anche l'intervallo di valori presi per eseguire la media generando quindi dei numeri non più veritieri ma duramente compromessi; in caso contrario lo sviluppo dell'operazione porterebbe ad un'amplificazione degli errori, in quanto il range di valori sarebbe troppo piccolo per poter eseguire una media. In generale si impone come passo di filtraggio un tempo pari all'intervallo necessario ad eseguire il movimento di studio (considerando il movimento di una bracciata la scelta di *passo_t*=1 è ottima).

```
passo_t=table2array(parametri(4,2));
passo=passo_t/dt;
for j=1:n_cams
    for i=1:size(pos,1)
        a=squeeze(pos(i,1:2,:,j))';
        aa=smoothdata(a,'lowess',passo)';
        poses_multi(i,1:2,:,j)=aa;
    end
end
```

La funzione di Matlab che permette di attuare questo processo è *"smoothdata"*, in cui una volta inseriti i dati da filtrare *(pos)* e scelto il metodo di approssimazione migliore *(lowess)*, vengono forniti in uscita i valori finali filtrati.
Per un confronto dei diversi metodi di filtraggio e di studio dei dati in output riferirsi al *Capitolo 4*.

Il processo di triangolazione utilizza la funzione *"triangulateMultiview"* con la quale si ottiene un unico file di coordinate di punti partendo da numero di file pari al numero delle camere.

```
pairsIdx=[(1:33)' (1:33)'];
for jj=1:frames
    vSet = imageviewset;
    points=SURFPoints(squeeze(poses_multi(:,:,jj,1)));
    vSet =
addView(vSet,1,rigid3d(orientation(:,:,1),location(1,:)),'Points',points);

    for i = 2:n_cams
        points=SURFPoints(squeeze(poses_multi(:,:,jj,i)));
       vSet = addView(vSet,i,rigid3d(orientation(:,:,i),
location(i,:)),'Points',points);
        vSet = addConnection(vSet,i-1,i,'Matches',pairsIdx);
    end

    tracks = findTracks(vSet);
    cameraPoses = poses(vSet);
    xyzPoints(:,:,jj)=triangulateMultiview(tracks,cameraPoses,intrinsics);
    clear vSet
end
```

Come si nota dallo script precedente nei primi due blocchi sono presenti altrettanti cicli *for*: il primo ha il compito di caricare i dati della camera di riferimento e il secondo di abbinare a questi i parametri delle altre macchine. Il termine *"vset"* è necessario al fine di collegare ognuno dei 33 punti rilevati in ogni frame della prima camera i suoi corrispondenti nei frames delle altre macchine, passando per la scrittura *"SURF"*. Una volta creata tramite la funzione *"addview"* questa variabile comprende tutti i dati dei punti, orientazione, posizione della prima camera. Analogamente, con il secondo ciclo for verranno ottenuti anche quelli delle successive macchine e con il passaggio successivo di *"addConnection"* si uniscono i risultati.

A questo punto nel terzo blocco di codici si estraggono da *"vset"* i vari punti associati su più viste tramite *"findTrack"* creando una struttura con i valori *{x,y}* di tutte le inquadrature relative ad ogni punto e una tabella di pose assolute associate alle viste contenute nel set tramite la funzione *"poses"*. Questi due blocchi di dati, insieme ai parametri intrinseci, formano gli input necessari alla funzione di triangolazione *(pointTracks, cameraPoses, intrinsic)*. Da notare che i punti di track prima dell'elaborazione sono relativi alle dimensioni in x e y, mentre con la triangolazione si ricava anche la z, ottenendo un file con *nFrames* blocchi di 33 coordinate in *{x,y,z}*.

Lo script termina con il processo di visualizzazione del corpo 3D.

Nella fase finale si ricostruisce il corpo del soggetto in una versione semplificata, definita comunemente *"skeleton"*, per ovvi motivi di calcolo e di possesso dei dati. Vengono creati i link di collegamento tra i punti associati, seguendo meticolosamente l'ordine con la quale questi sono stati assegnati e poi inseriti nei file *.txt* dal programma Mediapipe. Ad ogni numero corrisponderà una diversa parte del corpo che sarà associata ad un'altra così da ricostruire il soggetto.

Di nuovo si utilizzano due cicli for per visualizzare i diversi punti e poi unirli tramite un link rigido.

```
for j=1:mm

scatter3(squeeze(points(:,1,j)),squeeze(points(:,2,j)),squeeze(points(:,3,j))),ho
ld on

    for kk=1:size(link,1)
        a=link(kk,1)+1;
       b=link(kk,2)+1;
      plot3([squeeze(points(a,1,j)),squeeze(points(b,1,j))]',[squeeze(points
      (a,2,j)),squeeze(points(b,2,j))]',[squeeze(points(a,3,j)),squeeze(poin
      ts(b,3,j))]','r-')
    end
    % axis equal
    xlabel('X')
    ylabel('Y')
    zlabel('Z')
    axis([-2000 2500 -25000 -10000 -2000 2500])
    pause(.01)
    hold off
end
```

Tramite la funzione *"scatter3"* si graficano i punti in funzione dei frame, mentre con *"plot3"* il collegamento tramite una retta di questi. Importante è l'utilizzo del termine *"hold on/off"* che consente di mantenere la visualizzazione degli elementi precedenti durante lo svolgimento del ciclo.

Un ultimo problema riguarda la scala di visualizzazione che, essendo dipendente dalla distanza tra soggetto e camera, può variare da acquisizione ad acquisizione: sono allora presenti due soluzioni riguardanti lo spazio di visualizzazione, una solidale al soggetto tramite la funzione *"axis equal"*, che cambia quindi il range di valori in *{x,y,z}* con il proseguo del movimento, e un'altra fissa in cui l'intervallo di spazio negli assi è impostato dall'operatore (in genere la prima tipologia è utilizzata per determinare i valori estremi da utilizzare poi nel secondo caso).

L'inserimento delle camere avviene automaticamente adottando il secondo script per la visualizzazione, che sfrutta la funzione *"plot camera"*. Quest'ultima utilizza i dati estrinseci relativi delle camere per visualizzarle nella loro posizione ed orientazione.

La visualizzazione dei dati ha fatto sorgere problemi riguardanti il sistema di riferimento, in quanto la calibrazione e la rete neurale non adottano gli stessi criteri per la loro imposizione. Come già illustrato nel Paragrafo 1.1 il tool di calibrazione imposta l'origine degli assi in alto a sinistra, cosa non eseguita anche da Mediapipe. Durante la visualizzazione questo genera per cui una figura "capovolta", come è possibile notare in *Figura 3.6*, ma questo in realtà non influisce sui risultati finali essendo questi indipendenti dal plottaggio. Per ottenere un risultato più facilmente interpretabile (quindi solo dal punto di vista qualitativo) è possibile variare la posizione degli assi x, y e z. Da notare che questa soluzione non è però applicabile alla visualizzazione con camera, poiché altrimenti queste risulterebbero invertite, dato che la loro posizione è ottenuta direttamente dai parametri di calibrazione, e quindi non modificabile.

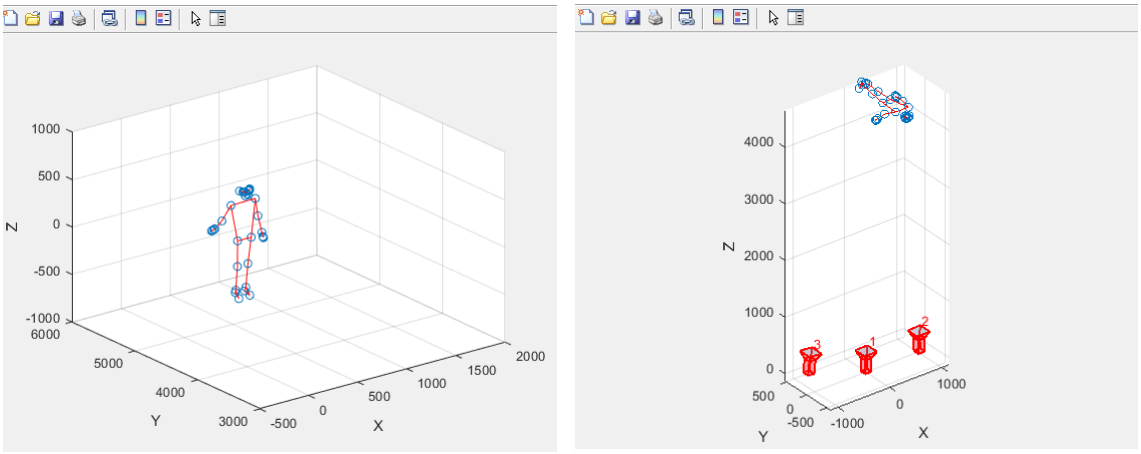


Figura 3.6: visualizzazione del movimento (a sinistra) e con camere (a destra)

3.3 Studio cinematico

Lo studio cinematico condotto nel caso bicamera riguarda principalmente l'andamento della posizione e velocità di un qualsiasi punto di tracking in funzione dei frame e la focalizzazione della traiettoria compiuta da una generica parte del corpo con relativa evoluzione grafica.

Per il primo studio, una volta eseguiti i passaggi preliminari riguardanti la calibrazione e l'ottenimento dei file di posizione, si derivano quest'ultime così da generare un file *"v"* contenente le velocità in x e y di ognuno dei 33 punti in funzione dei frames.

```
parte=16;

figure
subplot(2,1,1)
plot(squeeze(poses(parte,1,:)),'b-*')
xlabel('Frames')
ylabel('X')

title(cell2mat(target(parte,2)))
subplot(2,1,2)
plot(squeeze(v(parte,1,:)),'r-*')
xlabel('Frames')
ylabel('V_x')
```

Inizialmente si impone il numero della parte di cui si vuole studiare la cinematica, successivamente, nel primo e secondo blocco, si esegue la visualizzazione della compente della velocità scelta. Per farlo si utilizza la classica funzione *"plot"* e *"subplot"* che genera contemporaneamente finestre con grafici che descrivono l'andamento della posizione x in funzione dei frame, e della velocità *Vx* sempre in funzione delle immagini.

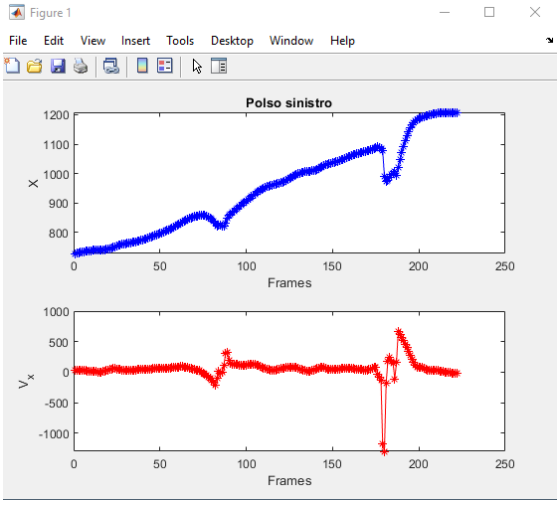


Figura 3.7: Grafici di posizione e velocità del polso in un'acquisizione

Come si nota dalla *Figura 3.7* l'utilizzo di questa tipologia di visualizzazione dei dati è molto utile per studiare l'andamento e la fedeltà dei dati raccolti rispetto i movimenti reali e di valutare anche la bontà di quest'ultimi da un punto di vista sportivo.

Capitolo 4

Risultati e analisi critica

In questa sezione verranno presentati i risultati più rilevanti riguardanti varie prove effettuate, focalizzando l'attenzione sulle acquisizioni con dati più attendibili e precisi, sia nel caso bicamera che nel caso con un numero superiore. Saranno poi affrontati in modo critico i risultati così da valutare il sistema creato e la capacità dello stesso di acquisire i dati e riprodurli virtualmente.

4.1 Condizione stereo camere

Durante l'iniziale fase di studio è stato conveniente posizionare la camera di riferimento in modo che percepisca il soggetto esattamente perpendicolare, senza angoli di inclinazione né con il pavimento né con l'orizzonte, al fine di facilitare l'analisi e l'interpretazione dei dati di calibrazione e della ricostruzione virtuale della scena con camere plottate. Essendo lo scopo della trattazione quello di costruire un sistema di rilevamento del movimento con posizione delle camere quanto più arbitrarie possibili, sottolineiamo che il setup precedente è da preferire nelle fasi iniziali di studio ma successivamente non obbligatorio, non essendo ovviamente indispensabile all'applicazione stereoscopica.

Detto questo, e aggiungendo che per la seconda camera non sono mai presenti restrizioni riguardanti le inclinazioni, sono state effettuate varie prove, illustrate in *Figura 3.1*, tra cui le più rilevanti sono il caso di parallelismo dell'asse ottico con la *Camera 1* (c) e quello di convergenza di entrambi gli obiettivi verso il soggetto (b).

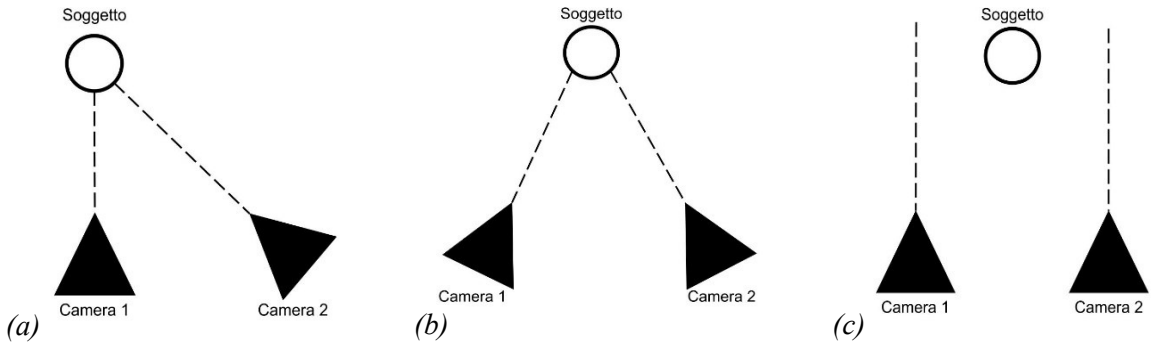


Figura 3.1: Casi particolari di posizione relativa tra camere e soggetto: una perpendicolare e una inclinata (a), entrambe inclinate (b), entrambi perpendicolari (c).

Infine, al solo scopo di studio, è stata effettuata una prova con camere avente assi ottici inclinati di 90° l'una rispetto l'altra.

4.1.1 Acquisizione 1

La prima configurazione proposta comprende due camere posizionate come in *Figura 4.1* con le relative distanze, e il soggetto che si sposta verso destra. Il movimento scelto è pensato per simulare un tipico movimento acquatico, con un avanzamento verso destra e due bracciate.

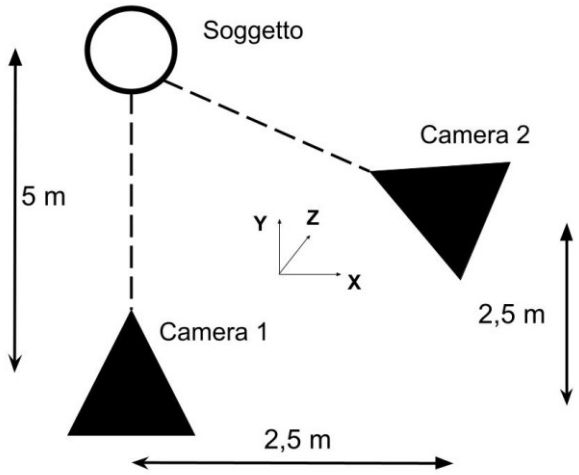


Figura 4.1: configurazione hardware acquisizione A

Tramite lo script si è ottenuta la visualizzazione tridimensionale di *Figura 4.2*, e con l'elaborazione dei dati la descrizione della traiettoria, visibile in *Figura 4.3*, della parte del corpo scelta, in questo caso il polso destro. Il grafico ha in ascisse la posizione X e nelle ordinate la Z.

Da una prima analisi critica si può notare che lo spazio di acquisizione è stato rilevato in modo accurato dal sistema, sia da un punto di vista di orientazione delle camere sia di posizione relativa tra esse e il soggetto. Analogamente, confrontando il video reale con la traiettoria di *Figura 4.3* è possibile affermare che il movimento riprodotto è molto affidabile, tale da poter facilmente identificare le due bracciate eseguite dal soggetto. Dalla densità di punti è possibile ipotizzare un primo andamento delle velocità del polso, che sarà minima in corrispondenza di valori (in metri) di ascissa 750, 900 e 1100, dovuti allo stato pressoché di quiete del polso, e massimi nel resto del grafico.

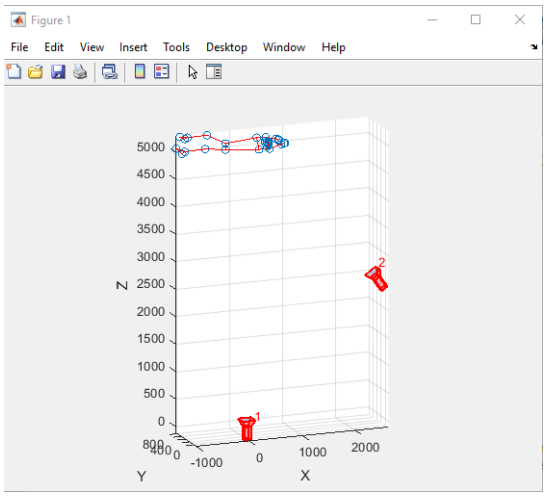


Figura 4.2: visualizzazione tridimensionale del movimento

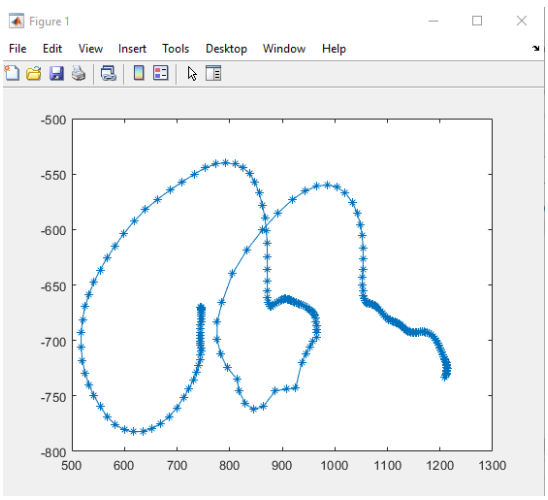
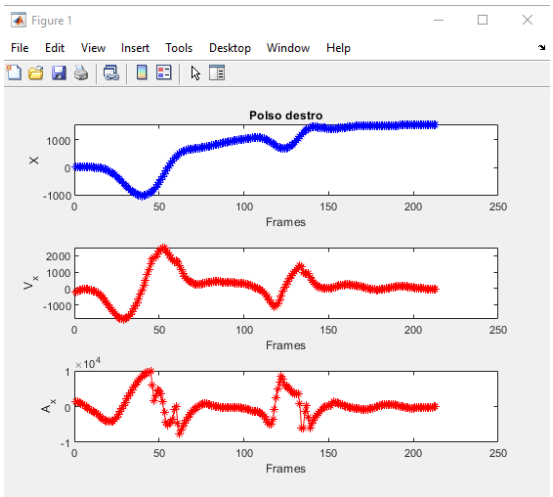


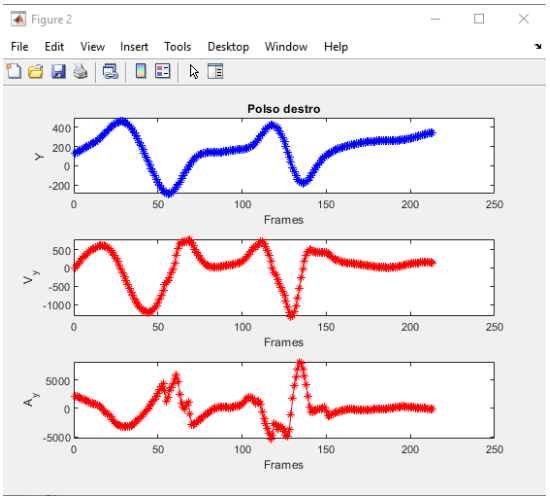
Figura 4.3: visualizzazione traiettoria del polso

In *Figura 4.4* sono graficati gli andamenti della posizione, velocità e accelerazione del punto scelto, relativo ad ogni asse, in funzione dei frames (per la definizione degli assi si faccia riferimento alla *Figura 4.1*).

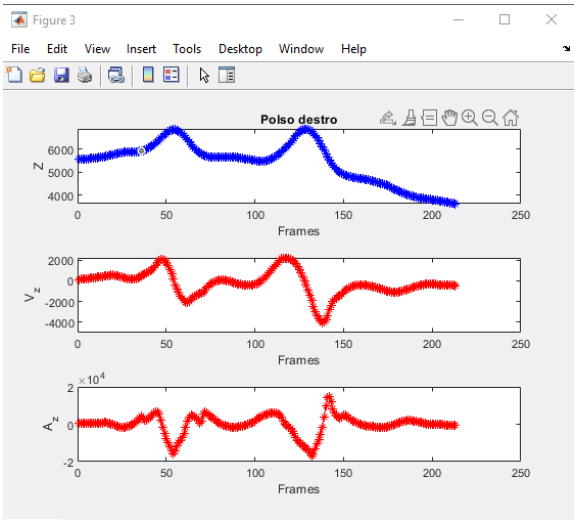
Tramite questi grafici si può concludere che lo studio dell'andamento della X permette di determinare in quali frame è avvenuta la bracciata e il momento in cui è stata raggiunta anche la velocità massima. Eventuali valori fuori scala permettono di identificare in quali punti il sistema non è riuscito, seppure adottando la soluzione stereoscopica e la triangolazione dei dati, ad acquisire la posizione degli elementi. Questo è più evidente se si considera una parte del corpo "nascosta" dal corpo stesso: data la configurazione iniziale è ovvio pensare che parti come polso sinistro, ginocchio sinistro o spalla sinistra, siano stati acquisiti in modo più difficoltoso.



(a)



(b)



(c)

Figura 4.4: visualizzazione in X (a), Y (b) e Z (c) della posizione, velocità e accelerazione del polso destro nella configurazione A

Nel caso in esame si può affermare che il sistema è comunque riuscito a soddisfare abbondantemente i requisiti di acquisizione, identificando ed elaborando anche parti eventualmente nascoste. Ne è la prova la *Figura 4.5* che identifica i dati cinematici della spalla sinistra.

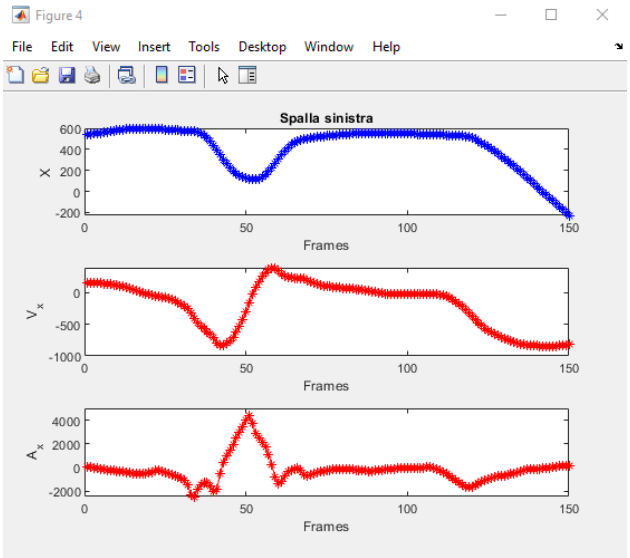


Figura 4.5: visualizzazione in X della spalla sinistra nella configurazione A

4.1.2 Acquisizione 2

In questa acquisizione la posizione hardware è descritta in *Figura 4.6*, in cui le camere sono disposte in modo perpendicolare tra loro. In questo caso il movimento del soggetto non comprende una traslazione come nella prova precedente ma è puramente limitato a braccia e gambe.

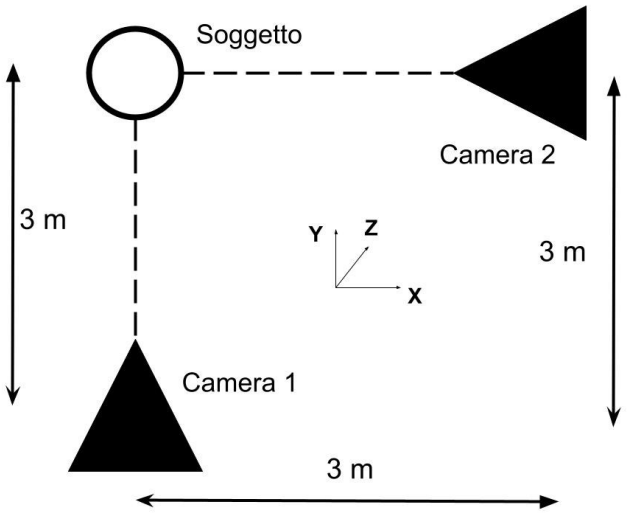


Figura 4.6: Configurazione hardware acquisizione B

Sottolineiamo che la prova seguente ha principalmente scopo di studio e di verifica delle potenzialità del sistema. La camera due, inquadrando frontalmente il soggetto, non è posizionata in modo ottimale considerando il movimento in esame, poiché in questa vista esso avviene principalmente lungo la direzione di profondità, che sappiamo essere quella più critica dal punto di vista dell'acquisizione. In *Figura 4.7* è mostrata la rappresentazione 3D del movimento, e in *Figura 4.8* la traiettoria globale percorsa dal polso destro.

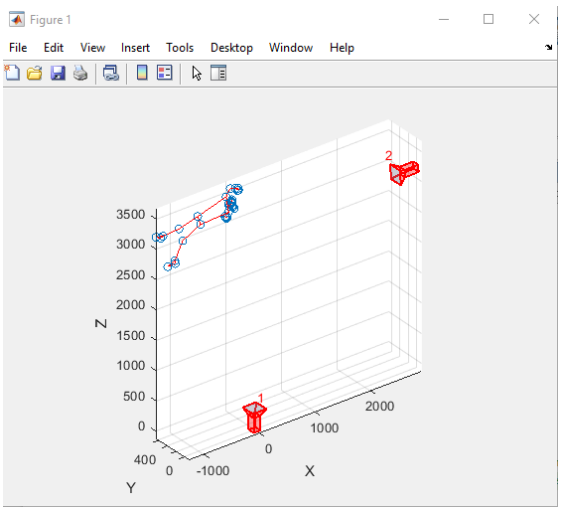


Figura 4.7: visualizzazione tridimensionale del movimento

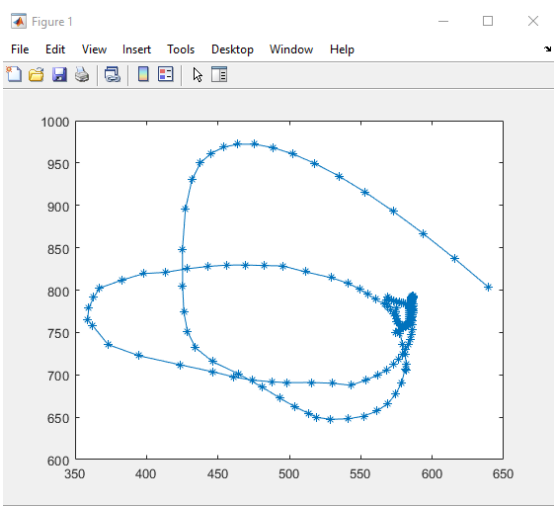
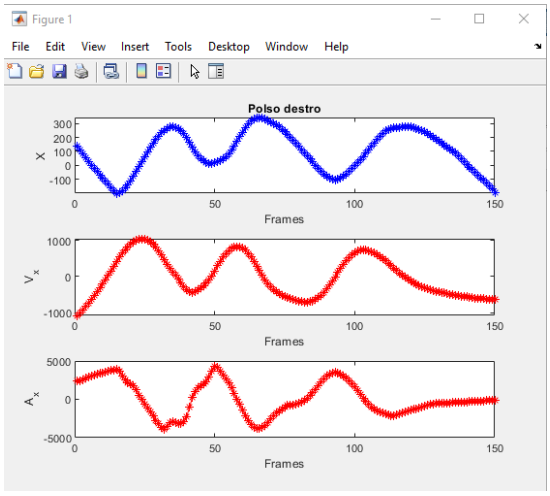
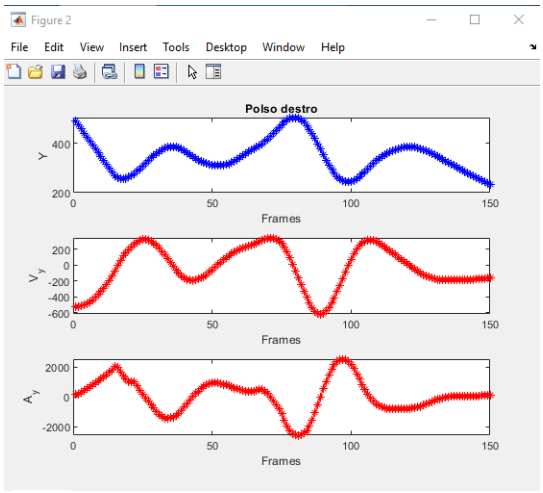


Figura 4.8: visualizzazione traiettoria del polso

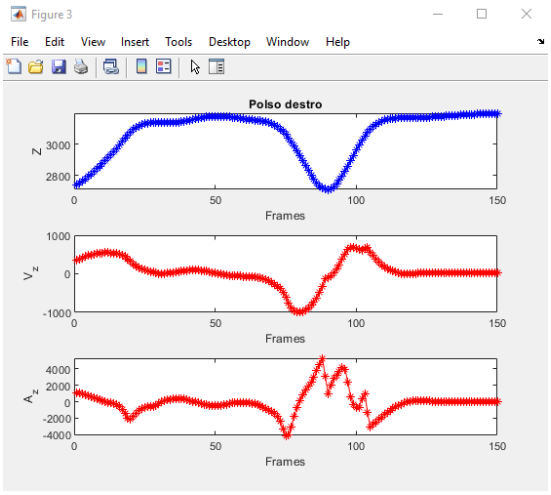
Per quanto riguarda i grafici delle storie temporali del polso, queste sono rappresentate in *Figura 4.9*, ma l'aspetto più importante si può notare in *Figura 4.10*, in cui è mostrato l'andamento lungo X del punto identificato come *"naso"*. Considerando il movimento in esame, la sua posizione dovrebbe essere pressoché costante: infatti come è possibile notare si hanno solamente delle piccole variazioni in corrispondenza delle due bracciate, dovute probabilmente ad un reale movimento del soggetto o da un errore di acquisizione. Questo fortifica l'ipotesi di un buon rilevamento da parte del sistema che quindi, anche in condizioni più sfavorevoli come queste, permette comunque di ottenere dati cinematici.



(a)



(b)



(c)

Figura 4.9: visualizzazione in X (a), Y (b) e Z (c) della posizione, velocità e accelerazione del polso destro nella configurazione B

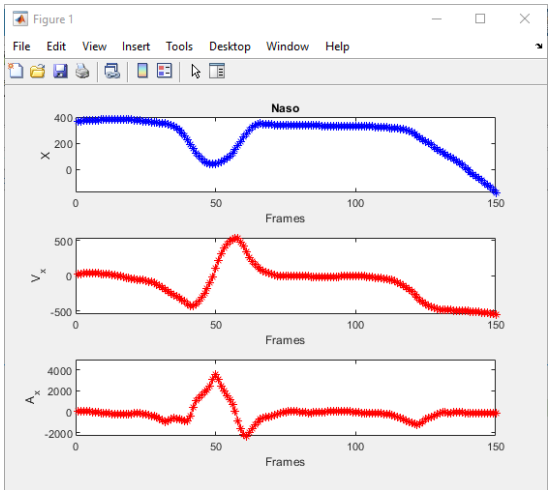


Figura 4.10: visualizzazione in X della posizione, velocità e accelerazione del naso nella configurazione B

4.1.3 Analisi del filtraggio

In questa parte sono presentati i principali risultati del confronto tra vari metodi di filtraggio forniti dalla funzione di Matlab *"smoothdata"*. Come già introdotto, questa funzionalità del programma permette, a seconda del tipo di dato in input (tabelle, array...), di eseguire un filtraggio dei valori fornendo una media degli elementi nell'intorno della finestra di dati considerati.

Il codice utilizzato è:

```
---------------------------------------------------------------------------------------------------
B = smoothdata(__,method,window)
---------------------------------------------------------------------------------------------------
```

"Window" rappresenta appunto la finestra di campionamento, cioè l'intervallo a ridosso del punto da filtrare nella quale saranno presi i dati di input necessari per il processo: è stato scelto un valore pari a *passo_t* uguale a 1 (ossia al tempo in secondi) diviso *dt*, dove questo rappresenta il tempo di acquisizione di un frame (dt=1/Framerate). I motivi della scelta dell'ampiezza di questo parametro sono stati già affrontati, per cui si consulti il Paragrafo 3.2 *"Processo di triangolazione e visualizzazione"*

I vari metodi di filtraggio gestiscono i dati in maniera differente, fornendo in output risultati diversi e rendendo quindi necessario scegliere un sistema rispetto un altro.

Per poter definire il migliore da adottare sono stati eseguiti dei test elaborando i dati tramite i vari metodi e visualizzando l'andamento grafico di un generico punto (polso destro) prima e dopo il filtraggio. I grafici sono riferiti alle posizioni dei punti prima della triangolazione, per cui sono presenti gli spostamenti rilevati da entrambe le

camere. Per ultimo è presente il plottaggio dello spostamento triangolato. I grafici di *Figura 4.11* sono descritti in frame (asse x) in funzione dello spostamento (asse y), e sono ora illustrati:

a. *"Gaussian"*: smussa i dati campionati tramite un filtro gaussiano, ponendo un peso maggiore al centro degli stessi.

b. *"Movmean"*: è un metodo abbastanza standard che sfrutta la semplice media matematica, utile per ridurre le tendenze periodiche dei dati.

c. *"Lowess"*: viene eseguita una regressione lineare sulla finestra impostata, producendo meno discontinuità. Questo è il metodo utilizzato nel nostro caso, in quanto tramite un passaggio visivo si può notare come la curva di spostamento risulti più fluida e non alterata.

d. *"Rlowess"*: è analogo al caso precedente, ma in questa tipologia la regressione lineare si fa più robusta. Questo permette di gestire meglio i dati anomali ma va a modificare eccessivamente la curva, e quindi lo spostamento.

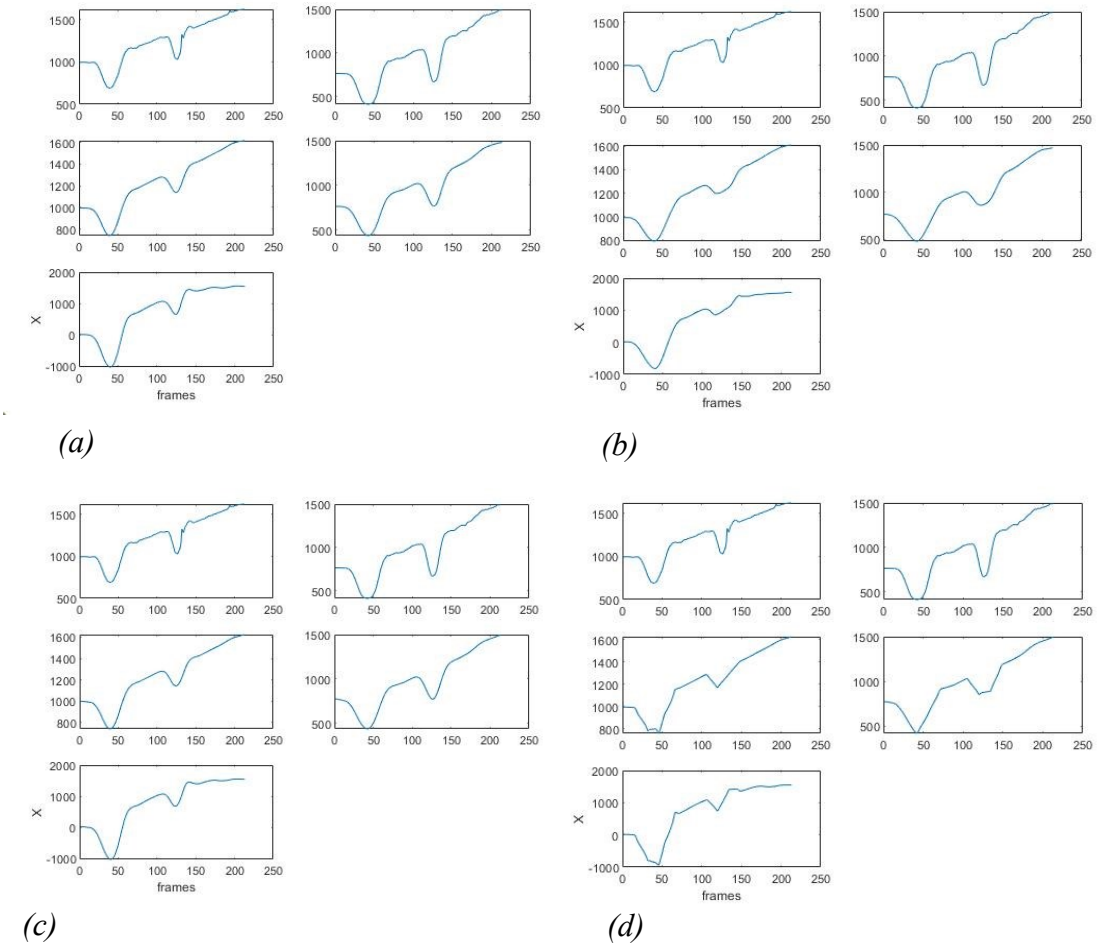


Figura 4.11: Grafici di filtraggio

Come si può notare, le differenze tra il metodo *a.* e *c.* sono minimi considerando poi il processo di triangolazione e visualizzazione, ma questo non è altrettanto vero considerando filtraggi più pesanti come il caso *d.*

Per concludere viene proposto un confronto, sempre adottando lo schema grafico visto finora (*Figura 4.12*), riguardante l'importanza della scelta di un buon intervallo di campionamento (finestra) che, a prescindere del metodo di filtraggio utilizzato, ha un ruolo fondamentale ed incisivo sui risultati.

Aumentando troppo questo dato, la media dei punti campionati per il processo di *smooth* è tale da inficiare i risultati finali, e quindi i movimenti caratteristici.

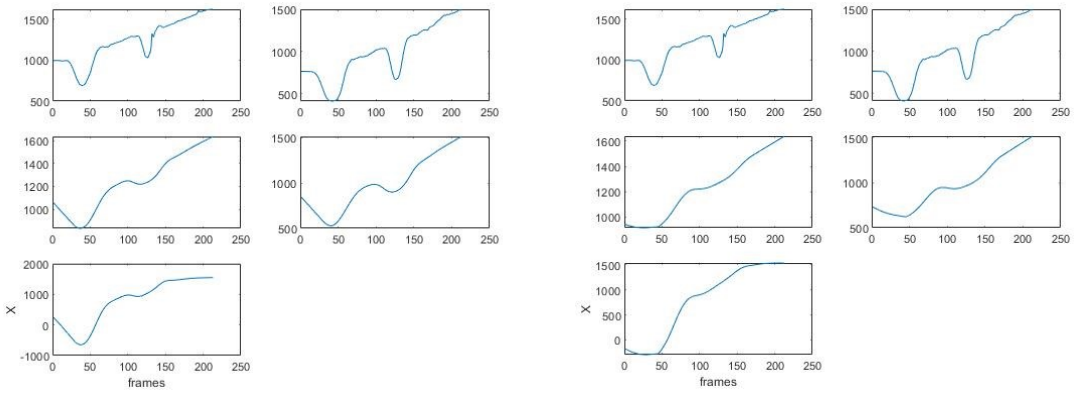


Figura 4.12: Confronto tra diversi passi di filtraggio.
Passo_t= 2 (sinistra) e Passo_t= 3 (a destra)

4.2 Condizione n-camere

4.2.1 Acquisizione 3

In questo test di acquisizione n-camera sono state adottate tre macchine, disposte come in *Figura 4.13*.

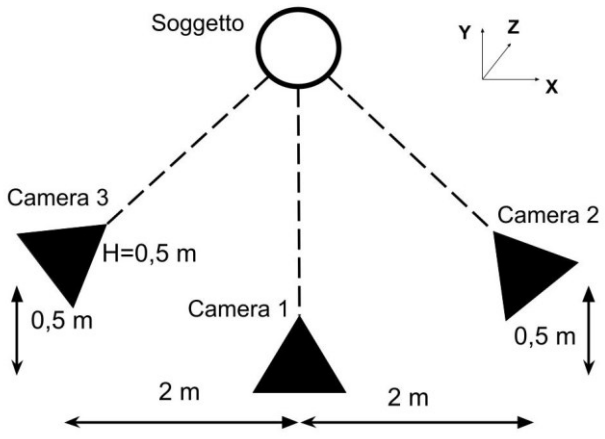


Figura 4.13: Configurazione hardware acquisizione C

Due camere sono state collocate sulla struttura illustrata nel Capitolo 2, di cui la centrale (1) perpendicolare al soggetto e l'altra (2) inclinata verso di esso; la terza camera (3) è stata inserita su un cavalletto in modo simmetrico alla (2) rispetto la centrale ma ad una altezza diversa, di circa mezzo metro. Questo significa che la camera (3) avrà doppia inclinazione, rispetto gli assi Z e X (in *Figura 4.14* il sistema hardware).

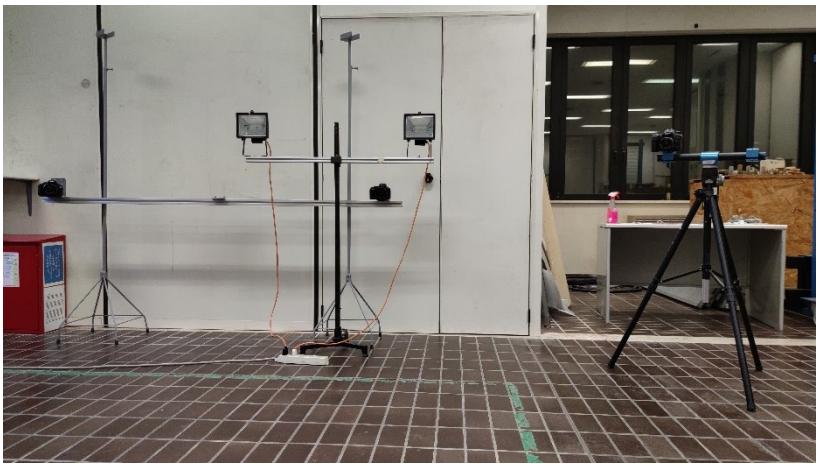


Figura 4.14: Configurazione hardware fisica acquisizione C

L'utilizzo dello script automatico ha permesso di ottenere l'andamento tridimensionale del movimento illustrato in *Figura 4.15*.

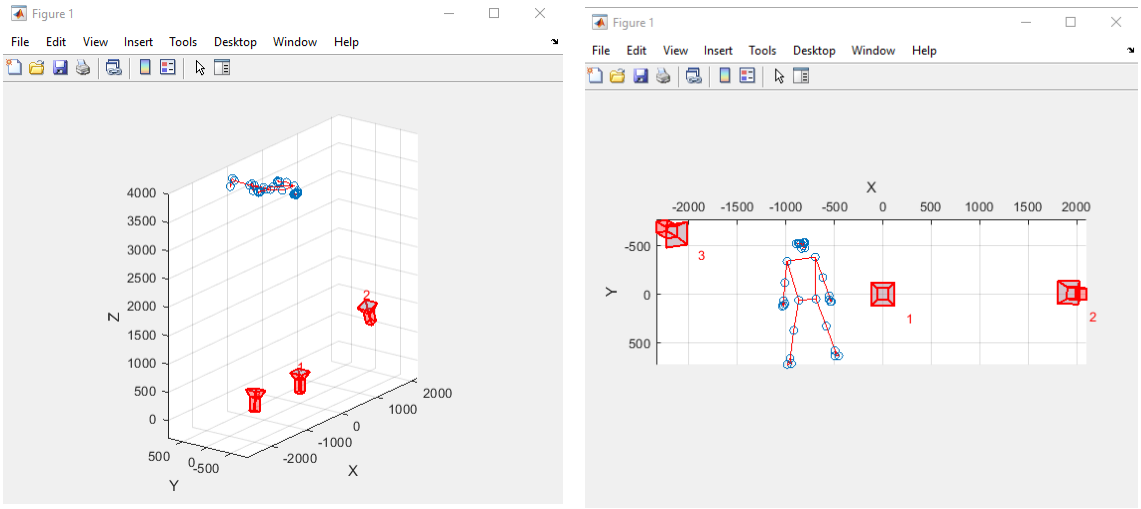


Figura 4.15: Visualizzazione tridimensionale del movimento

Una semplice analisi dei dati ottenuti permette di affermare che le camere sono state rilevate in modo accurato dal sistema; per quanto riguarda il movimento questo risulta fluido e preciso.

Un problema riscontrabile riguarda invece l'altezza dell'individuo: esso risulta infatti almeno 20 cm più basso rispetto alla realtà. Con gli attuali dati in possesso è possibile ipotizzare che questo problema di scala sia da attribuire alla rete neurale o a giochi di prospettiva delle varie macchine dovuta dalle diverse angolazioni.

A scopo di verifica è stato eseguito uno spostamento "quadrato" dal soggetto, con passi di circa un metro in tutte le direzioni (avanti, indietro, destra, sinistra). Come è possibile notare dalla sequenza delle immagini in *Figura 4.16*, considerando la scala degli assi Z e X i passi risultato infatti distanziati della stessa misura, inoltre anche l'ordine di evoluzione temporale dei movimenti è corretto.

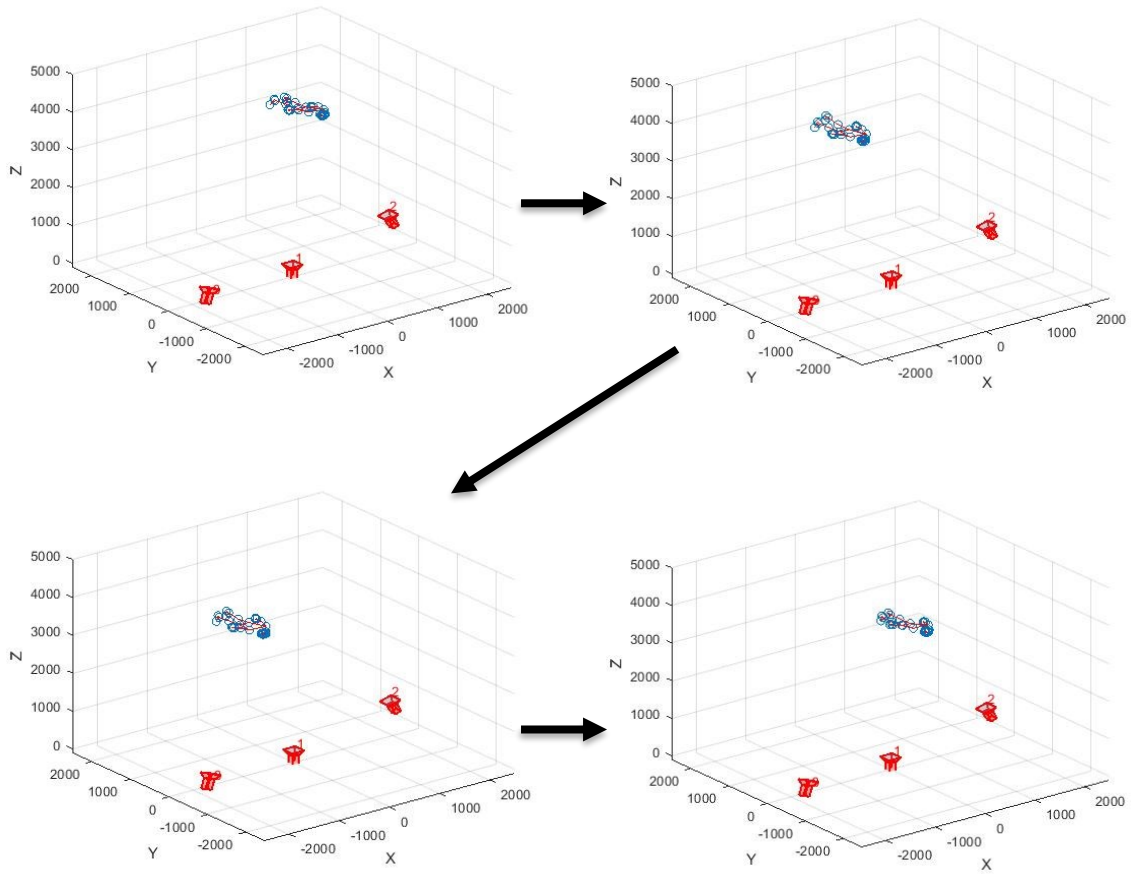


Figura 4.16: Visualizzazione di un movimento su un quadrato

Capitolo 5

Conclusioni

Lo studio di un nuovo metodo di acquisizione del movimento in situazioni di difficoltà di rilevamento (come quello in acqua) ha permesso di stabilire i fattori caratterizzanti questo campo. Dal lato software abbiamo i sistemi di reti neurali, il cui studio è attualmente sempre più in crescente sviluppo, che costituiscono una solida ed essenziale base per l'analisi, che sarà tanto migliore quanto più robusto è il programma; dal lato hardware, invece, non essendo necessarie apparecchiature specifiche, si possono adottare comuni reflex o macchine compatte (quali ad esempio le Raspberry Pi) per condurre valutazioni accurate tramite programmi di calcolo matematico.

L'utilizzo di un sistema di stereo camere, che elimina la necessità di adottare tecnologie specifiche per il rilevamento della profondità, e le numerose prove eseguite, che hanno provato l'affidabilità e solidità del sistema, unite all'assenza di vincoli riguardanti il numero di camere, il tipo di movimento e l'ambiente di acquisizione, lo rende un sistema economicamente e tecnologicamente vantaggioso, con possibilità di utilizzo in vari campi oltre a quello della ricerca, e una valida alternativa a sistemi di tracking più complessi che necessitano invece di particolari competenze.

L'utilizzo del seguente approccio di rilevamento si può allora considerare una delle migliori scelte nel campo dell'acquisizione qualitativa del movimento, dove l'analisi interessa principalmente la posizione e i parametri cinematici degli arti.

Bibliografia

[1] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank, “Camera Self-Calibration: Theory and E x p e r i m e n t s.”

[2] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000, doi: 10.1109/34.888718.

[3] Q. Wang, L. Fu, and Z. Liu, “Review on camera calibration,” in *2010 Chinese Control and Decision Conference, CCDC 2010*, 2010, pp. 3354–3358. doi: 10.1109/CCDC.2010.5498574.

[4] J. I. Woodfill, G. Gordon, and R. Buck, “Tyzx deepsea high speed stereo vision system,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2004, vol. 2004-January, no. January. doi: 10.1109/CVPR.2004.469.

[5] D. Wan and J. Zhou, “Stereo vision using two PTZ cameras,” *Computer Vision and Image Understanding*, vol. 112, no. 2, pp. 184–194, Nov. 2008, doi: 10.1016/j.cviu.2008.02.005.

[6] J. Han, L. Shao, D. Xu, and J. Shotton, “Enhanced computer vision with Microsoft Kinect sensor: A review,” *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013, doi: 10.1109/TCYB.2013.2265378.

[7] K. Kumatani *et al.*, “Microphone Array Processing for Distant Speech Recognition: Towards Real-World Deployment.”

[8] J. Smisek, M. Jancosek, and T. Pajdla, “3D with Kinect,” in *Consumer Depth Cameras for Computer Vision*, Springer London, 2013, pp. 3–25. doi: 10.1007/978-1-4471-4640-7_1.

[9] D. M. Swoboda, “A Comprehensive Characterization of the Asus Xtion Pro Depth Sensor! !,” 2014.

[10] A. F. Bobick and J. W. Davis, “The recognition of human movement using temporal templates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, Mar. 2001, doi: 10.1109/34.910878.

[11] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553. Nature Publishing Group, pp. 436–444, May 27, 2015. doi: 10.1038/nature14539.

[12] A. Alvin, N. Husna Shabrina, A. Ryo, and E. Christian, “Hand Gesture Detection for American Sign Language using K-Nearest Neighbor with Mediapipe,” *Ultima Computing : Jurnal Sistem Komputer*, vol. 13, no. 2, 2021.

[13] Inc. © 1994-2022 The MathWorks, “https://it.mathworks.com/help/matlab/.”

Ringraziamenti

Vorrei innanzitutto ringraziare la mia famiglia che mi ha permesso di poter arrivare fino a questo punto, sostenendo sempre psicologicamente ma anche fornendomi ogni mezzo necessario per poter seguire al meglio questo percorso di studi.

Ringrazio i miei amici, vecchi e nuovi, che mi hanno permesso di risollevarmi anche nei momenti più difficili, restandomi sempre vicino e aiutandomi nel momento del bisogno.

Una menzione va all'università, che mi ha permesso in questi anni di conoscere moltissime persone interessanti, e con alcune di instaurare un forte legame di amicizia, tale da spingermi sempre ad andare avanti, a fare di meglio e ad impegnarmi, anche quando ero io a non crederci.

In particolare, vorrei ringraziale il Professor Castellini, che mi ha permesso di approcciarmi a questo mondo che si è rivelato per me molto stimolante ed interessante, fornendo sempre con simpatia e curiosità nuovi approcci e metodi di analisi agli argomenti del tirocinio e di unire una mia passione, che è la fotografia, al lavoro della tesi.