UNIVERSITÀ DEGLI STUDI DI PADOVA

UNIVERSITÁ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria Industriale DII

Corso di Laurea Magistrale in Ingegneria Meccanica

Misura della posizione del pilota tramite una action camera

Relatore: **Prof. Roberto LOT**
Co-relatore: **Ing. Enrico GIOLO**

Laureando: **Alberto BRAI**
Matricola: **1058226**

Anno Accademico 2014-2015

Questa tesi è stata scritta con LaTeXsu Mac utilizzando il modello *Tesi Classica* di *Lorenzo Pantieri* disponibile nel sito http://www.lorenzopantieri.net. Le immagini utilizzate sono in parte realizzate dall'autore e in parte tratte dalle opere citate in bibliografia. L'imagine *"Mano con sfera riflettente"* è tratta dalle opere di Escher.

Il vero mistero del mondo è il visibile, non l'invisibile.

— Oscar Wilde

Alla mia famiglia.

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

In questa tesi sarà messo a punto un sistema in grado di determinare la posizione e l'orientamento del pilota durante la guida di una motocicletta. Questo metodo utilizza una telecamera montata sul codone della moto che riprende i movimenti del pilota. Tramite le immagini aquisite dalla telecamera si andrà a stimare la posizione del pilota rispetto alla moto. Dopo aver messo a punto il metodo tramite una procedura di calibrazione, verranno effettuate delle prove sperimentali con una motocicletta strumentata con dei sensori di coppia che misurano le forze scambiate dal pilota e si confronteranno i riultati ottenuti dai sensori, con un modello che utilizza la posizione del pilota ottenuta dalla telecamera per calcolare la coppia di rollio esercitata dal pilota.

Abstract

In this thesis will be developed a system capable of determining the position and orientation of a rider during driving a motorcycle. This method uses a camera mounted on the tail of the motorcycle that recording the movements of the pilot. Through the images from the camera we estimate the position of the rider than the motorcycle. After having correct the method through a calibration procedure, will be carried out experimental tests with a motorcycle instrumented with sensors that measure the torque forces exchanged by the pilot and will compared the results obtained by the sensors, with a model that uses the position of rider obtained from the camera to calculate the roll torque exerted by the rider.

Introduzione

Questa tesi si pone l'obiettivo di mettere a punto un sistema di misura della posizione e dell'orientamento del pilota durante la guida di una motocicletta. Questo metodo non dovrà utilizzare sensori a contatto in modo da non interferire con le azioni di guida del pilota. L'idea che verrà sviluppata è quella di utilizzare una telecamera montata sul codone della moto che riprende i movimenti del pilota, il quale avrà posizionato sulla schiena un particolare bersaglio. Tramite le immagini aquisite dalla telecamera si andrà a stimare la posizione del pilota rispetto alla moto.

Una volta messo a punto il metodo tramite una procedura di calibrazione verranno effettuate delle prove sperimentali con una motocicletta strumentata con dei sensori di coppia che misurano le forze scambiate dal pilota con la motocicletta e si confronteranno i riultati ottenuti dai sensori con un modello che utilizza la posizione del pilota ottenuta dalla telecamera per calcolare la coppia di rollio esercitata dal pilota.

La tesi è suddivisa in due parti:

Parte Prima che tratta lo sviluppo e la calibrazione del metodo per la stima della posizione del pilota, composta dai seguenti capitoli:

- **Capitolo uno** descrive le teorie e le tecniche di base utilizzate per realizzare il sistema di misura;
- **Capitolo due** descrive in che modo è stato possibile determinare la posizione del pilota tramite l'analisi del filmato ottenuto con la telecamera;
- **Capitolo tre** contiene la procedura di calibrazione nella quale si confronterà la misura ottenuta con il metodo ideato con la misura ottenuta da un sistema di *motion capture*;

Parte Seconda che riguarda i test sperimentali effettuati con la moto strumentata e comprende:

- **Capitolo quattro** in cui è descritta la strumentazione installata sulla motocicletta utilizzata per i test su strada;
- **Capitolo cinque** che riguarda l'analisi dei test effettuati con la motocicletta.

Infine l'appendice A contiene una parte del codice in linguaggio Matlab per l'elaborazione del filmato e l'ottenimento della misura di posizione del pilota.

Parte I

Il metodo per la stima della posizione del pilota

Capitolo 1

Teoria

In questo primo capitolo verranno descritte le tecniche impiegate all'interno di questa tesi, necessarie a mettere a punto il sistema di misura della posizione del pilota tramite una action camera.

Il primo paragrafo affronta l'argomento della rappresentazione della posizione e dell'orientamento di un corpo nello spazio, strumento indispensabile per poter esprimere la posa del pilota rispetto alla motocicletta e poter effettuare dei cambi di coordinate fra un sistema di riferimento ed un altro. Si vedrà quindi il metodo che sarà impiegato nel corso di tutto il lavoro.

Il secondo paragrafo descrive il processo di formazione dell'immagine. Si mostrerà come i punti di un oggetto reale vengono proiettati sul piano dell'immagine della *camera*, si ricaverà quindi l'espressione che rappresenta la trasformazione prospettica. Si vedranno i parametri caratteristici di una fotocamera e in che modo influenzano la proiezione prospettica. Verrà inoltre descritto il problema della distorsione lenticolare, aspetto molto importante che in questo lavoro sarà determinante nel ottenimento di misure accurate proprio perchè le *action cameras* che saranno impiegate sono caratterizzate da una distorsione lenticolare molto elevata, causata dal loro elevato campo visivo.

Il terzo paragrafo descrive il procedimento di *calibrazione*, ovvero il processo attraverso il quale vengono determinati i parametri *intrinseci* ed *estrinseci* di una camera. Verrà inoltre descritto l'algoritmo che stima la posizione di un oggetto rispetto a una camera calibrata tramite l'analisi della proiezione di un insieme di punti dell'oggetto.

L'ultimo paragrafo infine illustra le tecniche di *segmentazione dell'immagine* tramite le quali è possibile individuare ed analizzare particolari regioni all'interno dell'immagine.

1.1 Rappresentazione della posizione e dell'orientamento

La posizione di punto p nello spazio rispetto ad un sistema di riferimento globale $\{O\}$ può essere descritta da un vettore di coordinate (x_p, y_p, z_p). Per descrivere la posizione e orientamento nello spazio di un corpo rigido, anzichè considerare le coordinate degli n punti del corpo rispetto al sistema di riferimento globale, è conveniente associare all'oggetto in questione un sistema di riferimento solidale ad esso e descriverne la posizione e l'orientamento rispetto al sistema di riferimento globale. La posizione relativa o *posa* di un sistema di riferimento rispetto ad un altro sistema di riferimento viene indicata con il simbolo ξ. Come rappresentato in figura 1.2, la posizione relativa di $\{B\}$ rispetto ad $\{A\}$ è descritta dalla trasformazione ${}^{A}\xi_{B}$. Questa trasformazione può anche rappresentare un movimento: si immagini ad esempio di applicare una traslazione ed una rotazione ad $\{A\}$ che lo trasforma in $\{B\}$. Le coordinate del punto p rispetto ad $\{B\}$ possono essere espresse rispetto ad $\{A\}$ dalla seguente:

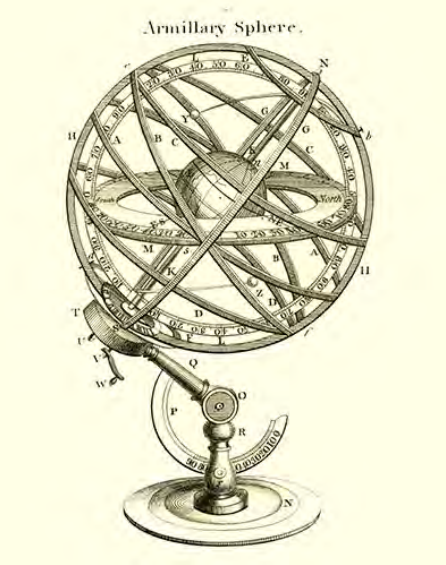


Figura 1.1: *Sfera armillare*

$$ {}^{A}\mathbf{p} = {}^{A}\xi_{B} \cdot {}^{B}\mathbf{p} \tag{1.1} $$

Le trasformazione che definiscono la posizione relativa tra due sistemi di riferimento possono essere combinate fra loro. Nel caso di figura 1.2 si può scrivere:

$$ {}^{A}\xi_{C} = {}^{A}\xi_{B} \oplus {}^{B}\xi_{B} \tag{1.2} $$

Si consideri ora la figura 1.3 e si immagini di voler determinare la posizione del robot nel sistema di riferimento globale tramite la telecamera fissa. Si ha:

$$ \xi_F \oplus {}^{F}\xi_R = \xi_R $$

sottraendo da entrambi i membri $\ominus\xi_F$ si ottiene:

$$ \ominus\xi_F \oplus \xi_F \oplus {}^{F}\xi_R = \ominus\xi_F \oplus \xi_R $$

e quindi:

$$ {}^{F}\xi_R = \ominus\xi_F \oplus \xi_R \tag{1.3} $$

si dimostra inoltre che:

$$ {}^{F}\xi_O = \ominus {}^{O}\xi_F \tag{1.4} $$

La trasformazione ξ può essere rappresentata da un punto di vista algebrico in diversi modi, in questo lavoro verrà utilizzata la rappresentazione matriciale.
Consideriamo il punto ${}^{B}\tilde{\mathbf{p}}$ in coordinate omogenee, l'equazione 1.1 si può scrivere:

$$ {}^{A}\tilde{\mathbf{p}} = {}^{A}T_{B} \cdot {}^{B}\tilde{\mathbf{p}} \tag{1.5} $$

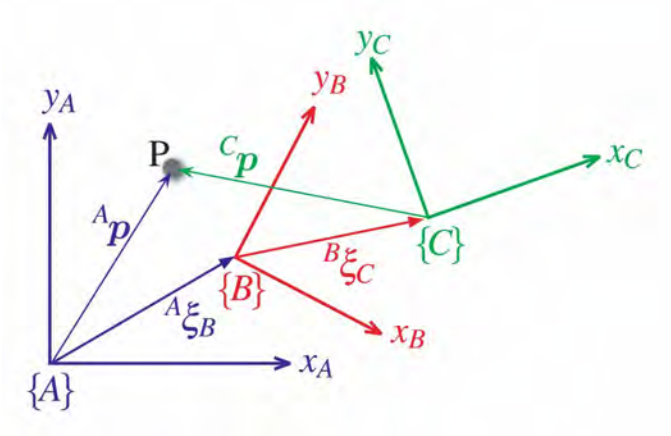


Figura 1.2: Posizione relativa fra sistemi di riferimento.

dove ${}^A T_B$ è una matrice 4×4 che trasforma i punti di $\{B\}$ in $\{A\}$ detta anche *trasformazione omogenea*. Appartiene allo spazio euclideo di dimensione 3 ovvero $T \in SE(3) \subset \mathbb{R}^{4\times4}$. La matrice T può essere scomposta in una matrice di pura traslazione e in una matrice di pura rotazione:

$$\mathbf{T} = \mathbf{T}_{trasl} \cdot \mathbf{T}_{rot} \tag{1.6}$$

dove la matrice di pura traslazione è:

$$\mathbf{T}_{trasl} = \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{t}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.7}$$

con x, y, z componenti del vettore spostamento $\mathbf{t}$ e $\mathbf{I}$ matrice identità. La matrice di pura rotazione è invece:

$$\mathbf{T}_{rot} = \begin{pmatrix} \mathbf{R}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{pmatrix} = \begin{pmatrix} rot_{11} & rot_{12} & rot_{13} & 0 \\ rot_{21} & rot_{22} & rot_{23} & 0 \\ rot_{31} & rot_{32} & rot_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.8}$$

Il minore $\mathbf{R}_{3\times3}$rappresenta la rotazione del sistema di riferimento rispetto al sistema di riferimento globale e le sue colonne sono composte dalle coordinate dei tre versori che compongono la terna. La matrice $\mathbf{R}_{3\times3}$ è quindi ortonormale (appartiene allo spazio ortornormale di dimensione 3 ovvero $\mathbf{T} \in SO(3) \subset \mathbb{R}^{4\times4}$) perciò gode della proprietà $\mathbf{R}^{-1} = \mathbf{R}^T$ e $det(\mathbf{R}) = 1$.

Una rotazione θ attorno ad un asse x, y o z si esprime rispettivamente con le

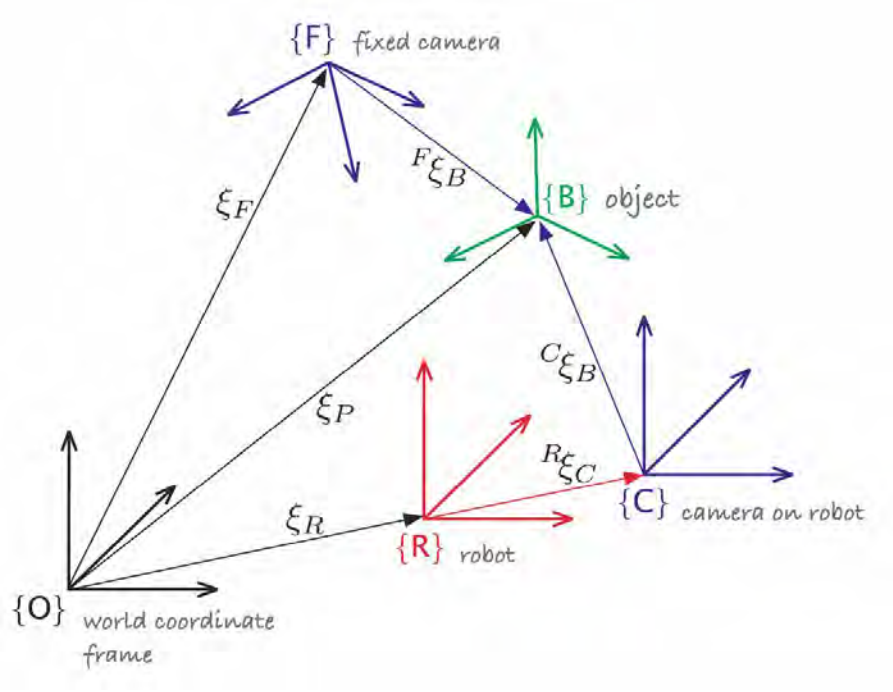


Figura 1.3: Sistemi di riferimento 3D multipli e posizione relativa fra di essi. *Fonte*: Peter Corke, *"Robotic, Vision and Control"*.

seguenti matrici ortornormali:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \tag{1.9a}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{1.9b}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.9c}$$

Quindi una rotazione attorno ai tre assi può essere espressa moltiplicando le matrici dell'equazione (1.9a). La sequenza con cui si moltiplicano però è molto importante, infatti è possibile dimostrare che invertendo l'ordine con cui le matrici vengono moltiplicate il risultato che si ottiene è diverso, per questo è fondamentale riferirsi a una sequenza prefissata.
Le classi fondamentali di rotazioni sono due: le rotazioni secondo *Eulero* e le rotazioni secondo *Cardano*. In questo lavoro verrà usata sempre la sequenza di *Cardano* altresì detta *roll, pitch and yaw* rappresentata dalla seguente:

$$\mathbf{R} = \mathbf{R}_x(\theta_r)\mathbf{R}_y(\theta_p)\mathbf{R}_z(\theta_y) \tag{1.10}$$

Tornando ora alla figura 1.3 se si volesse ad esempio scrivere l'equazione (1.3)

in forma matriciale si otterrebbe:

$$
\begin{aligned}
{}^{F}\mathbf{T}_{R} &= {}^{O}\mathbf{T}_{F}^{-1} \cdot {}^{O}\mathbf{T}_{R} = \\
&= \begin{pmatrix} \mathbf{R}_{OF} & \mathbf{t}_{OF} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \mathbf{R}_{OR} & \mathbf{t}_{OR} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} = \\
&= \begin{pmatrix} \mathbf{R}_{OF}^{T} & -\mathbf{R}_{OF}^{T}\mathbf{t}_{OF} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \mathbf{R}_{OR} & \mathbf{t}_{OR} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} = \\
&= \begin{pmatrix} \mathbf{R}_{OF}^{T} \cdot \mathbf{R}_{OR} & \mathbf{R}_{OF}^{T} \cdot \mathbf{t}_{OR} - \mathbf{R}_{OF}^{T}\mathbf{t}_{OF} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}.
\end{aligned}
$$

1.2 Generazione dell'immagine

É noto da tempo che un semplice forellino praticato attraverso una parete di una stanza buia (*camera obscura*) è in grado di creare l'immagine invertita dell'ambiente esterno all'interno della stanza stessa (figura 1.4). Questo forellino è denominato *obiettivo Stenopeico* (dal Greco: Piccolo-foro) o *pin-hole* .

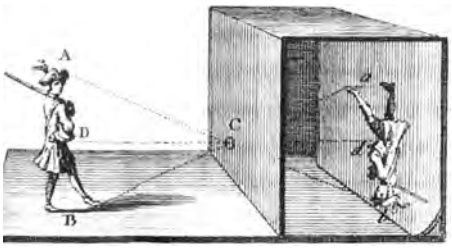


Figura 1.4: Un esempio di *camera obscura* da un illustrazione del *XVII* secolo.

Alcuni molluschi marini, come ad esempio il Nautilus, hanno gli occhi che funzionano secondo questo principio, mentre gli animali vertebrati hanno gli occhi dotati di una lente che forma un'immagine capovolta sulla retina, dove si trovano le cellule fotosensibili che convertono la luce in impulsi che vengono poi inviati al cervello. In linea di principio una fotocamera digitale funziona in modo simile: una lente di vetro o plastica forma un'immagine sulla superficie di un semiconduttore dotato di una serie di dispositivi fotosensibili che convertono la luce in un'immagine digitale.
Il processo di formazione dell'immagine, in un occhio o in una macchina fotografica, comporta quindi la proiezione del mondo 3D su una superficie a 2 dimensioni. Ovviamente però in questo passaggio le informazioni di profondità sono perse e non si è più in grado di stabilire, ad esempio, se un oggetto nell'immagine è un oggetto grande distante o un oggetto piccolo più vicino. Questa trasformazione da 3 a 2 dimensioni è nota come *proiezione prospettica*.

1.2.1 La proiezione prospettica

La figura 1.5 mostra gli aspetti elementari della formazione dell'immagine. L'asse z positivo è l'asse ottico della telecamera. La coordinata z dell'oggetto e l'immagine

da esso formata sono legate dalla seguente legge

$$\frac{1}{z_0} + \frac{1}{z_i} = \frac{1}{f}$$

che è la cosiddetta *"lens law"*; dove z_0 è la distanza dell'oggetto dalla lente, z_i è la distanza del piano dell'immagine dalla lente, e f è la lunghezza focale della lente. Se $z_0 > f$ l'immagine che si formerà, sul piano a distanza $z < f$, risulterà invertita.

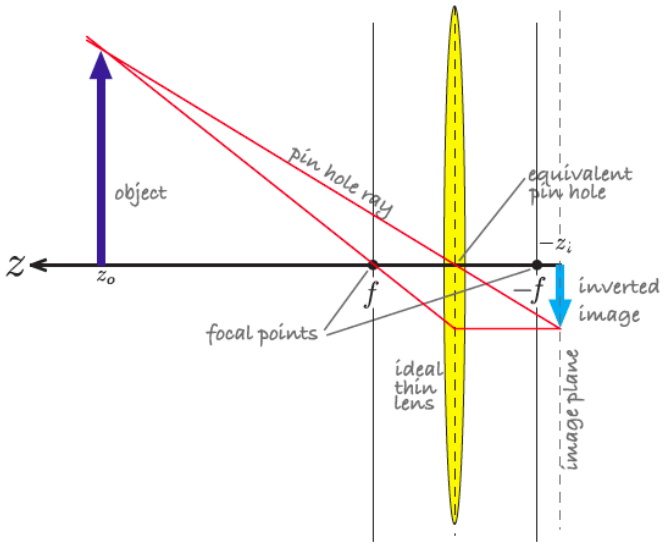


Figura 1.5: Elementi fondamentali della formazione dell'immagine; *fonte*: Peter Corke, *"Robotic, Vision and Control"*.

Il termine fuoco o focale in ottica indica il piano o il punto in cui singoli raggi fotonici formanti un fascio di radiazioni elettromagnetiche distinte e arrivate da un punto all'infinito, si incontrano e quindi si concentrano. La distanza tra il centro ottico della lente ed il piano focale, indica appunto la distanza focale come valore assoluto della lente (es: 50 mm); usata per catalogare anche gli obiettivi fotografici, la focale di un'ottica è sempre riferita alla messa a fuoco all'infinito.
Quasi tutti i vari strumenti ottici creati dall'uomo (macchine fotografiche, cannocchiali, binocoli, telescopi, etc.) utilizzano lenti o specchi per concentrare la luce dell'immagine catturata, direttamente sul piano focale. Quando la regolazione fornisce un risultato nitido, si dice che l'immagine è a fuoco, in caso contrario si dice che è fuori fuoco o sfocata.
Per analogia l'occhio umano funziona proprio come una macchina fotografica, dove la parte anteriore (cornea e cristallino) è simile ad un obiettivo dotato di meccanismi di regolazione della messa a fuoco (utile per distanze da 7 cm all'infinito) e dove la parte posteriore (retina e nervo ottico), funzionano come un sensore fotografico in grado di fornire, a distanza di 25 cm, una risoluzione massima o acutezza visiva (20/10) monoculare di 280 dpi, pari a circa 11 linee per mm. La vista binoculare umana mette a fuoco gli oggetti utilizzando due sistemi contemporaneamente: la telemetria a sdoppiamento d'immagine, basata sulla distanza tra le due assi ottiche

(parallasse) e l'accomodamento focale dei piani prioettati sulla rétina.
Nella fotocamere la messa a fuoco viene attuata allontanando o avvicinando opportunamente, sull'asse ottico, le lenti dell'obiettivo, in modo che il gruppo ottico si trovi alla distanza z_i. In questo modo è possibile mettere a fuoco qualsiasi piano immagine ripreso tra la distanza minima e l'infinito, proiettandolo nitidamente sul piano focale del sensore (lastra, pellicola, etc.). Per un oggetto all'infinito $z_i = f$.
L'allontanamento dell'ottica dal Sensore, provoca la messa a fuoco di oggetti sempre più vicini alla fotocamera: i tubi di prolunga e i soffietti posti tra corpo macchina e ottica, vengono utilizzati appunto in macrofotografia, per ingrandire i soggetti riducendo la minima distanza di messa a fuoco.
L'obiettivo stenopeico funziona senza lenti ottiche e senza l'ausilio di regolazioni di messa a fuoco, producendo un'immagine totalmente a fuoco (o quasi). Precisamente, più piccolo è il diametro del foro e più piccoli saranno i circoli di confusione dei punti immagine, rendendo più nitida o a fuoco la fotografia.
Una lente quindi ha la necessità di mettere a fuoco l'immagine, però con il compromesso di poter convogliare una maggior quantità di luce.

In *computer vision* si utilizza il modello di *central perspective imaging* rappresentato in figura 1.6. I raggi convergono sull'origine del sistema di riferimento della fotocamera $\{C\}$ e l'immagine, non invertita, viene proiettata sul piano dell'immagine situato a $z = f$. Utilizzando la similitudine fra triangoli si può dimostrare che la

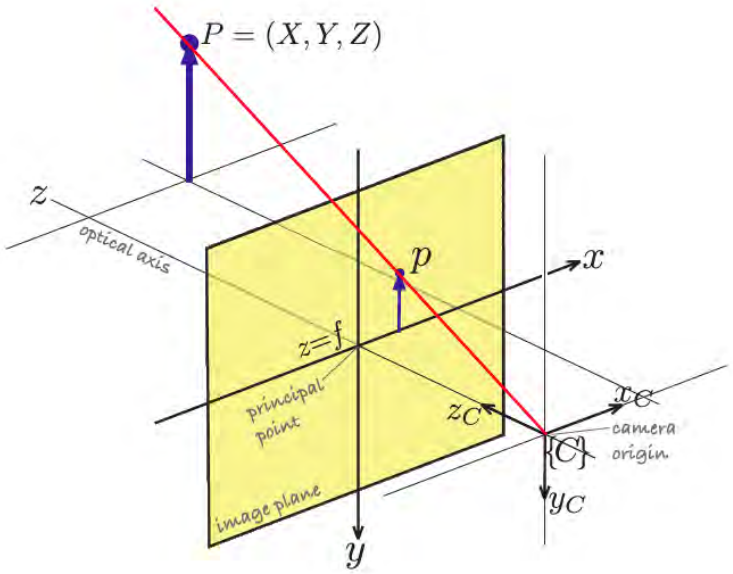


Figura 1.6: Modello central perspective imaging; *fonte*: Peter Corke, *"Robotic, Vision and Control"*.

proiezione di un punto $\mathbf{P} = (X, Y, Z)$, in coordinate globali, nel piano dell'immagine è data dal punto $\mathbf{p} = (x, y)$ tramite la seguente relazione:

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z} \tag{1.11}$$

che rappresenta la trasformazione prospettica, dall'oggetto reale all'immagine, ed ha le seguenti caratteristiche:

- Esegue una trasformazione dallo spazio tridimensionale al piano dell'immagine bidimensionale, $\mathbb{R}^3 \mapsto \mathbb{R}^2$;
- Linee rette restano proiettate come linee rette;
- Linee parallele vengono proiettate come linee che si intersecano in un punto di fuga o *vanishing point*, questo è valido sempre eccetto per linee in un piano parallelo al piano dell'immagine che non convergono.
- Coniche sono proiettate come coniche nel piano immagine. Ad esempio un cerchio viene proiettato come un'ellisse o al limite come un cerchio nel caso si trovi in un piano parallelo al piano dell'immagine;
- La trasformazione inversa non è univoca, cioè dato il punto proiettato $\mathbf{p} = (x, y)$ non è possibile determinare un unico punto $\mathbf{P} = (X, Y, Z)$, ma si può solamente affermare che il punto $\mathbf{P}$ cercato si trova lungo la semiretta $\overline{OP}$ come rappresentato in figura 1.6;
- La trasformazione non è conforme, cioè non conserva le forme dato che gli angoli interni non vengono conservati. Una traslazione, rotazione o una riduzione in scala sono esempi di trasformazione conforme. Una generica trasformazione affine non conforme comprende traslazioni, rotazioni e cambiamenti di scala differenti per ogni asse.

Esprimiamo ora il punto proiettato sul piano dell'immagine, in forma omogenea $\tilde{\mathbf{p}} = (x', y', z')$ dove:

$$x' = \frac{fX}{z'}, \quad y' = \frac{fY}{z'}, \quad z' = Z \tag{1.12}$$

oppure in forma matriciale:

$$\tilde{\mathbf{p}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{1.13}$$

le coordinate in forma non omogenea sono ovviamente:

$$x = \frac{x'}{z'}, \quad y = \frac{y'}{z'}.$$

Se esprimiamo il punto $\mathbf{P}$ in coordinate di fotocamera e in forma omogenea, cioè ${}^C\tilde{\mathbf{P}} = (X, Y, Z, 1)^T$, la trasformazione prospettica si può scrivere come:

$$\tilde{\mathbf{p}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot {}^C\tilde{\mathbf{P}} \tag{1.14}$$

ovvero

$$\tilde{\mathbf{p}} = \mathbf{C} \cdot {}^C\tilde{\mathbf{P}} \tag{1.15}$$

dove la matrice $\mathbf{C}$ è una matrice 3×4 detta anche *camera matrix* e ${}^C\tilde{\mathbf{P}}$ rappresenta le coordinate omogenee del punto rispetto al sistema di riferimento della fotocamera

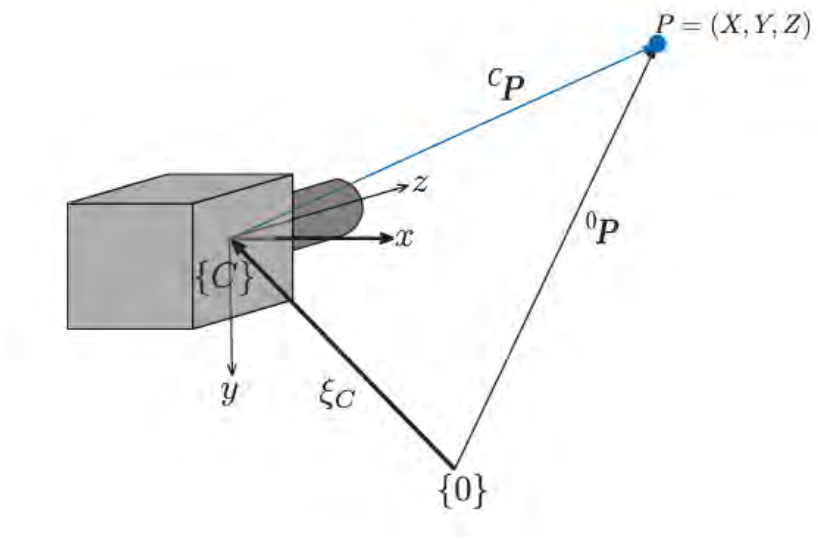


Figura 1.7: Sistema di riferimento della fotocamera; *fonte*: Peter Corke, *"Robotic, Vision and Control"*.

$\{C\}$. [...quanto segue: possibile nota] La tilde rappresenta una quantità omogenea, vedi appendice [...]. La matrice $\mathbf{C}$ può essere scomposta nel modo seguente

$$\tilde{\mathbf{p}} = \mathbf{C} \cdot {}^{C}\tilde{\mathbf{P}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot {}^{C}\tilde{\mathbf{P}}$$

dove la seconda matrice è detta *projection matrix*.

La fotocamera, in generale, avrà una posizione ξ_C arbitraria rispetto al sistema di riferimento globale, figura 1.7. La posizione del punto $\mathbf{P}$ rispetto alla camera è

$${}^{C}\mathbf{P} = (\ominus \xi_C) \cdot {}^{0}\mathbf{P} = {}^{C}\mathbf{T}^{-1} \cdot {}^{0}\mathbf{P} \tag{1.16}$$

In una fotocamera digitale il piano dell'immagine è composto da una griglia $W \times H$ di elementi foto-sensibili che corrispondono ai pixel dell'immagine, come mostrato in figura 1.8. Le coordinate dei pixel sono espresse dal vettore (u, v), chiaramente formato da numeri interi positivi; l'origine, per convenzione, si trova nell'angolo in altro a sinistra dell'immagine. Le coordinate dei pixel sono riferite alle coordinate del piano dell'immagine dalle seguenti relazioni

$$u = \frac{x}{\rho_w} + u_0, \quad v = \frac{y}{\rho_h} + v_0$$

dove:

- ρ_w e ρ_h sono rispettivamente la larghezza e l'altezza del pixel;
- (u_0, v_0) sono le coordinate del *principal point*, ovvero le coordinate del punto dove l'asse ottico interseca il piano dell'immagine.

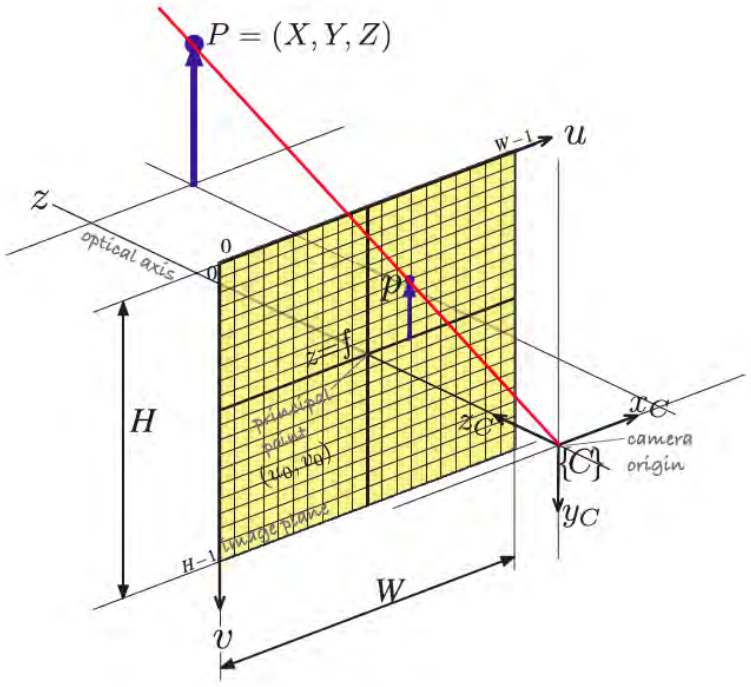


Figura 1.8: Modello central perspective imaging con matrice dei pixel sul piano dell'immagine; *fonte*: Peter Corke, *"Robotic, Vision and Control"*.

É possibile quindi esprimere le coordinate del punto, proiettato sul sensore, in pixel premoltiplicando l'equazione (1.14) per una matrice $\mathbf{K}$ che contiene i parametri della fotocamera

$$\tilde{\mathbf{p}} = \underbrace{\begin{bmatrix} 1/\rho_w & 0 & u_0 \\ 0 & 1/\rho_h & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot {}^{C}\tilde{\mathbf{P}} \tag{1.17}$$

dove $\tilde{\mathbf{p}} = (u', v', w')$ sono le coordinate omogenee del punto $\mathbf{P}$ in pixel. Le coordinate non omogenee sono ovviamente

$$u = \frac{u'}{w'}, \quad v = \frac{v'}{w'}$$

Combinando le equazioni (1.16) e (1.17) si ottiene l'*equazione generale della proiezione prospettica*:

$$\begin{aligned} \tilde{\mathbf{p}} &= \underbrace{\begin{bmatrix} f/\rho_w & 0 & u_0 \\ 0 & f/\rho_h & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{intrinseci}} \underbrace{({}^{O}\mathbf{T}_{C})^{-1}}_{\text{estinseci}} \tilde{\mathbf{P}} \\ &= \mathbf{K}\mathbf{P}_{\mathbf{0}}{}^{\mathbf{0}}\mathbf{T}_{\mathbf{C}}^{-\mathbf{1}}\tilde{\mathbf{P}} \\ &= \mathbf{C}\tilde{\mathbf{P}} \end{aligned} \tag{1.18}$$

dove tutti i termini sono contenuti nella matrice 3×4 della fotocamera, o *camera matrix*, $\mathbf{C}$ che rappresenta una trasformazione omogenea che effettua uno scaling,

una traslazione e una proiezione prospettica. La matrice $\mathbf{C}$ è spesso indicata anche come matrice di calibrazione della fotocamera.

Abbiamo già menzionato all'inizio del paragrafo qual è il problema fondamentale della proiezione prospettica: ovvero l'ambiguità nel distinguere fra un oggetto distante di grandi dimensioni e un oggetto vicino di piccole dimensioni. Possiamo riscrivere l'equazione (1.18) come:

$$\tilde{\mathbf{p}} = (\mathbf{C}\mathbf{H}^{-\mathbf{1}})(\mathbf{H}\tilde{\mathbf{P}}) = \mathbf{C}'\tilde{\mathbf{P}}'$$

dove $\mathbf{H}$ è una matrice arbitraria 3×3 non singolare. Questo implica che esistono infinite combinazioni di fotocamere $\mathbf{C}'$ e di punti $\tilde{\mathbf{P}}'$ che possono dare la stessa proiezione $\tilde{\mathbf{p}}$. Si comprende, quindi, come sia molto colplesso determinare le coordinate di un punto nello spazio, partendo dalla sua proiezione prospettica su un piano bi-dimensionale. Questo problema può essere risolto solo se:

- sono note tutte le informazioni sulla fotocamera, ovvero è nota la matrice $\mathbf{C}$;
- sono note le dimensioni dell'oggetto 3-dimensionale proiettato.

La proiezione può essere scritta anche come:

$$p = \mathcal{P}(\mathbf{P}, \mathbf{K}, \xi_C) \tag{1.19}$$

dove $\mathbf{P}$ è il punto nello spazio, $\mathbf{K}$ è la matrice dei parametri della fotocamera, e ξ_C è la posizione della fotocamera rispetto al sistema di riferimento globale. I parametri della fotocamera contenuti nella matrice $\mathbf{K}$ comprendono f, ρ_w, ρ_h, u_0 e v_0 e sono detti parametri intrinseci perchè rappresentano le caratteristiche insite della fotocamera e del sensore, mentre i parametri cosiddetti estrinseci descrivono la posizione della telecamera e comprendono un minimo di sei parametri per descrivere le traslazioni e le rotazioni nel sistema di riferimento globale. Ci sono quindi un totale di 11 parametri, la matrice della fotocamera ha 12 elementi per cui un grado di liberta, il fattore di scala, resta vincolato. In pratica questi parametri non sono noti e devono essere stimati sulla base di una procedura di calibrazione della telecamera che verrà trattata in 1.3.

Il campo di visuale o *field of view* di una fotocamera è funzione della lunghezza focale f. Una lente con un ampio angolo di visuale avrà una lunghezza focale corta, una lente teleobiettivo avrà una lunghezza focale lunga, mentrer un obiettivo zoom avrà una lunghezza focale variabile. Il campo di visuale può essere determinato in riferimento alla figura 1.8.
In direzione orizzontale l'angolo di visuale è:

$$\theta_h = 2\tan^{-1}\frac{W\rho_w}{2f}, \quad \theta v = 2\tan^{-1}\frac{H\rho_h}{2f} \tag{1.20}$$

si può notare inoltre che il campo di visuale è funzione anche delle dimensioni del sensore, che sono date da $W\rho_w \times H\rho_h$.

(a) Distorsione a botte **(b)** Distorsione a cuscino

Figura 1.9: Esempio di immagini distorte

1.2.2 Distorsione lenticolare

Nessuna lente è perfetta e le lenti economiche usate in molte fotocamere sono ben lontane da essere perfette. Le imperfezioni delle lenti producono distorsioni come l'aberrazione cromatica, l'aberrazione sferica o astigmatismo (variazione della messa a fuoco attraverso la scena), e distorsioni geometriche che provocano uno spostamento del punto proiettato rispetto a dove si dovrebbe trovare in accordo con l'equazione (1.13).
La distorsione geometrica è generalmente l'effetto più problematico che si può incontrare nelle fotocamere e comprende due componenti: radiale e tangenziale.
La distorsione radiale provoca una traslazione, dei punti proiettati sul piano dell'immagine, lungo una semiretta radiale che ha origine nel *principal point*. L'errore provocato dalla distorsione radiale può essere ben approssimato dal polinomio

$$\delta r = k_1 r^3 + k_2 r^5 + k_3 r^7 + \dots \tag{1.21}$$

dove r è la distanza del punto proiettato sul piano dell'immagine e il *principal point*.
La distorsione radiale si verifica in due modi:

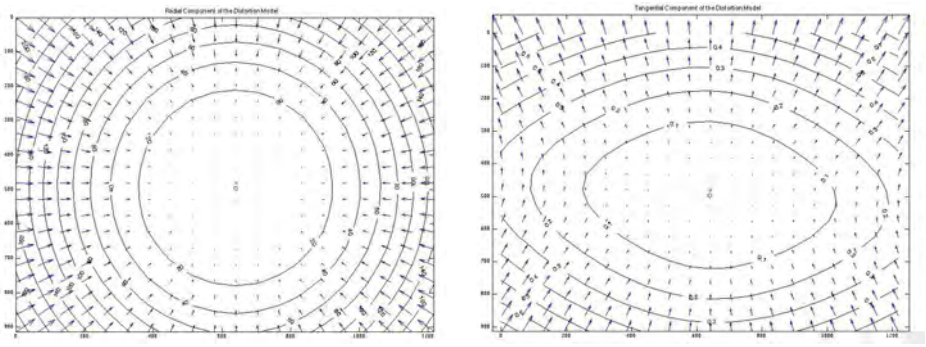
barrel distortion o distorsione a botte si verifica quando l'ingrandimento della lente diminuisce con l'aumentare della distanza dal punto principale. In presenza di questo effetto, linee rette verticali che si trovano vicino al bordo dell'immagine appariranno curve verso l'esterno.

pincushion distortion o distorsione a cuscino si verifica quando, al contrario, l'ingrandimento aumenta con l'aumentare della distanza dal punto principalee quindi, in presenza di questo effetto, linee rette verticali che si trovano vicino al bordo dell'immagine appariranno curve verso l'interno.

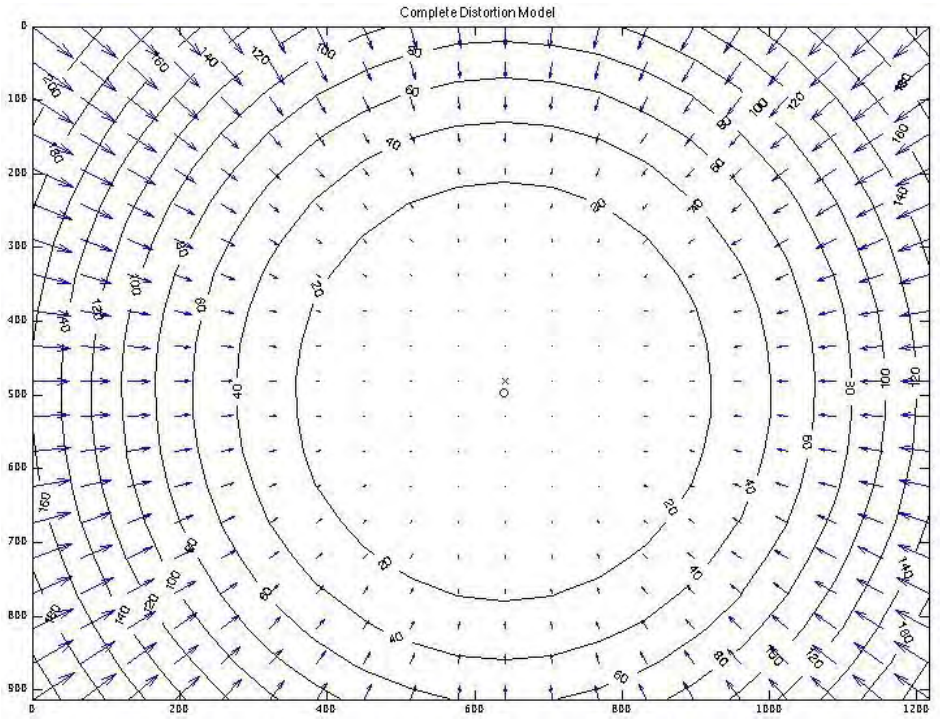
La distorsione tangenziale, o distorsione decentrante, si verifica in direzione ortogonale alla direzione radiale e solitamente è molto bassa.

Le coordinate dei punti (u, v) distorti sono dati da

$$u^d = u + \delta_u, \quad v^d = v + \delta_v \tag{1.22}$$



(a) Distorsione radiale. **(b)** Distorsione tangenziale



(c) Distorsione completa.

Figura 1.10: Distorisione della lente: vettori spostamento.

dove gli spostamenti sono

$$\begin{pmatrix}\delta_u\\ \delta_v\end{pmatrix} = \underbrace{\begin{pmatrix}u(k_1r^2+k_2r^4+k_3r^6+\dots)\\ v(k_1r^2+k_2r^4+k_3r^6+\dots)\end{pmatrix}}_{\text{radial}} + \underbrace{\begin{pmatrix}2p_1uv+p_2(r^2+2u^2)\\ p_1(r^2+2v^2)+2p_1uv\end{pmatrix}}_{\text{tangential}} \tag{1.23}$$

In figura 1.10 sono plottati i vettori spostamento dati dalla (1.23). I vettori indicano lo spostamento necessario per correggere la distorsione in diversi punti nell'immagine e indicano inoltre la distorsione radiale dominante. Tipicamente sono sufficienti tre coefficienti per descrivere la distorsione radiale (k_1, k_2, k_3); quindi il modello che rappresenta la distorsione della camera è parametrizzato da: $(k_1, k_2, k_3, p_1, p_2)$ che sono considerati parametri intrinseci aggiuntivi della camera.

1.3 Calibrazione della fotocamera

Il modello della prioiezione prospettica rappresentato dall'equazione (1.18), ha un certo numero di parametri che non sono noti. In genere il *principal point* non si trova al centro del sensore, la lunghezza focale di un obiettivo ha un errore di $\pm 4\%$ sul valore dichiarato dal costruttore, ed è corretta solo se l'obiettivo è focalizzato all'infinito. É risaputo inoltre che i parametri intrinseci variano se ad esempio una lente viene smontata e rimontata, nonchè durante gli interventi di regolazione della messa a fuoco e di apertura del diaframma. Gli unici parametri intrinseci che è possibile ottenere direttamente sono le dimensioni dei pixel dell'elemento fotosensibile ρ_u e ρ_v tramite le informazioni fornite dal costruttore. I parametri estrinseci, cioè la posa della fotocamera, indicano dove si trova esattamente il punto centrale della fotocamera rispetto al sistema di riferimento globale.

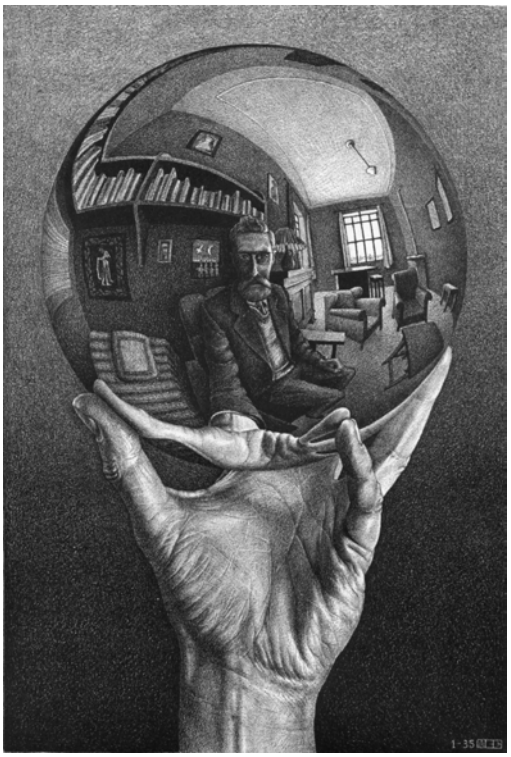


Figura 1.11: *Mano con sfera riflettente*, di *M. Escher*.

La calibrazione della fotocamera è il processo di determinazione dei parametri intrinseci ed estrinseci della telecamera. Le tecniche di calibrazione si basano sull'acquisizione, con la fotocamera da calibrare, di punti reali nello spazio le cui coordinate relative spaziali sono note e sull'analisi delle proiezioni di questi punti ottenute nelle immagini acquisite. Lo stato dell'arte della tecnica per la calibrazione delle fotocamere è rappresentato dal *Bouguet Camera Calibration Toolbox for MATLAB® di Jean-Yves Bouguet*, che richiede semplicemente una serie di immagini di una scacchiera calibrata tramite le quali è possibile stimare parametri intrinseci (compresi i parametri di distorsione) e la posa della scacchiera rispetto alla fotocamera in ogni immagine. Le tecniche di calibrazione classiche richiedono una singola vista di un bersaglio 3-dimensionale calibrato ma non sono in grado di stimare, tuttavia, il modello distorsivo della fotocamera.

In questo lavoro verranno utilizzate due telecamere:

- **GoPro HD Hero** in modalità R4 che offre una risoluzione di 1280×960 pixel a 30 fps (figura 1.12a);
- **GoPro Hero3+ black edition** in modalità 1440p che offre una risoluzione di 1920×1440 pixel a 48 fps (figura 1.12b).

Queste telecamere appartengono alla famiglia delle *action cameras* e offrono un elevato campo visivo che raggiunge i 170°. Nella prossima sezione si vedrà il procedimento utilizzato per determinare i parametri intrinseci ed estrinseci di queste due telecamere.

(a) (b)

Figura 1.12: GoPro HD Hero (a); GoPro Hero 3+ (b)

1.3.1 Bouguet Camera Calibration Tool

Il *Bouguet Camera Calibration Toolbox* è uno strumento per la calibrazione di fotocamere che utilizza un target planare sul quale è impressa la trama di una scacchiera calibrata composta da quadrati dalle dimensioni note. Il toolbox richiede una serie di immagini della scacchiera, catturate con la fotocamera da calibrare, da diverse posizioni e angolazioni. Tipicamente venti immagini sono sufficienti. Il toolbx si lancia con il seguente comando

```
>> calib_gui
```

Comparirà a display una finestra del tipo *graphical user interface (GUI)*, in figura 1.13.

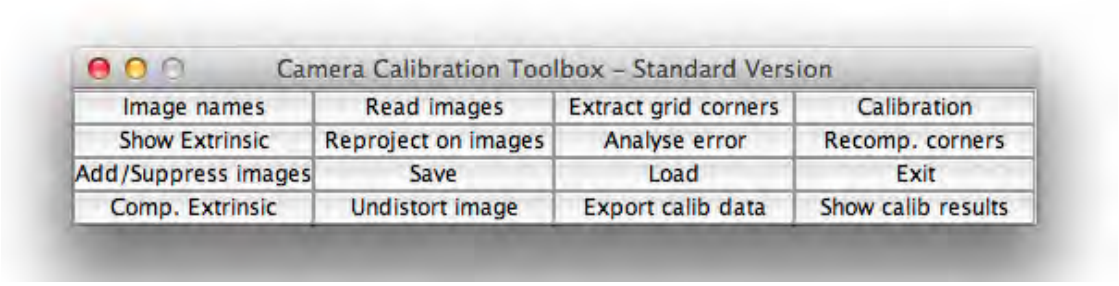


Figura 1.13: Camera calibration toolbox GUI.

Il primo step consiste nel caricare le immagini tramite il pulsante Image Names. Le immagini dovranno essere numerate e avere lo stesso formato, ad esempio:

```
image01.jpg, image02.jpg, image03.jpg, ...
```

e quindi l'intestazione `"image"` dovrà essere comune per tutte. Il set completo di immagini, una volta caricate, verrà plottato a schermo (figura 1.14)

Il secondo step consiste nell'individuare gli angoli della scacchiera in ogni immagine, il comando si lancia cliccando sul pulsante Extract grid corners.
Ogni immagine caricata verrà plottata a display una alla volta e l'utente dovrà cliccare in cirrispondenza dei quattro angoli della scacchiera. Solitamente si parte dall'angolo più in alto a sinistra e si procede nell' individuazione dei successivi in senso orario. Il primo angolo che verrà individuato rappresenta l'origine del sistema di riferimento della scacchiera. Quindi, una volta lanciato il comando, compariranno a schermo le seguenti istruzioni:

Figura 1.14: Immagini caricate

```
>> Extraction of the grid corners on the images
Number(s) of image(s) to process ([] = all images) =
Window size for corner finder (wintx and winty):1
wintx ([] = 15) =
winty ([] = 15) =
```

e verra richiesto di inserire le dimensioni del riquadro `(wintx, winty)` entro il quale il software cercherà l'angolo. Nel caso della calibrazione della GoPro sono partito inserendo una dimensione del riquadro abbastanza elevata `(wintx, winty)=(15,15)` il motivo verrà spiegato più avanti.
A questo punto si procede all'identificazione manuale per ogni immagine degli angoli della scacchiera (figura 1.15).

Una volta identificati i corner dell'immagine verranno richieste le dimensioni in millimetri dei riquadri della scacchiera `dX, dY` e il numero di riquadri lungo la direzione x e y (quest'ultima selezione è opzionale, il toolbox può anche riconoscere automaticamente i riquadri). L'operazione si ripete poi per tutte le immagini successive.

Molto spesso i punti in cui il codice prevede che si trovino gli angoli, non sono abbastanza vicini agli angoli dell'immagine reale per consentire un estrazione dell'angolo efficace, come in figura 1.16. Questo si verifica a maggior ragione se la fotocamera ha una lente con un'elevata distorsione, come nel caso della GoPro. In questo caso è necessario inserire manualmente un coefficiente correttivo `kc` che tiene conto della distorsione.

```
>> If the guessed grid corners (red crosses on the image) are not close to
the actual corners,
```

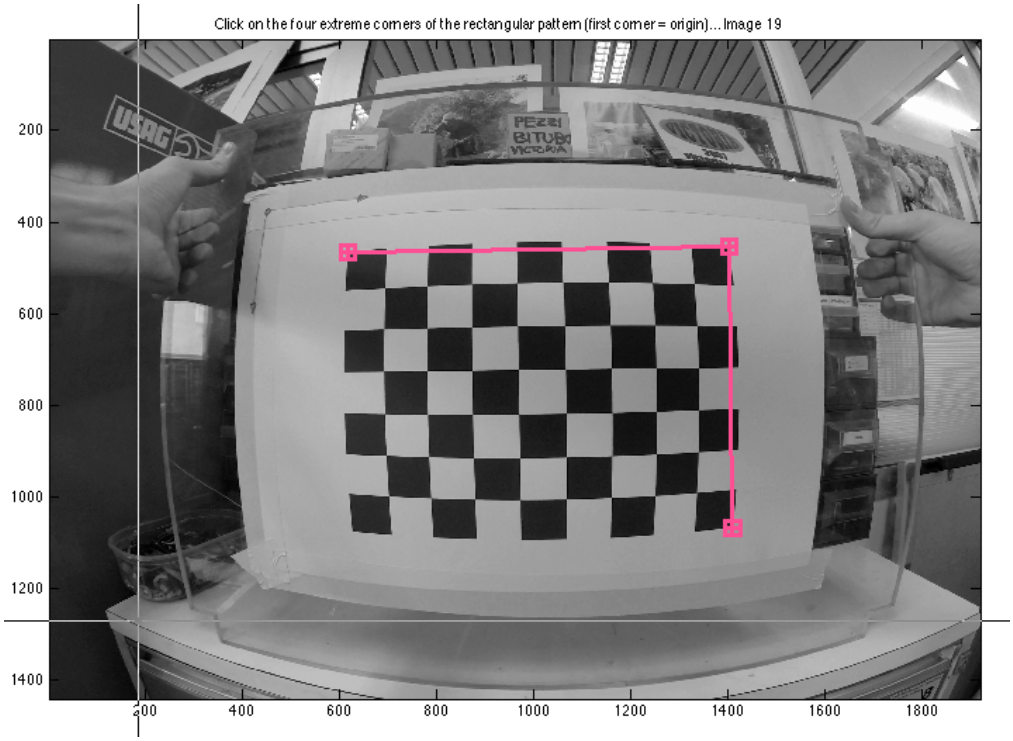


Figura 1.15: Estrazione corner: individuazione manuale degli angoli

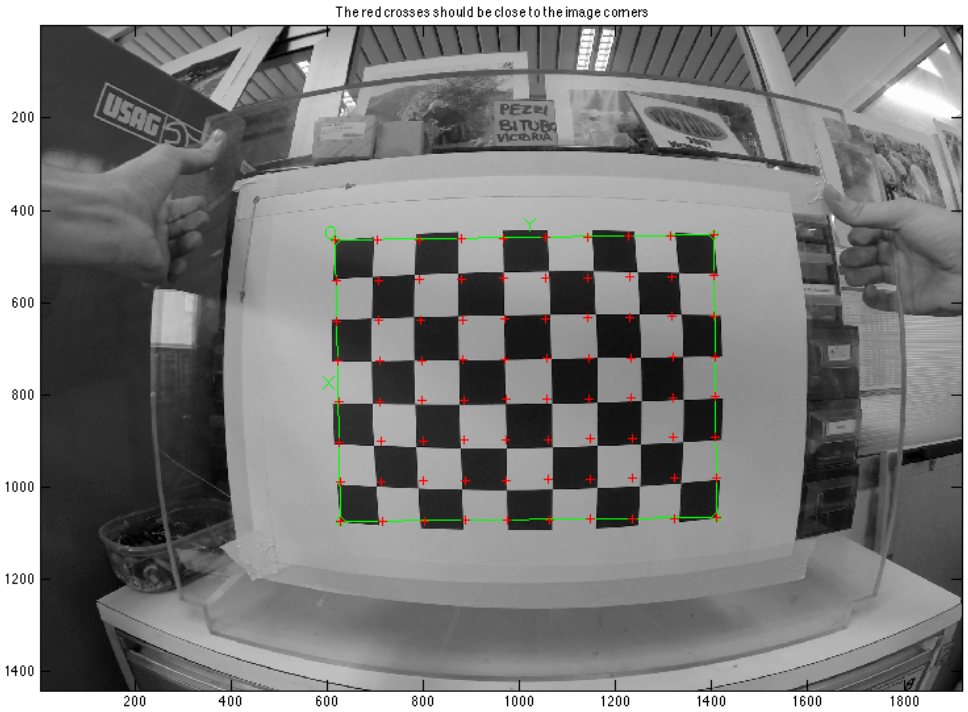


Figura 1.16: Estrazione corner: corner previsti dal codice senza coefficiente correttivo `kc`.

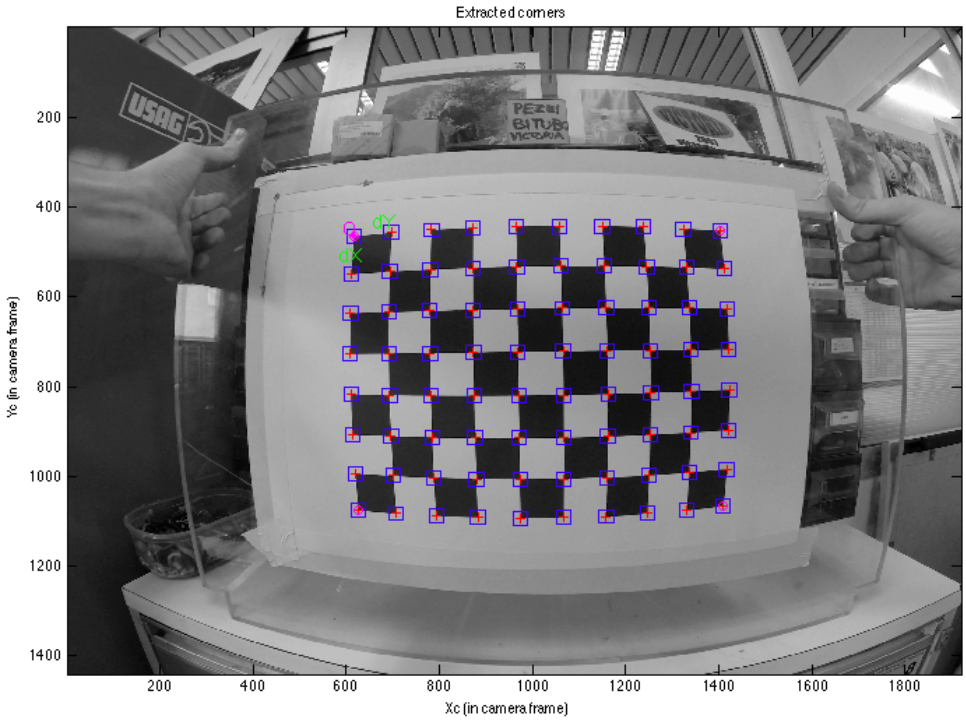


Figura 1.17: Corner estratti

```
it is necessary to enter an initial guess for the radial distortion factor kc
(useful for subpixel detection)
Need of an initial guess for distortion?  ([]=no, other=yes)
Use number of iterations provided
Use focal provided
Estimated focal:  5592.3749 pixels
Guess for distortion factor kc ([]=0):
```

Inserito un opportuno coefficiente `kc` il codice è in grado di riconoscere gli angoli ed estrarli. (figura 1.17) Il quadratino blu che compare in ogni angolo estratto rappresenta il riquadro, le cui dimensioni `(wintx, winty)` sono state date nel passaggio precedente, entro il quale il software andrà a cercare il corner. Nel caso della GoPro la distorsione è talmente elevata, soprattutto vicino ai bordi, che ho dovuto scegliere un riquadro di dimensioni elevate per riuscire ad individuare i corner, dato che il coefficiente `kc` in certi casi non era sufficiente. Una volta estratti i corner di tutte le immagini si procede alla calibrazione, tramite il punsante Calibration. Terminato il processo verranno plottati i risultati:

```
>> Calibration results after optimization (with uncertainties):
Focal Length:  fc = [857.51; 860.36] ± [2.99; 2.9]
Principal point:  cc = [984.92; 729.51] ± [2.31; 2.31]
Skew:  alpha_c = [0.00] ± [0.00] => angleofpixelaxes = 90.00 ± 0.00degrees
Distortion:  kc = [−0.23113; 0.04742; −0.00007; −0.00057; 0.00000] ± [0.00143
       0.00088; 0.00020; 0.00018; 0.00000]
Pixel error:  err = [0.711; 0.563]
Note:  The numerical errors are approximately three times the standard
```

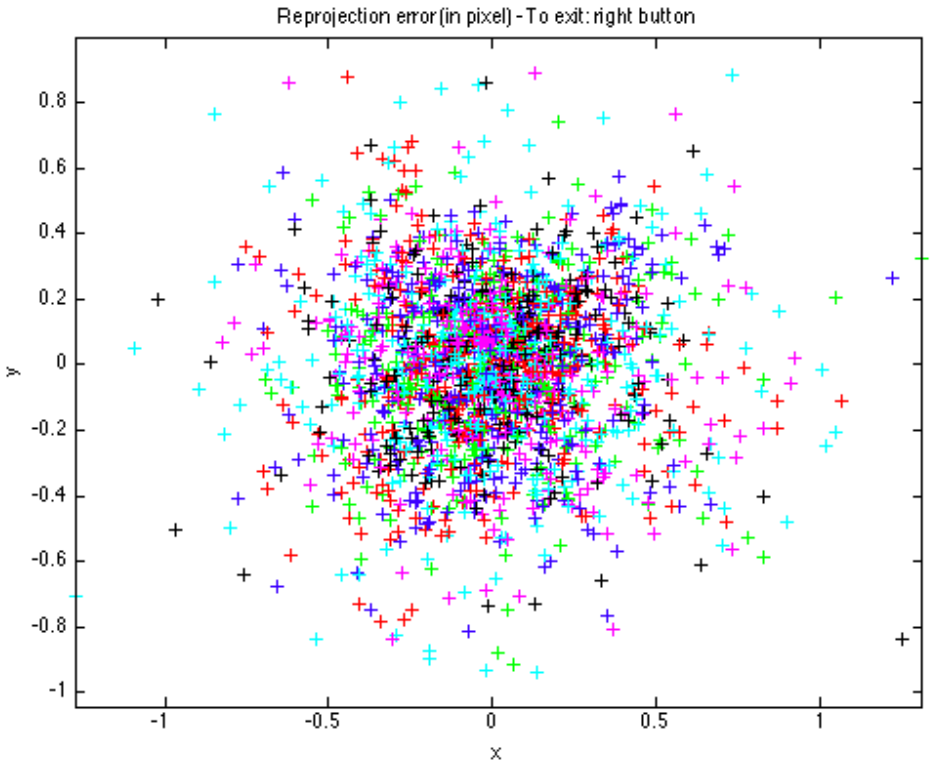


Figura 1.18: Pixel error

`deviations (for reference).`

Una volta effettuata la calibrazione è utile analizzare l'errore con il comando Analyse error che plotterà un grafico contentente gli errori in pixel delle immagini elaborate (figura 1.18) Se l'errore risulta troppo elevato è possibile ridurlo tramite il comando Recomp. corners che permette di riselezionare gli angoli della scacchiera ed effettuare una estrazione migliore dei corners, per le immagini che presentano un errore eccessivo. In questa fase solitamente si utilizza un riquadro più piccolo rispetto a quello iniziale.

Fatto ciò si ri-calibra il set di immagini, nel caso della **GoPro Hero 3+** si sono ottenuti i seguenti risultati:

```
>> Calibration results after optimization (with uncertainties):
Focal Length:  fc = [841.12; 843.11] ± [1.61; 1.59]
Principal point:  cc = [969.65; 722.93] ± [1.16; 1.04]
Skew:  alpha_c = [0.00] ± [0.00] => angleofpixelaxes = 90.00 ± 0.00degrees
Distortion:  kc = [−0.25316; 0.06659; −0.00011; −0.00014; 0.00000] ± [0.00098
       0.00075; 0.00010; 0.00010; 0.00000]
Pixel error:  err = [0.293; 0.262]
Note:  The numerical errors are approximately three times the standard
deviations (for reference).
```

Mentre per la **GoPro Hero 1** si è ottenuto:

```
>> Calibration results after optimization (with uncertainties):
```

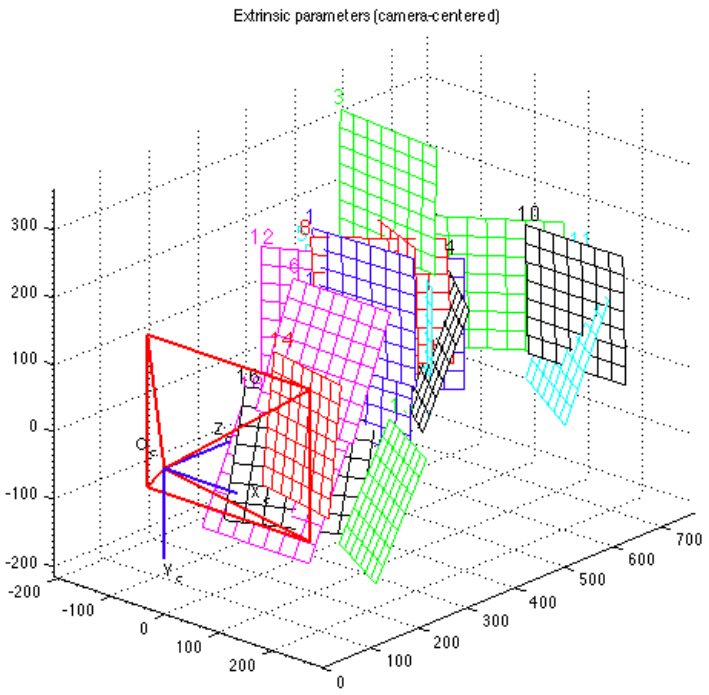


Figura 1.19: Parametri estrinseci: sistema di riferimento fotocamera.

`Focal Length:` $\texttt{fc} = [570.55; 571.32] \pm [4.49; 4.17]$
`Principal point:` $\texttt{cc} = [648.21; 495.63] \pm [3.89; 4.77]$
`Skew:` $\texttt{alpha}_c = [0.00] \pm [0.00]$ => `angleofpixelaxes` $= 90.00 \pm 0.00$`degrees`
`Distortion:` $\texttt{kc} = [-0.27956; 0.06391; -0.00045; -0.00054; 0.00000] \pm [0.00354$
$0.00225; 0.00084; 0.00054; 0.00000]$ `Pixel error:` $\texttt{err} = [0.413; 0.387]$
`Note: The numerical errors are approximately three times the standard deviations (for reference).`

Per quest'ultima si nota che l'errore è leggermente più elevato rispetto alla prima. Questo è dovuto al fatto che le immagini utilizzate per la calibrazione, che non erano altro che dei frame estratti da un filmato, non avevano una buona definizione e quindi i bordi dei riquadri della scacchiera risultavano poco definiti. Al contrario, le immagini estratte dal filmato fatto con la GoPro Hero 3+ presentavano un'ottima definizione e di conseguenza anche i bordi dei riquadri della scacchiera.

Il comando Show extrinsic permette di visualizzare i parametri estrinseci ovvero la posizione relativa della scacchiera rispetto alla camera (O_C, X_C, Y_C, Z_C) rappresentati nella figura 1.19. La piramide rossa rappresenta il campo di visuale (FOV) effettivo della fotocamera e come si puo notare, entrambe le GoPro hanno un campo di visuale molto ampio, che arriva anche fino a 170°.

In figura 1.20 sono plottati i parametri estrinseci rispetto al sistema di riferimento globale. Un FOV elevato come quello della GoPro comporta un elevata distorsione delle immagini, come si può ben notare nell figura 1.21a. Tramite il comando Undistort image è possibile rimuovere la distorsione della lente dalle immagini catturate con la fotocamera calibrata; in figura 1.21b ne è rappresentato un esempio. Come si puo notare, però, l'immagine non risulta perfettamente priva di distorsioni

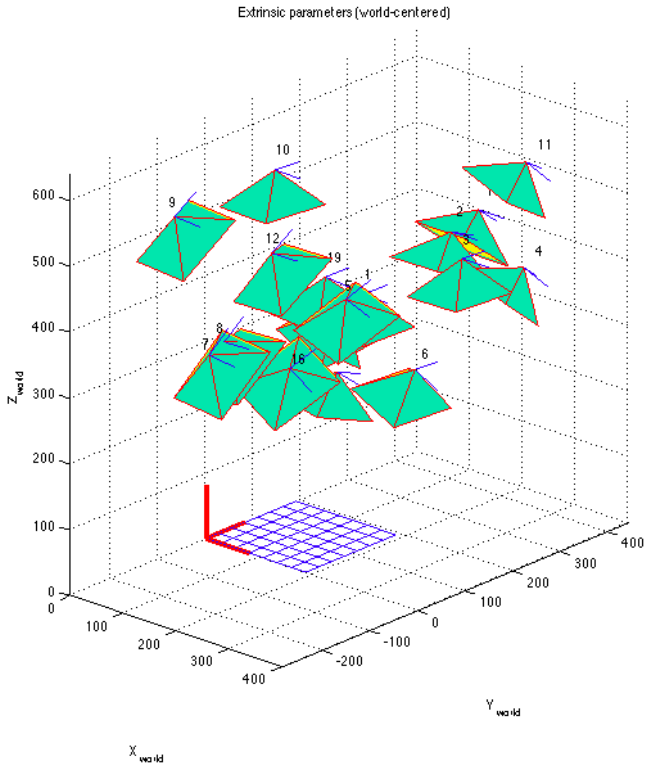
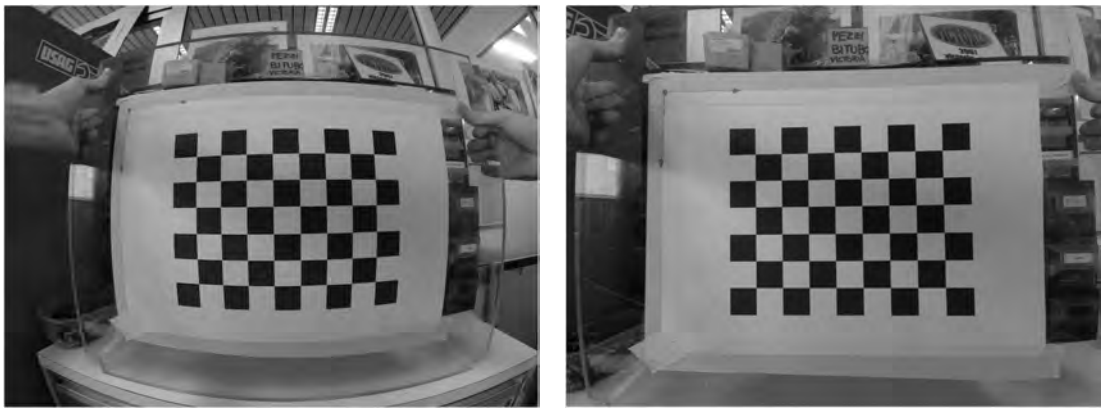


Figura 1.20: Parametri estrinseci: sistema di riferimento globale.

ma presenta ancora delle distorsioni in prossimità dei bordi. La funzione che permette di eliminare la distorsione utilizza i coefficienti $(k_1, k_2, k_3, p_1, p_2)$ ottenuti dalla calibrazione che non sono esatti ma presentano una certa incertezza. Inoltre, io credo personalmente che, a causa dell'elevata distorsione radiale che presenta la lente della GoPro, tre soli coefficienti per la determinazione della distorsione radiale in questo caso non siano sufficienti.

1.3.2 Central Camera Model

Una funzione molto utile, che verrà usata molto spesso in questo lavoro, è la Central Camera che è una *sub-class* della classe Camera che si trova all'interno del *Machine Vision Toolbox* for *MATLAB*® di *Peter Corke*, scaricabile dal sito `http://www.petercorke.com/`. Tramite questa funzione è possibile creare un modello di una *central perspective camera*, ovvero un modello che riproduce una fotocamera a prospettiva centrale, che ha quindi il punto focale a $z = 0$ e il piano dell'immagine a $z = f$. L'immagine non è quindi invertita (si veda a proposito 1.2.1). Il sistema di riferimento del modello è rappresentato nello schema seguente ed ha origine nell'angolo in alto a sinistra dell'immagine:



(a) Immagine originale distorta **(b)** Immagine elaborata non distorta

Figura 1.21: Esempio di correzione della distorsione

```
0------------> u X
|
| + (principal point)
|
| Z-axis is into the page.
|
v  v Y
```

Il comando per creare il modello della fotocamera è il seguente:

>> cam = CentralCamera(**options**)

e prevede le seguenti opzioni:

- 'name', N Name of camera
- 'focal', F Focal length [metres]
- 'distortion', D Distortion vector $(k_1, k_2, k_3, p_1, p_2)$
- 'distortion-bouguet', D Distortion vector $(k_1, k_2, p_1, p_2, k_3)$
- 'default' Default camera parameters: $1024 \times 1024, f = 8mm, 10\mu m$ pixels, camera at origin, optical axis is z-axis, u- and v-axes parallel to x- and y-axes respectively.
- 'image', IM Display an image rather than points
- 'resolution', N Image plane resolution:$N \times N$ or $N = [W; H]$
- 'sensor', S Image sensor size in metres (2×1)
- 'centre', P Principal point (2×1)
- 'pixel', S Pixel size: $S \times S$ or $S = [W; H]$

- 'noise', SIGMA Standard deviation of additive Gaussian noise added to returned image projections
- 'pose', T Pose of the camera as a homogeneous transformation
- 'color', C Color of image plane background (default [110.8])

Con i dati ottenuti dalla calibrazione delle GoPro sono stati creati dei modelli Central Camera.
Il modello per la GoPro Hero 1 è il seguente:

```
cam =
name: GOPRO HERO1 [central-perspective]
  focal length:   0.00251
  distortion:     k=(-0.2796, 0.06391, 0), p=(0.00045, 0.00054)
  pixel size:     (4.4e-06, 4.4e-06)
  principal pt:   (648.2, 495.6)
  number pixels:  1280 x 960
  noise: 0.05,0.05 pix
  T:
      1  0  0  0
      0  1  0  0
      0  0  1  0
      0  0  0  1
```

Mentre per la GoPro Hero 3+ il modello è:

```
cam =
name: GOPRO HERO3+ 1440p [central-perspective]
  focal length:   0.00261
  distortion:     k=(-0.2532, 0.06659, 0), p=(0.00011, -0.00014)
  pixel size:     (3.1e-06, 3.1e-06)
  principal pt:   (969.6, 722.9)
  number pixels:  1920 x 1440
  T:
      1  0  0  0
      0  1  0  0
      0  0  1  0
      0  0  0  1
```

1.3.3 Pose Estimation

La determinazione della posa ${}^{C}\xi_{T}$ del sistema di riferimento $\{T\}$ di un target rispetto al sistema di riferimento della fotocamera C è possibile se:

- sono note le dimensioni del target e le coordinate $(X_i, Y_i, Z_i), i \in [1, N]$ di un numero di N punti del target rispetto a $\{T\}$;
- sono noti i parametri intrinseci della fotocamera;

- sono note le coordinate (u_i, v_i) dei N punti del target proiettate sul piano dell'immagine della fotocamera.

Il problema di stimare la posizione usando le coordinate dei punti (u_i, v_i), $(X_i, Y_i, Z_i), i \in [1, N]$, e i parametri intrinseci della fotocamera è detto *Perspective-n-Point Problem, o PnP Problem.* Negli ultimi anni sono stati proposti vari algoritmi per la soluzione di questo problema. In questo lavoro verrà utilizzato l'algoritmo proposto da *Vincent Lepetit, Francesc Moreno-Noguer e Pascal Fua* denominato *EPnP: An Accurate O(n) Solution to the PnP Problem.* Si tratta di un algoritmo non iterativo la cui complessità computazionale cresce linearmente con il numero dei punti n, $O(n)$ a differenza di altri algoritmi nei quali la complessità computazionale cresce con $O(n^5)$ oppure $O(n^8)$. Questo algoritmo non iterativo presenta una migliore precisione e una minore complessità computazionale rispetto agli altri algoritmi non iterativi e rispetto agli algoritmi iterativi è molto più veloce con una piccola perdita di precisione. L'algoritmo è applicabile per $n \geq 4$ e gestisce sia punti complanari che non complanari.

Per mostrare il funzionamento di questo algoritmo si consideri una fotocamera calibrata vista nel paragrafo 1.3.2, ad esempio la GoPro Hero 1 . Siano date, inoltre le coordinate di quattro punti del target:

$$\mathbf{P} = \begin{bmatrix} X_{P_1} & X_{P_2} & X_{P_3} & X_{P_4} \\ Y_{P_1} & Y_{P_2} & Y_{P_3} & Y_{P_4} \\ Z_{P_1} & Z_{P_2} & Z_{P_3} & Z_{P_4} \end{bmatrix} \tag{1.24}$$

e le coordinate di questi punti proiettati sul piano dell'immagine:

$$\mathbf{p} = \begin{bmatrix} ux_1 & ux2 & ux_3 & ux_4 \\ vy_1 & vy_2 & vy_3 & vy_4 \end{bmatrix} \tag{1.25}$$

La posizione del target rispetto alla fotocamera è:

$>>$ `T_est` = `cam.estpose`$(\mathbf{P}, \mathbf{p})$

`T_est` è la matrice 4×4 che rappresenta la trasformazione ${}^{C}\xi_{T}$ cercata.

É importante che, le coordinate del punto $\{p_j\}$ che si trovano nella colonna j-esima della matrice $[\mathbf{p}]$, corrispondano alla proiezione del punto $\{P_j\}$ le cui coordinate dovranno essere rappresentate anch'esse dalla colonna j-esima della matrice $[\mathbf{P}]$. In altre parole, l'ordine dei punti, nelle due matrici che vengono date *"in pasto"* all'algoritmo, *deve* essere lo stesso.

1.4 Segmentazione delle immagini

La segmentazione delle immagini è il processo di partizionamento di un'immagine in regioni significative. L'obiettivo è di segmentare o separare i pixel che rappresentano oggetti di interesse da tutti gli altri pixel dell'immagine. Concettualmente si tratta di problema semplice, però dal punto di vista computazionale è molto complesso da implementare. Uno dei problemi principali riguarda la robustezza del metodo, infatti è sufficiente ad esempio cambiare l'illuminazione della scena

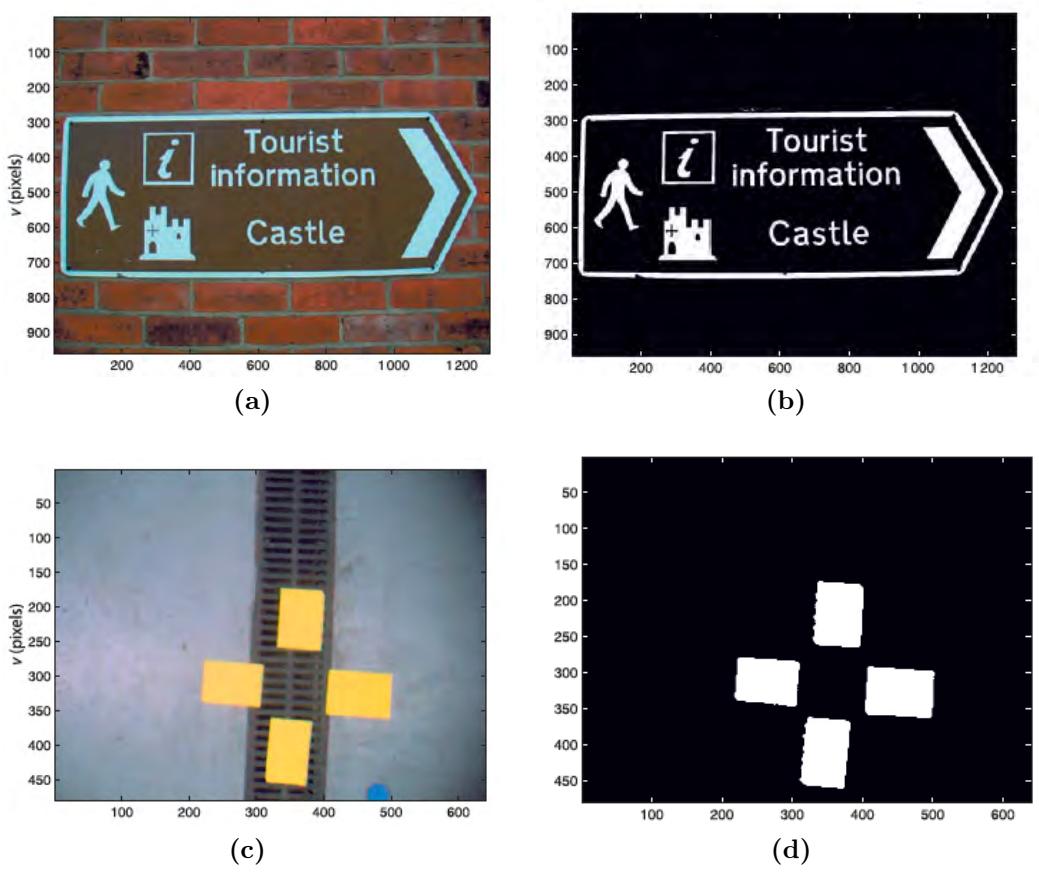


Figura 1.22: Esempio di binarizzazione delle immagini. I pixel appartenenti all'oggetto ($c = 1$) vengono rappresentati in bianco, mentre i pixel non appartenenti all'oggetto ($c = 0$) in nero. La figura (b) rappresenta il risultato della binarizzazione dell'immagine (a) effettuata dopo aver classificato i pixel aplicando una sogliatura. La figura (d) è il risultato della binarizzazione dell'immagine (c) dopo aver effettuato una classificazione dei pixel per colore o intensità cromatica.
Fonte: Peter Corke, *"Robotic, vision and control"*.

per ottenere dei risultati diversi. Il processo di segmentazione si suddivide in tre sottoproblemi. Il primo è il processo di *classificazione* dei pixel, nel quale ogni pixel viene asssociato ad una determinata classe C, con il valore $c \in \{0, 1, \ldots, C-1\}$. In questo lavoro verrano utilizzate due classi $C = 2$, si opererà quindi una classificazione binaria o binarizzazione (figura 1.22). I pixel appartenenti all'oggetto di interesse saranno classificati come ($c = 1$), mentre i restanti pixel non appartenenti all'oggetto saranno classificati con ($c = 0$). Si puo intuire come la classificazione è soggettiva per ogni immagine, cioè non esiste una suddivisione univoca per tutti i tipi di immagine ma è necessario adattarla alla scena che si intende analizzare e agli oggetti che si vogliono riconoscere. Nel caso di classificazioni con un numero di classi $C > 2$ i pixel vengono classificati per colore o livello cromatico, e in questo modo è possibile suddividere l'immagine contenente oggetti di colori diversi.

La seconda fase del processo di segmentazione è detta *rappresentazione* e consiste nell'individuazione degli insiemi spaziali $(S_1, S_2, \ldots, S_m)$ dei pixel appartenenti alle

varie classi, ovvero l'identificazione delle regioni che delimitano un gruppo di pixel della stessa classe. Ad ogni gruppo verrà assegnata un etichetta.

La terza ed ultima fase è detta di *descrizione*, i gruppi di pixel S_i verranno descritti da funzioni scalari o vettoriali che rappresentano ad esempio la dimensione, la forma e la posizione.

1.4.1 Classificazione

Nel processo di classificazione ad ogni pixel dell'immagine viene associato il valore di una determinata classe C. Operando una classificazione binaria ($C = 2$) otterremo una suddivisione fra pixel che appartengono all'oggetto e pixel che non appartengono all'oggetto.

Una regola comune è la seguente:

$$c[u,v] = \begin{cases} 0, & \text{se } I[u,v] < t; \\ 1, & \text{se } I[u,v] \geq t. \end{cases} \quad \forall (u,v) \in I. \tag{1.26}$$

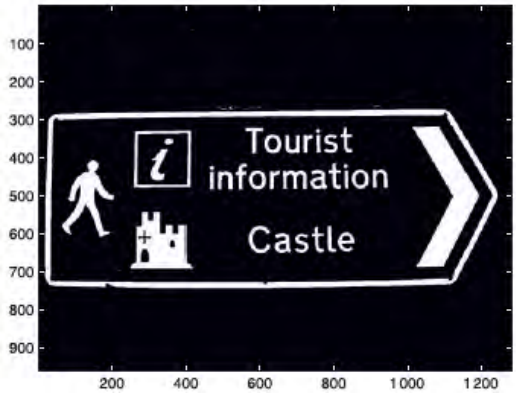
Questo approccio è detto *sogliatura* e t è detto *valore di soglia*.

L'istogramma di figura 1.23c presenta distribuzione bimodale con due picchi ben definiti che rappresentano due *famiglie* di pixel. Il picco più basso attorno allo 0.9 corrisponde ai pixel più luminosi e presenta una bassa variazione attorno al valore medio, mentre il picco attorno allo 0.3, più alto e largo del primo, rappresenta i pixel scuri ed ha una più elevata variazione della luminosità attorno al valore medio. Impostare un valore di soglia t, in modo da separare la due classi di pixel per questo caso, risulta molto semplice. La netta demarcazione fra i due picchi ci permette di poter scegliere un valore di t compreso fra 0.6 e 0.8.
Il *Metodo Otsu* permette di calcolare automaticamente il valore di soglia ottimale che separa un'immagine in due classi di pixel in modo da minimizzare la varianza dei valori all'interno di ciascuna classe e massimizza la varianza dei valori tra le classi, supponendo però che l'istogramma presenti due picchi distinti. Per l'immagine di figura 1.23, il *Metodo Otsu* calcola un valore di soglia $t = 0.6$.

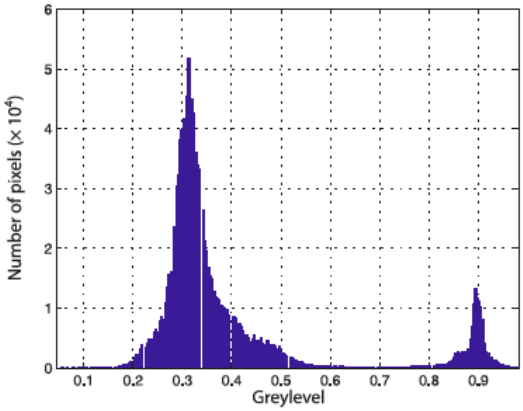
Chiaramente non tutte le immagini presentano una netta separazione come nel caso precedente. Si consideri ad esempio la stessa immagine nella quale, però, è stato modificato il valore di luminosità tramite l'aggiunta di un gradiente in direzione diagonale. Il risultato è mostrato in figura 1.24a e l'istogramma che si ottiene è radicalmente diverso: non presenta più i due picchi distinti e di conseguenza non è più possibile separare le due famiglie di pixel come nel caso precendente (figura 1.24b. In questo caso il *Metodo Otsu* calcola una soglia $t = 0.5412$ e il risultato dell'applicazione di tale soglia è mostrato in figura 1.24c. Nel caso quindi, di distribuzione non bimodale dei valori di luminosità, il *Metodo Otsu* fallisce.

Le tecniche basate sulla *sogliatura* sono notoriamente fragili: un leggero cambiamento nell'illuminazione della scena e il valore di soglia scelto diventa inappropriato. Nella maggior parte delle scene reali non si può ad esempio scegliere una soglia che seleziona un determinato oggetto piuttosto che un altro all'interno di un'immagine. Distinguere un oggetto dallo sfondo rimane quindi un problema di *computer vision* complesso.

(a) Immagine originale



(b) Immagine binarizzata



(c) Istogramma dei valori di pixel in scala di grigi.

Figura 1.23: Binarizzazione di un immagine.
Fonte: Peter Corke, *"Robotic, vision and control"*.

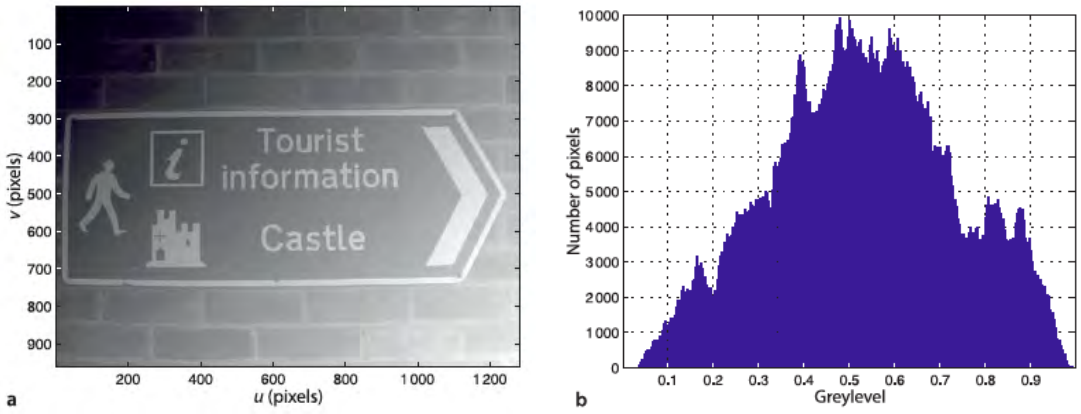
1.4.2 Rappresentazione

La *rappresentazione* consiste nel connettere pixel della stessa classe adiacenti per formare insiemi spaziali $S_1, S_2, \ldots, S_m$. Ad esempio, nell'immagine di figura 1.25, si possono identificare facilmente tre regioni bianche che vengono dette anche *blobs*. Una *blob* è un insieme di pixel della stessa classe, bianco in questo caso, che sono collegati tra loro. In altre parole, si potrebbe dire che una *blob* è una regione di pixel contigui appartenenti alla stessa classe. Il comando:

```
>> [label, m, parents, cls] = ilabel(im);
```

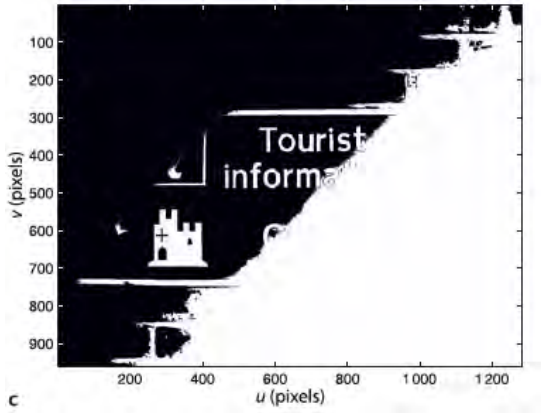
permette di individuare le *blobs* all'interno di un'immagine. La variabile `m` indica il numero di *blobs* individuate (nel caso della figura 1.25 si hanno tre regioni bianche e due nere quindi `m = 5`), mentre `label` è l'etichetta che viene assegnata ad ogni pixel della regione individuata. Quindi ad esempio, tutti i pixel appartenente alla regione 2 verranno individuati dall'etichetta 2.
Il vettore `parents` indica se una regione è contenuta in un altra e quale regione la contiene; `cls` indica la classe dei pixel di una regione (1 bianco, 0 nero).

Nell'analisi `ilabel` viene assunto come valore di default una connettività a 4-vie,



(a) Immagine in scala di grigi con gradiente di luminosità diagonale

(b) Istogramma dei valori di pixel in scala di grigi.



(c) Applicazione del valore di soglia dato dal Metodo Otsu. ($t = 0.54$)

Figura 1.24: Binarizzazione di un immagine.
Fonte: Peter Corke, *"Robotic, vision and control"*.

cioè si considerano i pixel connessi all'interno di una regione solo se sono a contatto ai lati. La connettività a 8-vie considera connessi all'interno di una stessa classe i pixel che sono a contatto anche agli angoli.

1.4.3 Descrizione

In questa fase le regioni precedentemente individuate vengono analizzate e descritte da funzioni scalari che restituiscono ad esempio le dimensioni, la forma, la posizione del centro, il perimetro, il momento d'inerzia, l'elissoide d'inerzia, ecc.... Queste funzioni vengono dette *RegionFeature*.

In questo lavoro verrà utilizzata una funzione contenuta in *Machine Vision Toolbox* for *MATLAB®* di *Peter Corke*, denominata `iblobs`, che permete di calcolare le *RegionFeature* di un immagine precendentemente binarizzata.
La funzione `iblobs` prevede una serie di opzioni che permettono di filtrare le regioni analizzate in modo da restiture le *RegionFeatures* delle regioni che effettivamente interessano.

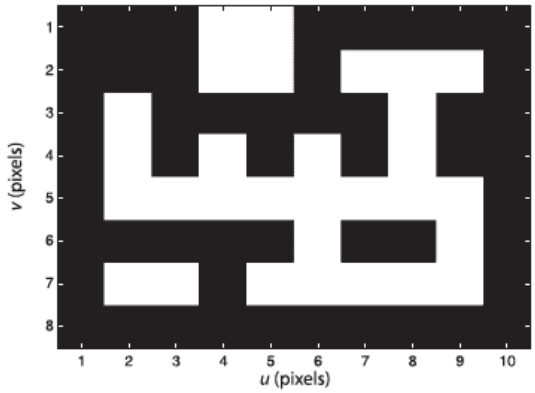


Figura 1.25: Individuazione delle regioni: immagine esempio.
Fonte: Peter Corke, *"Robotic, vision and control"*

Si riportano a seguire la sintassi e le opzioni per la funzione `iblobs`:

```
>> f = iblobs(im, options)
```

Options:

- 'aspect', A set pixel aspect ratio, default 1.0;
- 'connect', C set connectivity, 4 (default) or 8;
- 'greyscale' compute greyscale moments 0 (default) or 1;
- 'boundary' compute boundary (default off);
- 'area', $[A1, A2]$ accept only blobs with area in the interval A1 to A2;
- 'shape', $[S1, S2]$ accept only blobs with shape in the interval S1 to S2;
- 'touch', T accept only blobs that touch (1) or do not touch (0) the edge (default accept all);
- 'class', C accept only blobs of pixel value C (default all).

Se ad esempio effettuiamo la seguente *blob analysis*:

```
>> f = iblobs(im, 'class', 1, 'area', [1000 2000], 'shape',
[0.5 1], 'touch', 0, 'boundary')
```

per l'immagine di figura 1.22d, si otterrà il seguente risultato:

```
f =
   (1) area=1211, cent=(170.9,100.9), theta=1.55, b/a=0.689, class=1,
label=1, touch=0, parent=23, perim=190.3, circ=0.420;
   (2) area=1171, cent=(120.8,148.0), theta=-2.90, b/a=0.687, class=1,
label=8, touch=0, parent=23, perim=155.4, circ=0.610;
   (3) area=1377, cent=(212.9,155.6), theta=-2.98, b/a=0.715, class=1,
label=10, touch=0, parent=23, perim=158.3, circ=0.691;
   (4) area=1268, cent=(161.6,198.1), theta=1.67, b/a=0.630, class=1,
```

```
label=20, touch=0, parent=23, perim=174.7, circ=0.522.
```

dove fra le opzioni si è inserito:

- il valore per classe dei pixel pari a 1, cioè sono state elaborate le regioni formate solo da pixel bianchi,
- il valore dell'area massima e minima,
- il valore del rapporto massimo e minimo b/a fra i raggi dell'elissoide d'inerzia,
- l'opzione non tocca il bordo, in questo modo sono state elaborate solo le regioni che non toccano il bordo dell'immagine.

Come si può notare le regioni individuate appartengono ai quattro riquadri gialli dell'immagine originale 1.22c. Queste opzioni inserite sono molto utili, in quanto permettono di indviduare le regioni che realmente interessano fra tutte le regioni possibili. Se si avesse inserito alcuna opzione nel comando, la *blob analysis* effettuata sulla stessa immagine individuerebbe ben 117 regioni diverse.

Capitolo 2

Misura della posizione del pilota

In questo capitolo sarà descritto il metodo ideato per la determinazione della posizione del pilota. Questo metodo è stato pensato ed ideato completamente all'interno di questo lavoro di tesi. L'idea di base deriva dall'algoritmo di *pose estimation* (paragrafo 1.3.3) che per determinare la posizione di un oggetto nello spazio tramite una fotocamera calibrata necessita delle coordinate di almeno quattro punti di questo oggetto proiettati sul piano dell'immagine. Si è pensato quindi di creare un bersaglio o *target* che verrà posizionato sulla schiena del pilota e di riprendere i movimenti di quest'ultimo con una telecamera posizionata sulla parte posteriore della moto (figura 2.1). Successivamente, tramite le tecniche di segmentazione dell'immagine viste in 1.4, si ricaveranno le coordinate di quattro punti del target che verranno inserite nell'algoritmo di pose estimation. Quest'ultimo restituirà la posizione del target rispetto al sistema di riferimento della telecamera.

L'esatta posizione e orientamento del sistema di riferimento della telecamera però non si conosce con esattezza ed inoltre è molto difficile da determinare in quanto dipende dalla posizione del sensore all'interno della camera. Il piano dell'immagine infatti coincide con il sensore CMOS e quindi su quest'ultimo è collocato l'origine del sistema di riferimento della camera, di conseguenza la sua posizione e orientamento dipende da come il sensore è stato installato all'interno della telecamera.

Questo problema è stato risolto in modo molto semplice: l'idea è stata di posizionare un secondo target fisso sulla motocicletta e calcolare la posizione relativa *target pilota - target fisso*. In questo modo ci si svincola dal sistema di riferimento della telecamera e il nuovo sistema di riferimento è rappresentato dal target fisso di cui è possibile conoscere con esattezza la sua posizione ed orientamento.

La telecamera scelta è una GoPro in quanto offre un campo di visuale molto ampio che permette di riprendere bene il pilota anche da molto vicino.

Nel primo paragrafo è descritto il target realizzato, mentre nel secondo è illustrato il codice MatLab che effettua l'elaborazione del video ed ottiene la posizione del pilota.

2.1 Target

Nella scelta del tipo di target da utilizzare si è dovuto tenere conto di una serie di aspetti tali da rendere possibile una analisi robusta e soprattutto non eccessivamente

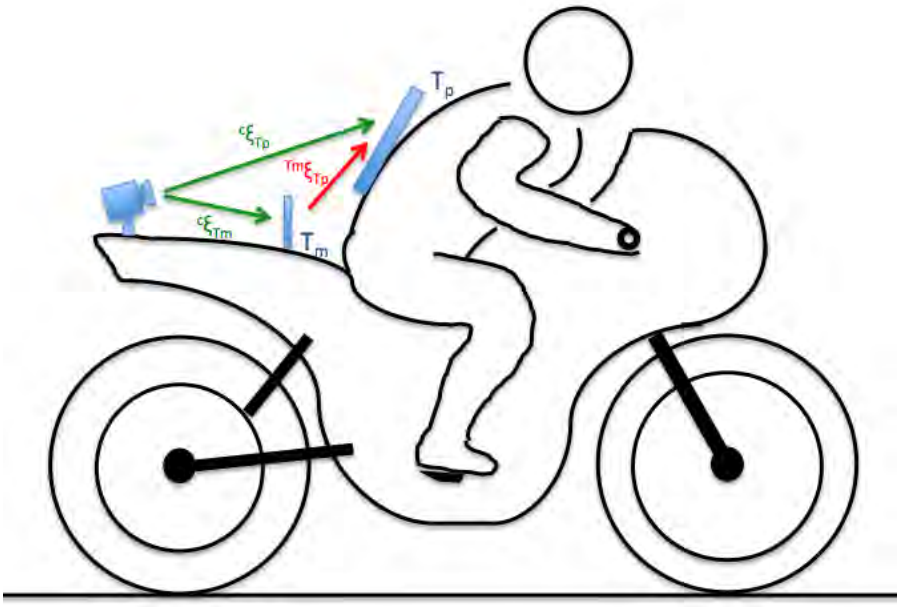


Figura 2.1: Schema del posizionamento dei target e della telecamera sulla motocicletta.

pesante dal punto di vista computazionale. In questa analisi, che vedremo nel paragrafo 2.2, si andrà ad effettuare sul singolo frame una *blob analysis* in modo da ottenere le coordinate delle proiezioni dei punti del target sul sensore della camera, e successivamente, tramite l'algoritmo di *pose-estimation* si andrà a determinare la posizione del target stesso. É chiaro quindi che il target deve avere determinate caratteristiche e soddisfare perlomeno queste due condizioni fondamentali:

- deve essere ben riconoscibile;
- i punti individuati dovranno essere associati in modo univoco e corretto ai punti del target.

Riuscire ad individuare i punti in modo univoco è il problema più ostico. Infatti, la *blob analysis* determina le regioni apparentemente senza un ordine preciso (che può essere ad esempio dalla più grande alla più piccola o viceversa). In realtà, analizzando l'algoritmo di *blob analysis* utilizzato, si è visto che le regioni sono determinate *"scansionando"* la matrice di pixel, di cui è formata l'immagine, da sinistra verso destra e dall'alto verso il basso: perciò la regione più in alto a sinistra sarà identificata come *blob(1)* e la regione più in basso a destra sarà la *blob(n)*. Questo criterio però, nel nostro caso, non è sufficiente a determinare le regioni univocamente, perchè se immaginiamo di ruotare ad esempio di $\pi/2$ il target attorno all'asse normale al suo piano, l'ordine con cui vengono determinate le regioni nell'immagine ruotata e in quella non ruotata *non* è lo stesso.
Le soluzioni pensate per risolvere questo problema sono state diverse: inizialmente si è pensato di utilizzare delle figure geometriche diverse fra loro e di individuare la regione i-esima tramite il rapporto *area/perimetro*. Infatti una stella ★, ad esempio, avrà un rapporto *area/perimetro* molto minore rispetto a un cerchio ◯. In questo caso però è necessario il calcolo del perimetro delle regioni da parte della

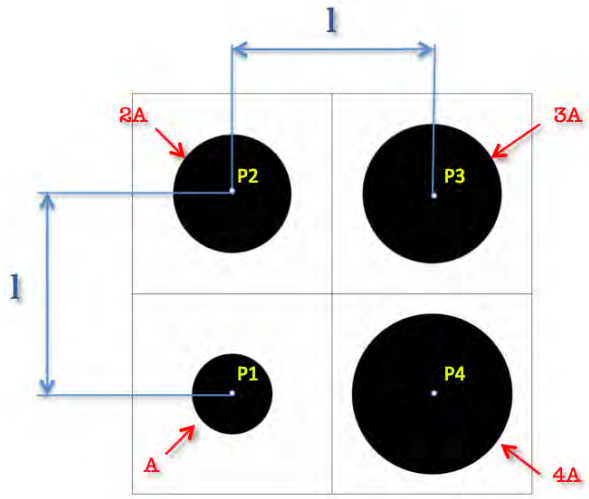


Figura 2.2: Modello rappresentativo del target ideato.

funzione *blob analysis* che comporta un aumento del tempo di analisi. Per questo motivo è stata adoperata una soluzione più semplice: ovvero sono state scelte quattro figure circolari con differenti aree e, l'*idea* è di ordinare per area crescente le regioni determinate dalla *blob analysis* in modo da eliminare l'ambiguità nel riconoscimento relle regioni.

ll target ideato è piano, cioè le figure circolari si trovano sullo stesso piano. Sarebbe stato possibile utilizzare anche un target tridimensionale (esempio: una piramide), soluzione che però non è stata presa in considerazione dato che i risultati ottenuti con il target piano sono stati molto soddisfacenti e inoltre perchè si tratterebbe di una soluzione molto più complessa. Si sono scelte inoltre quattro figure circolari perchè 4 è il numero minimo di punti necessario all'algoritmo *pose estimation* per determinare la posizione dell'oggetto. Utilizzare un numero maggiore di figure per avere un numero maggiore di punti non migliora l'accuratezza della misura a meno che i punti siano *non* complanari.

Le aree delle figure circolari sono in rapporto fra loro come rappresentato in figura 2.2. I centri P_1, P_2, P_3, P_4 rappresentano i vertici di un quadrato di lato $l = 10\,cm$ per il *target-pilot* e $l = 5\,cm$ per il *target-moto* e rappresentano i punti $\{P_j\}$ che verranno determinati con la blob analysis.

Affinche il target sia ben riconoscibile è necessario garantire un elevato contrasto fra le figure circolari e lo sfondo. Per questo il *target-pilota.* è stato realizzato in legno e rivestito con una stoffa nera, le figure circolari invece sono bianche, realizzate in carte e incollate sulla base rivestita. Rivestire con della stoffa nera evita di avere dei riflessi sul target a causa della luce del sole, che potrebbero dare dei problemi alla *blob analysis* (figura 2.3).

I target viene assicurato al pilota tramite quattro cinghie: due cinghie elastiche che vengono legate attorno al torace e due cinghie che passano attorno alle spalle. In questo modo il target è ben fissato al pilota e non vi sono rischi che si sposti durante la guida (figura 2.4).

Il *target-moto* non è necessario che sia rivestito in stoffa, quindi è stato realizzato tramite una semplice stampa su foglio di carta delle figure circolari, che sono in

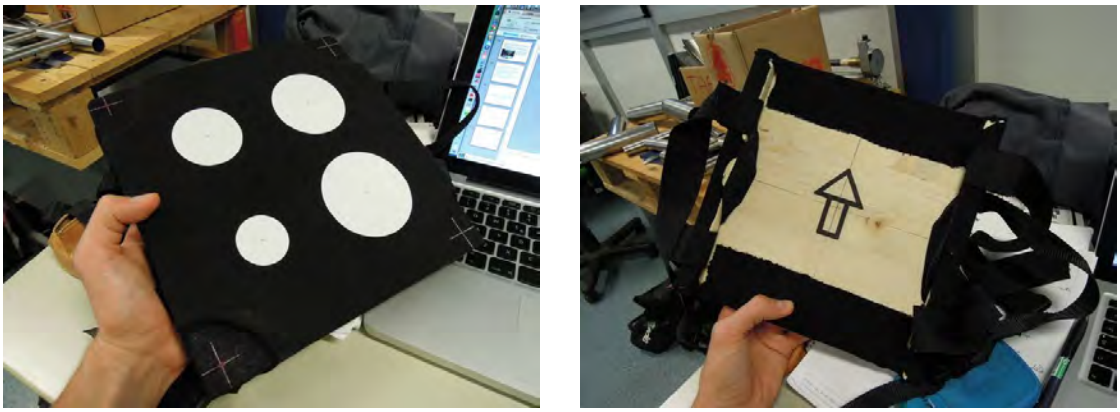


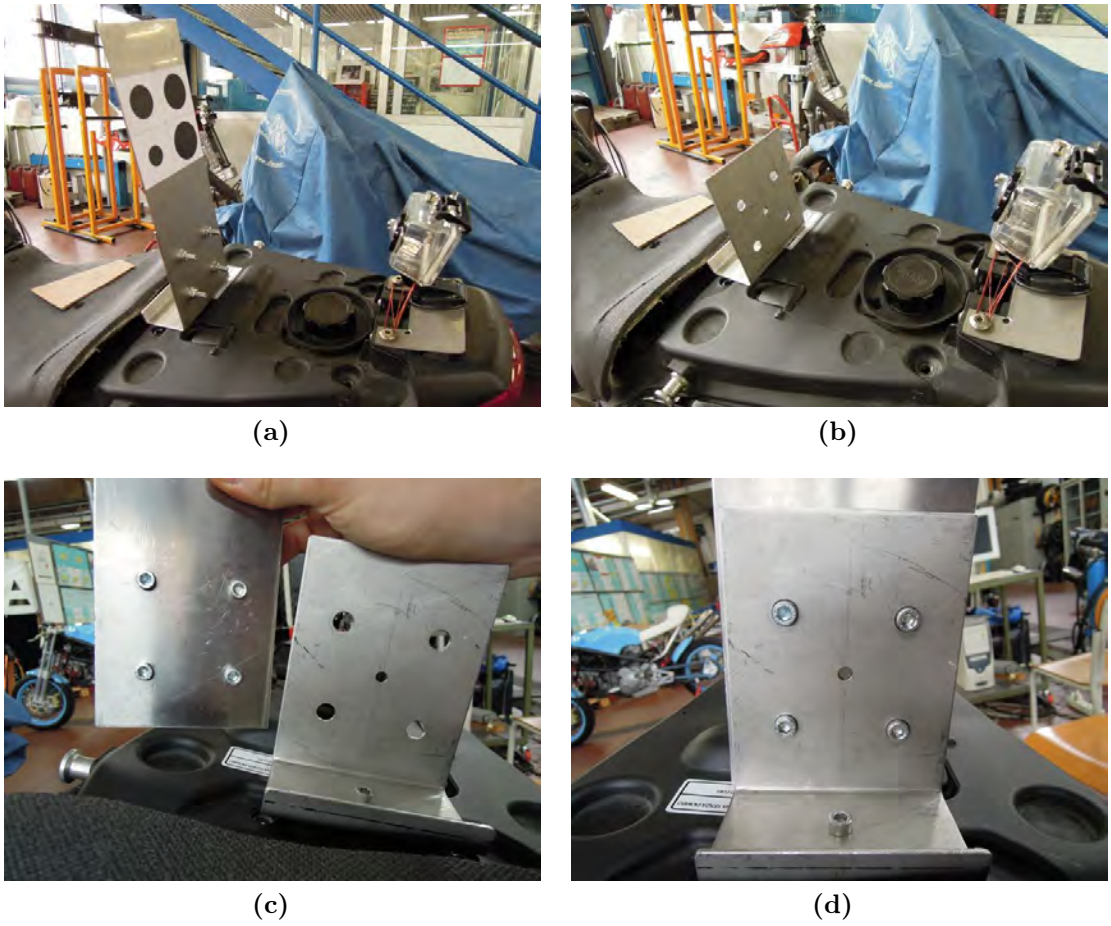
Figura 2.3: *Target-pilot*

(a) **(b)**

Figura 2.4: *Target-pilot*: fissaggio del target al pilota. Si notano le cinghe elastiche attorno al torace e le due chinghe attorno alle spalle.

questo caso nere su sfondo bianco. Il target è rappresentato in figura 2.5 e come si può notare è dotato di un particolare supporto *"ad incastro"* e permette di rimuovere la parte superiore rapidamente e senza dover svitare o avvitare delle viti durante i test. Questa soluzione è stata adottata per poter posizionare il *target-moto* il più possibile al centro dell'inquadratura in modo da ridurre l'errore dovuto alla distorsione dell'immagine e anche per il fatto che, se fosse posizionato nella parte bassa dell'inquadratura (soluzione che era stata adottata inizialmente), quando si andrà a correggere la distorsione della lente parte della cornice più esterna dell'immagine verrà tagliata (i pixel vengono stirati verso l'esterno) con il rischio di perdere una parte del target stesso.

Rimuovere il bersaglio durante i test potrebbe sembrare una cosa insolita dato che il *target-moto* è necessario per determinare la posizione relativa *pilota-motocicletta*, però dato che questo, a differenza del *target-pilot*, è un target statico, e sufficiente al limite un solo frame per determinare la sua posizione. Per questo, durante i test il *target-moto* verrà posizionato mentre il pilota è in sella alla motocicletta, saranno registrati alcuni secondi di filmato e poi sarà rimosso per poter effettuare le guide e rendere ben visibile il pilota, che altrimenti sarebbe coperto (figura 2.6).



(a) **(b)**

(c) **(d)**

Figura 2.5: *Target-moto*: particolare del posizionamento sulla motocicletta. *Foto(a)*: target in posizione; *Foto(b)*: target rimosso; *Foto(c)*: particolare del supporto rimovibile; *Foto(d)*: altro particolare del supporto rimovibile

2.2 Codice

Nelle righe che seguono verrà descritto il funzionamento del codice per la stima della posizione del pilota. Il codice è scritto in linguaggio MatLab e si suddivide nelle seguenti parti:

- video processing;
- pose estimation *target-pilota*;
- pose estimation *target-moto*;
- posizione relativa *target-pilota - target-moto*;

2.2.1 Video Processing

Lo script video_processing esegue le seguenti operazioni:

1. caricamento video in memoria;

Figura 2.6: Target moto: *camera view*

2. selezione dello tratto di interesse da analizzare;

3. individuazione dei quatto punti del *target-pilot* (ovvero blob-analysis sul target del pilota) e correzione della distorsione della lente ;

4. individuazione dei quatto punti del *target-moto* (ovvero blob-analysis sul target fisso) e correzione della distorsione della lente;

5. salvataggio dei dati.

Il caricamento del video in memoria si esegue con il comando:

```
>> video = VideoReader(videoName)
```

dove `videoName` è il nome del file video da caricare compreso di estensione ad esempio `videoTest.avi` o `videoTest.mp4`. Una volta caricato il video si procede inserendo il punto in cui si intende far partire l'analisi, il punto di fine (in secondi) e la frequenza a cui si intende analizzare il filmato, ovvero il numero di frame per secondo che si intende analizzare. Ad esempio se si vuole analizzare un spezzone che va dal minuto 1′30” al minuto 3′45” inseriremo:

```
>> Insert starting time [s]:  150
>> Insert ending time [s]:  225
>> Insert frequency [frames/s]:  30
⟶ Start frame:  4500
⟶ End frame:  6749
⟶ Number of frame that will be analyse:  2249
```

A questo punto si passa all'analisi di ogni singolo frame che verrà estratto dal video. Ad ogni frame corrisponderà un immagine di dimensioni (in pixel) pari alla risoluzione del video. Nel caso della **GoPro Hero 1** la risoluzione utilizzata è pari a $1280 \times 960\,px$ @ $30\,fps$. La *blob-analysis* non verrà però effettuata sull'intero frame perchè necessita di un tempo di calcolo troppo elevato (circa 10 secondi per frame) e inoltre si avrebbero delle difficolta nel riconoscimento delle regioni appartenenti al target a causa dell'eccessivo numero di regioni che verrebbero individuate. L'esigenza di realizzare un *codice* che non sia eccessivamente pesante dal punto di vista computazionale e che sia in grado di individuare le regioni

di interesse senza richiedere, volta per volta, l'intervento manuale dell'utente ha portato a pensare ad una soluzione diversa.

Anzichè analizzare l'intero frame, si è pensato di analizzare solamente una finestra ridotta di dimensioni tali da contenere esclusivamente il target. Questa finestra corrisponde ad un *ritaglio* del frame originale ed avrà quindi dimensioni molto ridotte rispetto all'immagine di origine. Si avrà quindi una drastica riduzione dei tempi di elaborazione dell'immagine da parte della funzione *blob-analysis* proprio perchè il numero di pixel da analizzare, ora è nettamente inferiore, ed inoltre sarà molto più semplice riconoscere esclusivamente le regioni appartenenti al target. Con questa soluzione, il tempo di elaborazione scende sotto i $0.5\,[s]$ per frame contro i $10[s]$ del frame completo. L'implementazione di questa tecnica ha però un problema di fondo: il target si muove e potrebbe uscire dalla finestra ridotta. La soluzione ideata per risolvere questo problema è stata di realizzare una *finestra mobile* che *"insegue"* il target. Nelle righe che seguono verrà illustrata l'implementazione della finestra mobile e il *modus operandi* del codice.

Il primo passo consiste nella definizione della finestra. Il codice plotterà a display l'immagine del primo fotogramma da analizzare e "chiederà" all'utente di selezionare la finestra entro la quale è contenuto il target. La selezione avviene attraverso due *click*: il primo in corrispondenza del vertice in alto a sinistra (figura 2.7a) e il secondo in corrispondenza del vertice in basso a destra (figura 2.7b). Tramite questi due *click* diamo al calcolatore le coordinate in pixel della finestra ridotta.

```
>> ⟶ IDENTIFY THE RIDER TARGET ...
>> Select upper-left and lower-right corner ...
>> DONE!
>> Running blob-analysis ...
```

Queste coordinate sono espresse nel sistema di riferimento del frame, che ha origine nel vertice in alto a sinistra. Ad esempio, facendo riferimento sempre alla figura 2.7, si ottengono le seguenti coordinate:

```
>> pts1(x,y) = [497; 260]
>> pts2(x,y) = [775; 445]
```

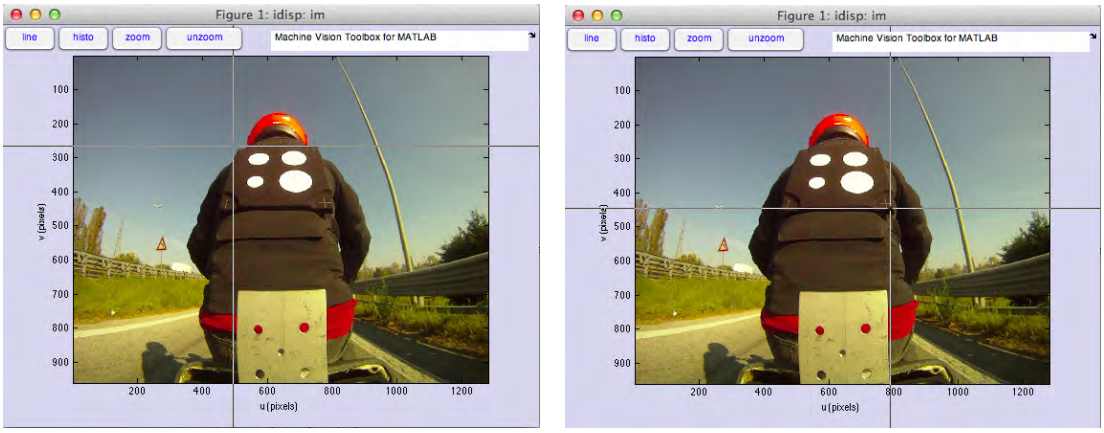
e quindi le dimensioni della finestra ridotta sono:

$$x_{fin} = \frac{775 - 497}{2} = 139\,px; \quad y_{fin} = \frac{445 - 260}{2} = 93\,px$$

A questo punto l'immagine corrispondente alla finestra selezionata viene prima binarizzata, tramite una sogliatura che sfrutta il metodo di Otsu (si veda 1.4.1), e successivamente verrà eseguita la *blob-analysis* che determinerà le coordinate in pixel dei quattro centri delle figure circolari del target.
Queste coordinate, però, sono determinate rispetto all'origine del sistema di riferimento della finestra ridotta, perciò è necessario trasformarle rispetto al sistema di riferimento del frame. Quindi se $({}^{f}u_i, {}^{f}v_i)$ sono le coordinate del punto i-esimo in coordinate della finestra ridotta, le nuove coordinate, rispetto al sistema di riferimento del frame, si ottengono tramite le seguenti equazioni:

$$u_i = {}^{f}u_i + u_{pts1}; \quad v_i = {}^{f}v_i + v_{pts1};$$

dove (u_{pts1}, v_{pts1}) non sono altro che le coordinate, nel sistema di riferimento del frame, dell'angolo in alto a sinistra della finestra ridotta.



(a) Selezione del primo angolo **(b)** Selezione del secondo angolo

Figura 2.7: Identificazione del target: selezione della finestra mobile

A questo punto, quindi, è stata determinata la matrice $\{\mathbf{p}_1\}$ contenente le coordinate dei quattro punti del target per il primo frame. Ora, come già accennato poc'anzi, se la finestra ridotta rimanesse fissa durante l'analisi dei frame successivi, succederebbe inevitabilmente che, ad un certo punto, il target esca dalla finestra; per questo motivo è necessario realizzare una finestra mobile che insegua il target. La regola implementata nel codice per realizzare la finestra ad inseguimento è la seguente:

Le coordinate del punto centrale della finestra al frame j-esimo sono date dalle coordinate del centro del target calcolate al frame (j-1)-esimo.

Quindi la finestra è centrata sul target e si sposterà con un ritardo di un frame rispetto ad esso: è per questo che è stata denominata *finestra ad inseguimento*. Un ritardo di un frame corrisponde a un ritardo di un trentesimo di secondo in un video a 30 fps.

Per quanto riguarda la *blob-analysis*, i parametri dati per filtrare le regioni devono essere *adattativi* perchè, quando il target si muove, la prospettiva farà variare le aree ed i rapporti fra gli assi dell'elissoide d'inerzia delle *blobs*. La *blob-analysis* è implementata nella funzione denominata `blobs_00`. Le opzioni inserite per individuare esclusivamente le figure circolari del target sono:

touch $= \mathbf{0}$;

class $= \mathbf{1}$;

area $= [\texttt{A1}, \texttt{A2}]$;

shape $= [\texttt{S1}, \texttt{S2}]$.

Quindi impostato per il primo fotogramma un range per le aree $[A_1, A_2]$ e un range per il rapporto b/a dei semiassi dell'elissoide d'inerzia $[S_1, S_2]$, per il fotogramma successivo i nuovi parametri saranno:

$$\begin{aligned} \texttt{area} &= [\,\texttt{area\_min} + \texttt{c}, \quad \texttt{area\_max} + \texttt{c}\,] \\ \texttt{shape} &= [\,\texttt{shape\_min} + \texttt{d}, \quad \texttt{shape\_max} + \texttt{d}\,] \end{aligned}$$

dove:

- `area_min` e `area_max` sono l'area minima e massima trovate al fotogramma precedente;
- `shape_min`, `shape_max` sono i rapporti b/a minimo e massimo trovati al fotogramma precedente;
- `c` e `d` sono delle costanti.

In questo modo il codice cercherà di individuare delle regioni simili alle regioni trovate al passo precedente, adattando così l'analisi volta per volta in ogni frame. Può comunque capitare che, in alcuni frame, il codice non individui solamente le quattro aree del target ma, ad esempio, solo tre di queste oppure addirittura delle regioni non appartenenti al target.
Per questo motivo è stata inserita una funzione di controllo, che *se* le regioni individuate non sono quattro, l'analisi viene fermata e:

1. viene plottato a schermo l'immagine della *finestra ridotta* di analisi con evidenziate le regioni trovate;
2. vengono plottati i risultati della blob analysis;
3. viene chiesto all'utente se desidera riselezionare la finestra ridotta oppure se intende modificare i parametri dell'analisi.

Vediamo ora degli esempi in cui l'analisi è stata arrestata perchè non sono state individuate le quattro regioni del target.
Il codice plotterà un messaggio del tipo:

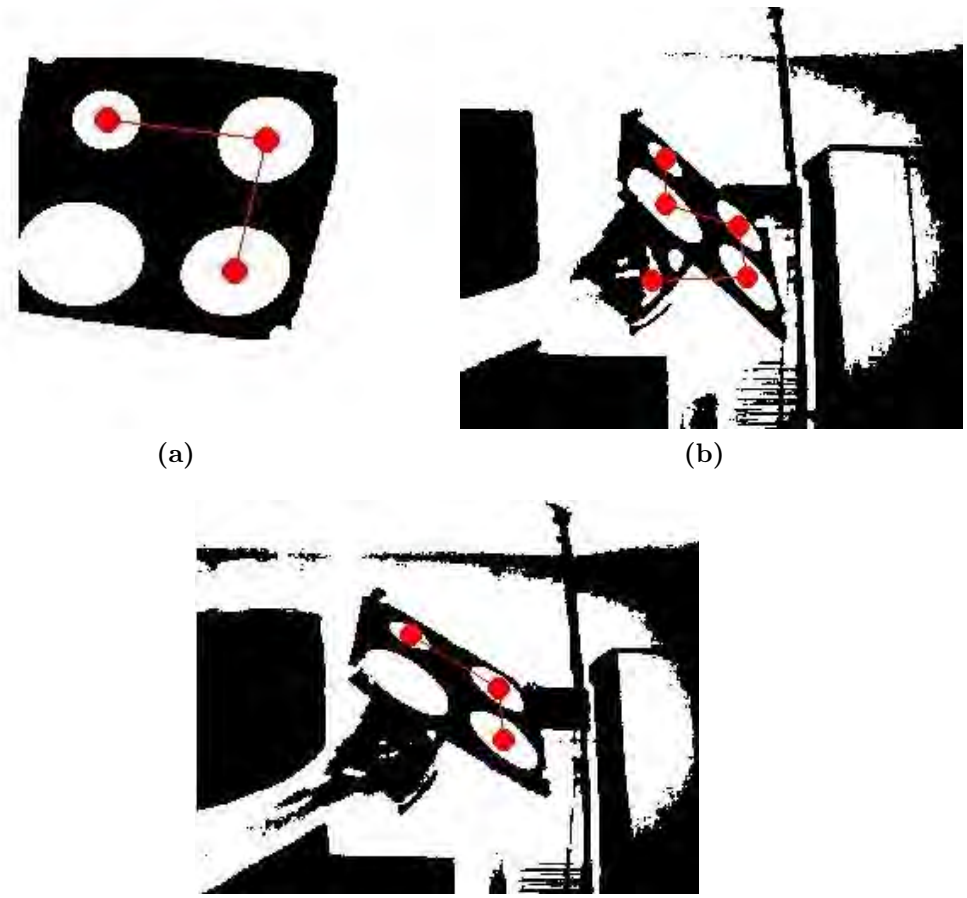
```
>> Warning:  iblobs failed at frame 320.  Identified blobs...

   Area      Shape      Theta
  0.8220     0.8841    -3.1259
  1.6890     0.8697     2.8314
  1.9600     0.8021     2.9512

>> retry to individuate target on current frame?  ( 1 = YES; [ ] =
other opt)
```

e a display verrà plottata l'immagine in cui c'è stato l'errore, in modo che l'utente possa capire qual'è stato il problema (figura 2.8).

Nel caso di figura 2.8a si nota che sono state individuate solo tre regioni (indicate dai marcatori rossi) perchè la quarta tocca il bordo e quindi viene automaticamente scartata. In questo caso è sufficiente selezionare una finestra più grande digitando `1` : il codice plotterà a schermo il frame completo e l'utente andrà a ridefinire la finestra ridotta di analisi (come fatto per il frame iniziale).



(a) **(b)**

(c)

Figura 2.8: Esempi di blob analysis error.

Nella figura 2.8b invece le regioni identificate sono 5. In questo caso è necessario modificare manualmente i parametri di `area` e `shape` scegliendoli in base ai risultati ottenuti dalla blob analysis cointenuti nel messaggio di errore:

```
>> Warning:  iblobs failed at frame 320.  Identified blobs...

  Area      Shape      Theta
 0.1890     0.3204    -2.3746
 0.9190     0.3316    -2.3786
 0.3620     0.3127     0.8274
 0.6270     0.3294     0.8487
 0.1030     0.2916     1.7068

>> retry to individuate target on current frame?  ( 1 = YES; [ ] =
other opt) 2

shape = [ 0.16943 0.4925 ]
>> try new shape limits?  [S1 S2] = [0.3 0.5]

area = [ 100 1952 ]
>> try new area limits?  [A1 A2] = [150 1000]
```

La regione che non appartiene al target ha un valore di `shape = 0.2916`. É sufficiente, quindi, un valore di $S1 > 0.3$ per eliminarla.

Per quanto riguarda invece il caso di figura 2.8c, non è possibile identificare tutte e quattro le regioni a causa di un riflesso sul bordo di una di esse. In questo caso modificare i parametri o riselezionare la finestra non avrà alcun effetto, quindi al comparire dei messaggi di errore si darà sempre `Invio`, cioè non si inserirà nessun parametro, il codice automaticamente prenderà il risultato del frame precedente e passerà al successivo; quindi quel particolare frame in cui l'analisi non è possibile viene scartato e sostituito con il risultato del frame precedente.

Lo stato di avanzamento dell'analisi è monitorato da una barra di avanzamento che riporta il numero di frame analizzati rispetto al totale da analizzare (figura 2.9).

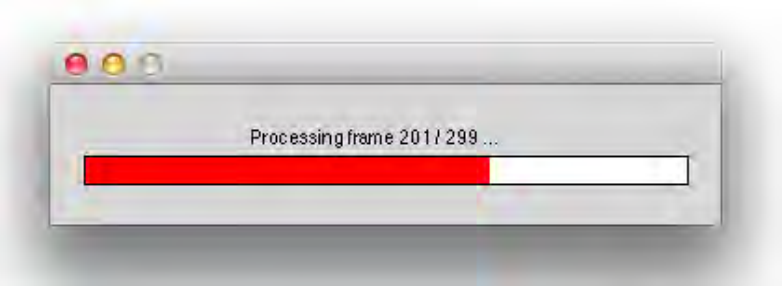


Figura 2.9: Barra di avanzamento analisi.

Terminata l'analisi i risultati sono contenuti nella matrice `centers` avente dimensione $(2, 4, \texttt{n\_frame})$, dove l'ultima dimensione rappresenta il frame analizzato. A questo punto il codice chiederà se si intende correggere la distorsione della lente:

```
>> Do you want to correct lens distortion?  ( [ ]=Yes, other=No)
```

In caso di risposta affermativa verrà richiamata la funzione `undistortPixel` che applicherà la correzione esclusivamente alle coordinate dei pixel contenute nella matrice `centers`. In altre parole, non verrà corretta la distorsione sul frame completo, ma saranno corretti solamente i quattro pixel trovati dalla *blob-analysis* effettuata in precedenza. Questo procedimento è concettualmente errato perchè la *blob-analysis* dovrebbe essere effettuata già sul frame non distorto, cioè bisognerebbe correggere prima la distorsione dell'intero frame e poi effettuare l'analisi. Correggere la distorsione di ogni frame è un processo molto oneroso e se applicato ad ogni frame del video l'analisi richiederebbe diverse ore di elaborazione. Per questo motivo ci si accontenta di correggere la distorsione esclusivamente dei quattro pixel del target, con la consapevolezza che questo comporta l'introduzione di un certo errore, che verrà valutato nel capitolo 3 relativo alla calibrazione, ma che comporta un vantaggio enorme in termini di tempo di elaborazione, basti pensare che correggere l'intero frame significa correggere $1280 \times 960 = 1.23 * 10^6 pixel$.

Terminata questa prima parte relativa al *target-pilot*, il codice passa all'analisi del *target-moto*.

```
>> Do you want to analyze even the target moto (fixed target)?  (
[ ]=Yes, other=No)
```

Tabella 2.1: Tempi di analisi (1800 frame).

Function name	Calls	Total Time (s)	Self Time (s)
video_processing	1	150,8	6,8
blobs_00	1800	75,4	38,6
VideoReader.read	1801	59,4	58,6
iblobs	1800	22,6	4,0
imoments	13069	14,6	9,4
tb_otparse	14873	7,0	2,9
click_and_crop	2	7,0	0,0
ginput	2	6,7	0,1
close	1802	6,6	0,1
isprop	22110	4,1	4,1
waitbar	1801	4,0	3,0
undistortPixel	1	2,4	0,1
normalize_pixel_02	7200	2,3	0,2
comp_distortion_oulu	7200	2,1	2,1
im2bw	1800	1,8	0,4
rgb2gray	1800	1,2	0,1
graythresh	1800	1,2	0,2

```
>> The Target Moto it is on the same video of Rider Target?  (
[ ]=Yes, other=No) NO
>>Insert video name with suffix (.avi, .mp4):  targetMoto.mp4
⟶ IDENTIFY THE MOTO TARGET ...
>> Select upper-left and lower-right corner ...
>> DONE!
>> Running blob-analysis ...
```

L'analisi è molto simile all'analisi vista per il *target-pilot* con alcune differenze:

- è possibile caricare il nuovo video nel caso che il *target-moto* dovesse trovarsi in un video diverso da quello caricato all'inizio;
- il numero di frame analizzato è fissato a priori ed è pari a 50;

La selezione della finestra ridotta e la correzione della distorsione avvengono, invece, con le stesse modalità dell'analisi precedente.

Terminata l'analisi dei target il codice salverà le matrici contenenti i centri nella cartella di lavoro corrente in due file con i seguenti nomi:

`centers_pilot.mat`;

`centers_moto.mat`.

La tabella 2.1 riporta i tempi di esecuzione delle principali funzioni richiamate durante un analisi del *target-pilot* di uno spezzone di video della durata di 60 *secondi*

a 30 fps (1800 frame). Come si può vedere, l'analisi è durata in totale $150,8\,[s]$, quindi circa $2,5\,min/1\,min\,video$. Dai tempi di esecuzione di ogni singola funzione si nota, in particolare che la funzione `undistortPixel` impiega solamente $2,4\,s$ per correggere la distorsione dei pixel relativi ai punti del target, mentre se avessimo corretto la distorsione dei frame completi sarebbero state necessarie più di 2 ore.

2.2.2 Pose estimation target

Ricavate le coordinate dei punti dei target proiettati sul sensore piano dell'immagine, si passa ora alla stima della posizione dei due target rispetto alla camera tramite gli script `pose_est_pilot` e `pose_est_moto`. Questi due script sono molto simili tra loro, l'unica differenza è:

`pose_est_pilot` stima la posizione camera-pilota e quindi restituisce n matrici $T_{pilot}(4,4,i)$; dove $i = 1, 2, \ldots, n.$;

`pose_est_moto` stima la posizione camera-moto e restituisce *una* matrice $T_{moto}(4,4)$ che è la matrice ottenuta mediando i 50 frame analizzati nel passo precedente.

La posizione del *target-moto* è rappresentata da una sola matrice proprio perchè è un target statico. Quindi al limite basterebbe un solo fotogramma per ottenere la sua posizione, però, per tener conto della dispersione degli errori, sono state prese 50 posizioni (vedi 2.2.1) e poi ne è stata fatta la media.
Al termine dell'esecuzione di questi due script otterremo quindi due file che contengono le posizioni dei due target rispetto alla camera:

`T_pilot.mat`;

`T_moto.mat`.

salvati anch'essi nella cartella corrente di lavoro.

2.2.3 Posizione relativa *target moto - target pilota*

Per calcolare la posizione relativa *target moto - target pilota* si faccia riferimento alla figura [INSERIRE FIGURA]. Si può scrivere:

$$^{C}\xi_{T_p} = {}^{C}\xi_{T_m} \oplus {}^{T_m}\xi_{T_p}$$

sottraendo $,^{C}\xi_{T_m}$ da entrambi i membri si ha:

$$\ominus\, {}^{C}\xi_{T_m} \oplus {}^{C}\xi_{T_p} = \ominus\, {}^{C}\xi_{T_m} \oplus {}^{C}\xi_{T_m} \oplus {}^{T_m}\xi_{T_p}$$

e quindi si ottiene la forma finale che rappresenta la posizione relativa *target moto - target pilota*:

$$^{T_m}\xi_{T_p} = \ominus\, {}^{C}\xi_{T_m} \oplus {}^{C}\xi_{T_m}$$

che in forma matriciale si scrive:

$$^{T_m}[\mathbf{T}]_{T_p} = {}^{C}[\mathbf{T}]^{-1}_{T_m} * {}^{C}[\mathbf{T}]_{T_p}$$

dove ${}^{C}[\mathbf{T}]_{T_m}$ è la posizione *camera-target moto*, mentre ${}^{C}[\mathbf{T}]_{T_p}$ è la posizione *camera-target-pilot*.

L'operazione appena vista viene eseguita dallo script `rel_pos_MOTOpilot`, che salva nella cartella corrente il file `T_motopilot.mat` contente le posizioni relative *target moto- target pilota*.

Capitolo 3

Calibrazione

Lo stumento di *pose estimation* messo a punto necessità di una calibrazione per poter capire quale sia il suo potenziale in termini di qualità della misura.
Effettuare una calibrazione di un sistema di questo tipo, in grado di determinare tutti i gradi di libertà di un oggetto nello spazio, non è semplice in quanto è necessario conoscere l'esatta posizione e orientamento del target per ogni posizione rilevata, per poi confrontarla con la misura data dal sistema. Il problema principale è rappresentato dal fatto che per avere un discreto numero di dati è necessario affettuare diverse acquisizioni e, per ognuna di esse rilevare, la posizione e l'orientamento del target nello spazio; il che diventa un lavoro piuttosto oneroso perchè richiede molto tempo per misurare accuratamente ogni posizione. Per questo motivo abbiamo pensato di sfruttare il sistema di *motion capture* del *laboratorio di bio-meccanica* del *Dipartimento di Ingegneria Industriale* dell Università degli studi di Padova, che è in grado di rilevare il movimento nello spazio di particolari marcatori. Questi marcatori verranno applicati al target e il sistema di *motion capture* ne restituirà le posizioni nello spazio.

3.1 BTS

I sistemi di *motion capture* (figura 3.1) sono dei sistemi impiegati principalmente in *sport engineering* per catturare e studiare il movimento durante l'esecuzione di gesti atletici. Il sistema usato nei nostri test è un sistema *optoelettronico* ed è costituito da sei telecamere ad infrarossi a $60\,Hz$ (figura 3.2) e da dei particolari marcatori detti *markers* (figura 3.3) che riflettono la luce nello spettro dell'infrarosso e quindi visibili da queste telecamere. Questo sistema è in grado di rilevare la posizione (x, y, z) dei marker nello spazio e, affinchè il sistema possa triangolare la sua posizione, il marker deve essere visibile da almeno due telecamere. Prima di eseguire i test, è necessario calibrare il volume in cui si intende operare. Durante la procedura di calibrazione, le telecamere identificano una terna di riferimento comune, che viene posizionata in un punto del volume da calibrare, e che diventerà l'origine del sistema di riferimento del volume calibrato. Inoltre vengono valutati i parametri intrinseci ed estrinseci di ogni camera. Solo tramite questa procedura è possibile effettuare la ricostruzione ed ottenere le posizioni dei *markers* all'interno del volume calibrato.

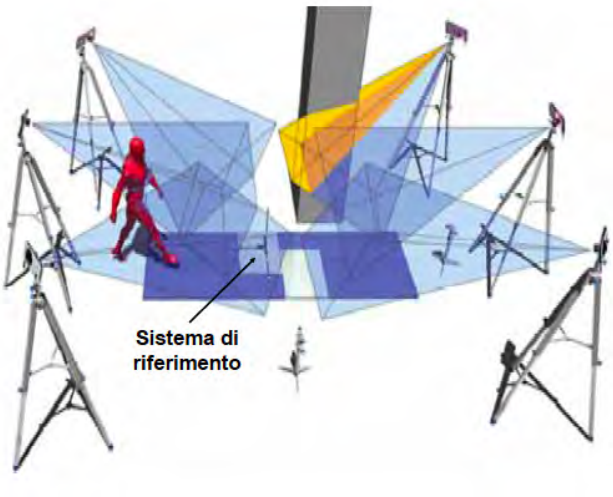


Figura 3.1: Optoelectronic motion capture system. Fonte: *Dispense del corso* "Sport engineering and rehabilitation devices" 2013, *prof.* N. Petrone.

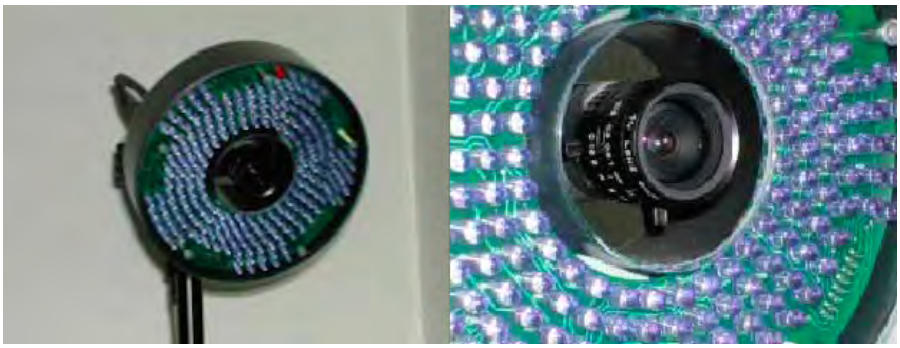


Figura 3.2: Videocamera ad infrarossi

3.2 Test

Per effettuare i test sono stati posizionati tre *marker* sul target fisso (figura 3.4) e quattro *marker* sul target mobile (figura 3.5). Sul target fisso i marker sono stati posizionati in modo che i segmenti che congiungono i loro centri formino un angolo retto in modo che, in fase di elaborazione dei dati, si possa costruire una terna orientata come il target.

I test sono stati effettuati compiendo una serie di movimenti all'interno del volume di interesse e registrando con entrambe le telecamere simultaneamente. Le telecamere in questione sono:

Figura 3.3: Markers

- GoPro Hero 1 con risoluzione $1280 \times 960p$ @ 30 *fps*;
- GoPro Hero 3+ con risoluzione $1920 \times 1440p$ @ 48 *fps*.

3.3 Elaborazione dati

Optoelectronic System

Il sistema di *motion capture* restituisce le coordinate (x, y, z) di ogni marker, presente all'interno del volume calibrato, rispetto al suo sistema di riferimento globale. Per poter conoscere la posizione e l'orientazione del target mobile rispetto al target fisso si procede nel seguente modo:

- si crea una nuova terna centrata ed orientata sul target fisso e solidale ad esso;
- si crea una nuova terna centrata ed orientata sul target mobile e solidale ad esso;
- si calcola la posizione relativa fra le due terne.

Nel caso del target fisso, tramite le coordinate dei punti dei tre marker posizionati su di esso, si ottengono i due vettori:

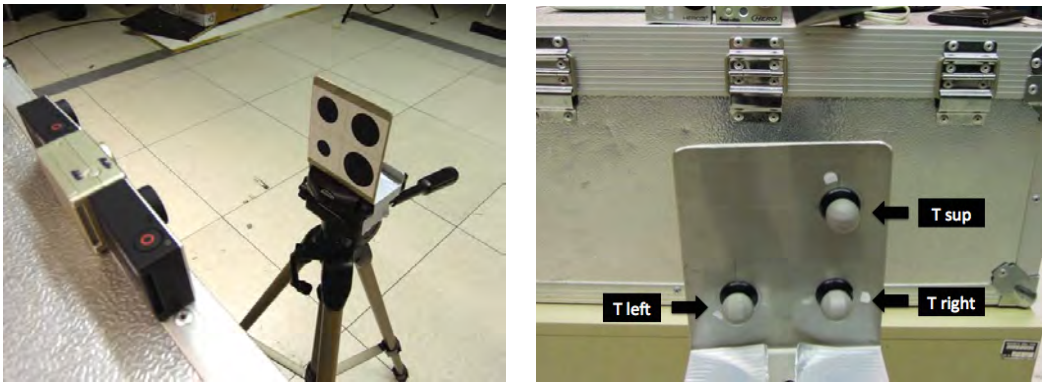
$$\vec{\mathbf{v}}_z = (T_{left} - T_{right})^T; \quad \vec{\mathbf{v}}_y = (T_{sup} - T_{right})^T;$$

Tramite la seguente:

$$\vec{\mathbf{v}}_x = \vec{\mathbf{v}}_y \times \vec{\mathbf{v}}_z \tag{3.1}$$

si ottiene il vettore $\vec{\mathbf{v}}_x$ ortogonale al piano formato dai due vettori $(\vec{\mathbf{v}}_z, \vec{\mathbf{v}}_y)$. Ora i tre vettori $(\vec{\mathbf{v}}_x, \vec{\mathbf{v}}_y, \vec{\mathbf{v}}_z)$ non rappresentano una terna di vettori ortogonali fra loro perchè, per quanto i marker siano stati posizionati in modo accurato, i due vettori $\vec{\mathbf{v}}_z$ e $\vec{\mathbf{v}}_y$ non saranno mai ortogonali. Per ottenere una terna di vettori ortogonali si dovrà quindi ricalcolare il vettore $\vec{\mathbf{v}}_y$ con la seguente:

$$\vec{\mathbf{v}}_y = \vec{\mathbf{v}}_z \times \vec{\mathbf{v}}_x \tag{3.2}$$



(a) **(b)**

Figura 3.4: Disposizione delle telecamere (a) e dei marker sul target fisso (b).

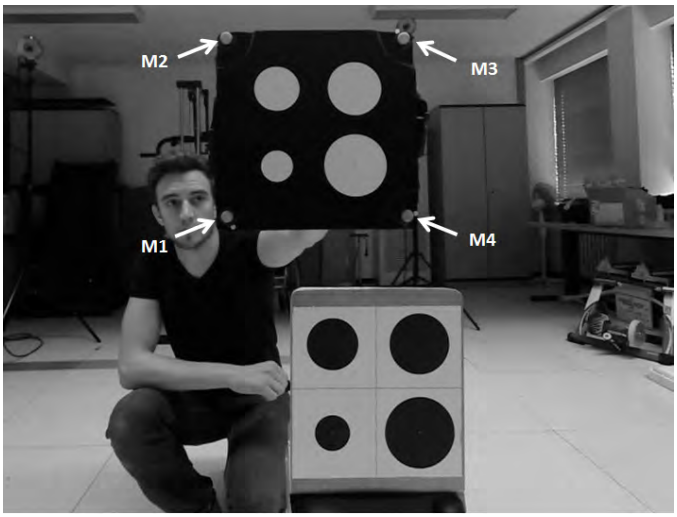


Figura 3.5: Target mobile: disposizione dei marker

dato che $\vec{\mathbf{v}}_z$ e $\vec{\mathbf{v}}_x$ sono ortogonali per definizione. La matrice di trasformazione dal sistema di riferimento globale O al nuovo sistema di riferimento centrato sul marker T_{right} è quindi:

$$ {}^{O}[\mathbf{T}]_{T_{right}} = \begin{pmatrix} \|v_x(1)\| & \|v_y(1)\| & \|v_z(1)\| & T_{right}(x) \\ \|v_x(2)\| & \|v_y(2)\| & \|v_z(2)\| & T_{right}(y) \\ \|v_x(3)\| & \|v_y(3)\| & \|v_z(3)\| & T_{right}(z) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.3} $$

Come si può notare, le colonne del minore 3×3 che rappresenta le rotazioni, sono date dalle componenti dei vettori $(\vec{v}_x, \vec{v}_y, \vec{v}_z)$ normalizzati.
La nuova terna creata, centrata in T_{right}, deve essere traslata in corrispondenza del centro del target fisso, quindi:

$$ {}^{O}[\mathbf{T}]_{tf} = {}^{O}\,[\mathbf{T}]_{T_{right}} \cdot {}^{T_{right}}[\mathbf{T}]_{tf} \tag{3.4} $$

dove ${}^{T_{right}}[\mathbf{T}]_{tf}$ è:

$$ {}^{T_{right}}[\mathbf{T}]_{tf} = \begin{pmatrix} 1 & 0 & 0 & x_c \\ 0 & 1 & 0 & y_c \\ 0 & 0 & 1 & z_c \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Per quanto riguarda il target mobile il procedimento è lo stesso, con due uniche differenze:

- la terna è centrata sul punto ottenuto calcolando la media fra i punti centrali dei due segmenti congiungenti i markers del target mobile diagonalmente opposti;
- la terna è calcolata per ogni posizione assunta dal target.

anche in questo caso si otterrà quindi una matrice ${}^{O}\mathbf{T}_{tp}$ che rappresenta la trasformazione dal sistema di riferimento globale O al sistema di riferimento centrato sul target mobile.

La posizione relativa fra il target fisso e il target mobile è quindi:

$$ {}^{T_{tf}}[\mathbf{T}]_{T_{tp}} = {}^{O}[\mathbf{T}]_{T_f}^{-1} \cdot {}^{O}[\mathbf{T}]_{T_{tp}} \tag{3.5} $$

Videocamera

Dal codice (paragrafo 2.2) si ottiene la posizione relativa target fisso - target mobile che però è espressa rispetto al sistema di riferimento della videocamera che ha un orientamento diverso dal sistema di riferimento del sistema di *motion capture* (vedi 1.3.2). Affinchè vi sia quindi compatibilità nei risultati, quando si andrà ad effettuare il confronto fra i due sistemi, è necessario che le terne siano orientate tutte allo stesso modo. In figura 3.6 è rappresentato l'orientameto delle due terne nel volume di calibrazione; la trasformazione necessaria ad orientare la terna della videocamera come la terna del sistema di *motion capture* è data dalla seguente:

$$ {}^{T_{mO}}[\mathbf{T}_{cam}]_{T_{pO}} = {}^{O}[\mathbf{T}]_{T_f} \cdot {}^{T_f}[\mathbf{T}_{cam}]_{T_p} \cdot {}^{T_m}[\mathbf{T}]_{O} \tag{3.6} $$

dove:

$$ {}^{O}[\mathbf{T}]_{T_f} = Troty(-\pi/2) \cdot Trotx(\pi) = $$
$$ = \begin{pmatrix} \cos(-\pi/2) & 0 & \sin(-\pi/2) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-\pi/2) & 0 & \cos(-\pi/2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\pi & -\sin\pi & 0 \\ 0 & \sin\pi & \cos\pi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. $$

$$ {}^{T_m}[\mathbf{T}]_{O} = Troty(\pi/2) \cdot Trotz(-\pi) = $$
$$ = \begin{pmatrix} \cos(\pi/2) & 0 & \sin(\pi/2) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\pi/2) & 0 & \cos(\pi/2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(-\pi) & -\sin(-\pi) & 0 & 0 \\ \sin(-\pi) & \cos(-\pi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. $$

É possibile, inoltre, dimostrare che:

$$ {}^{O}[\mathbf{T}]_{T_f} = {}^{T_m}[\mathbf{T}]_{O}^{-1}. \tag{3.7} $$

Calcolo dell'errore

L'errore è stato valutato come differenza fra la misura ottenuta dal sistema di *motion capture* e la misura ottenuta dalle telecamere che è stata elaborata nei due modi seguenti:

1. misura ottenuta elaborando il filmato privo di distorsione della lente;
2. misura ottenuta elaborando il filmato *con* distorsione della lente e correzione della distorsione a posteriori sulle coordinate dei centri rilevate.

In questo modo è stato possibile valutare l'effetto della correzione della distorsione effettuata solamente sulle coordinate dei centri (vedi paragrafo 2.2.1) rispetto all'analisi del filmato privo di distorsione ottenuto correggendo i frames completi.

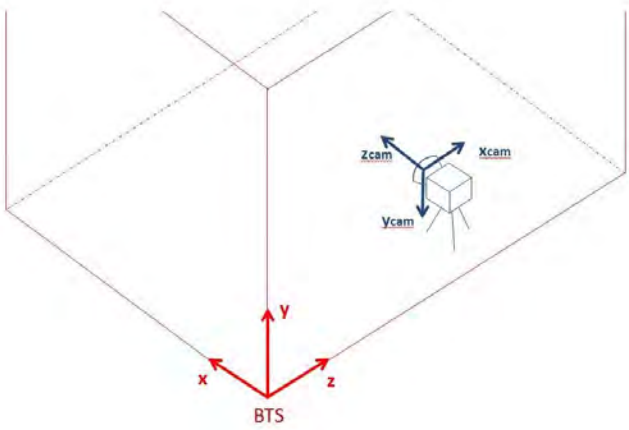


Figura 3.6: Orientamento dei sistemi di riferimento del BTS e della telecamera.

3.4 Risultati dei test

Le prove effettuate sono suddivise in tre parti:

1. traslazioni lungo x, y, z;
2. rotazioni attorno a x, y, z;
3. movimenti combinati.

Nelle pagine che seguono sono rappresentati i risultati dei test. I dati sono stati filtrati tutti a $2\,Hz$ tramite un filtro *low pass* di ordine 10. In figura 3.7 è riportato l'andamento della coordinata X rilevato dal sistema BTS e dalle due telecamere, in figura 3.8 è riportato l'andamento della coordinata X e il corrispondente errori in [mm] delle due telecamere, valutati come $x_{BTS} - x_{GoPro}$ e in figura 3.9, invece, si confrontano gli errori con la velocità di spostamento in [mm/s].
Infine, in figura 3.10, è riportato il valore assoluto dell'errore commesso dalle telecamere confrontato con il valore assoluto dell'errore commesso applicando la distorsione ai soli quattro pixel che corrispondono ai centri delle *blobs* (vedi 2.2.1).
Per quanto riguarda le traslazioni lungo gli assi Y e Z, i risultati sono riportati, allo stesso modo, dalla figura 3.11 alla 3.14 per l'asse Y e dalla figura 3.15 alla 3.18 per Z. I dati delle rotazioni attorno ai tre assi x, y, z sono rappresentati dalla figura 3.19 alla figura 3.30. Il significato dei grafici è lo stesso descritto per le traslazioni.
I risultati del test con movimenti combinati sono riportati dalla figura 3.31 alla 3.34.

3.4.1 Traslazioni

Asse X

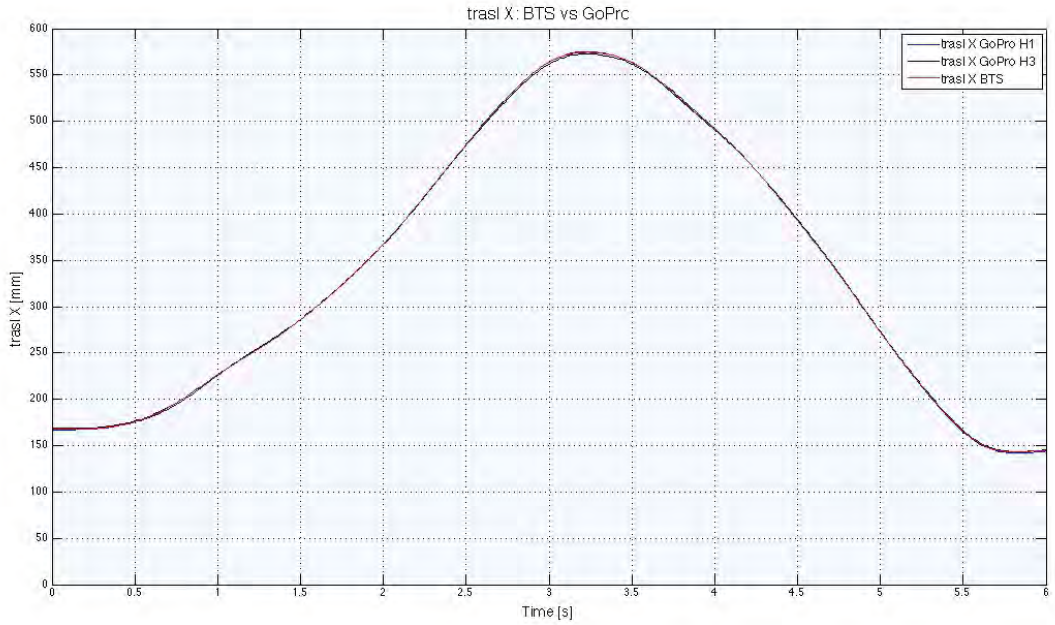


Figura 3.7: Traslazione lungo l'asse X: confronto fra i risultati ottenuti dal *BTS motion capture system* e i risultati delle due telecamere.

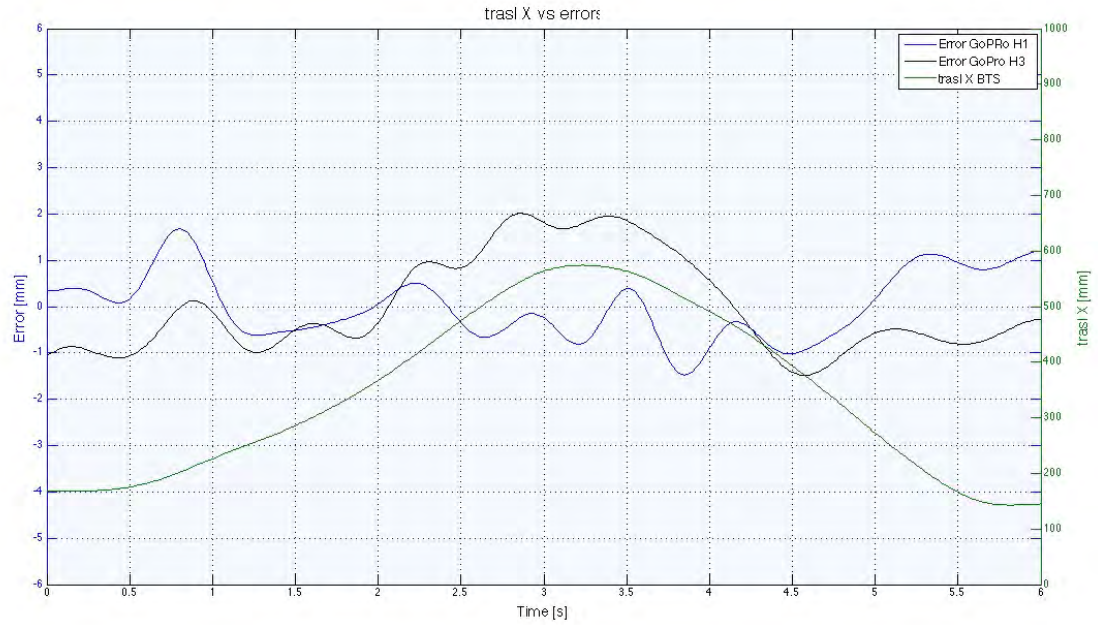


Figura 3.8: Traslazione lungo l'asse X ed errori sulla misura ottenuta dalle telecamere.

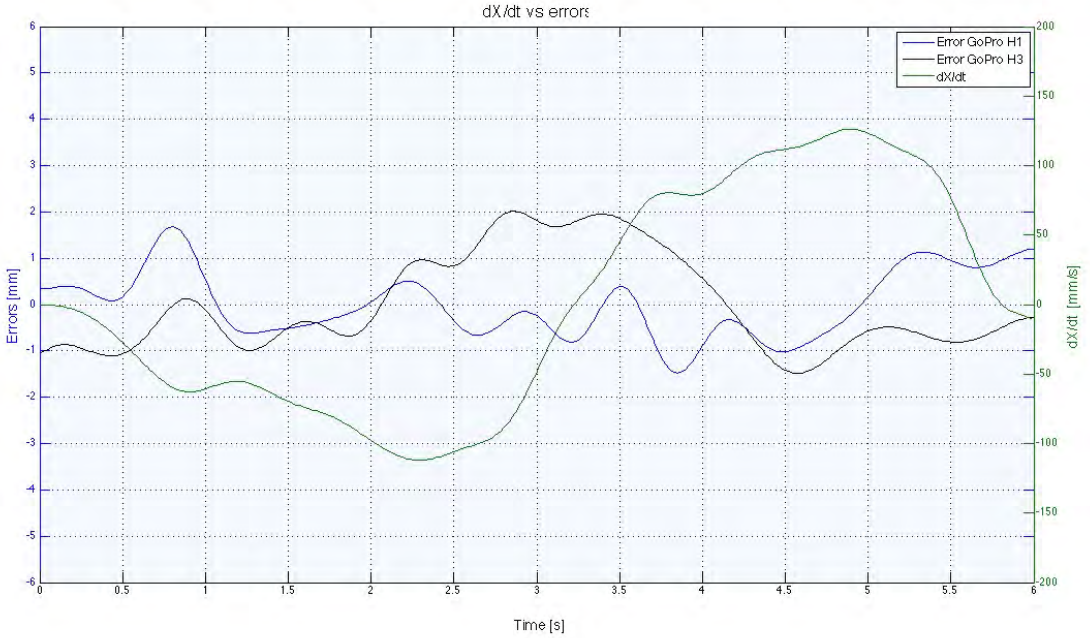


Figura 3.9: Velocità di traslazione lungo l'asse X ed errori sulla misura ottenuta dalle telecamere.

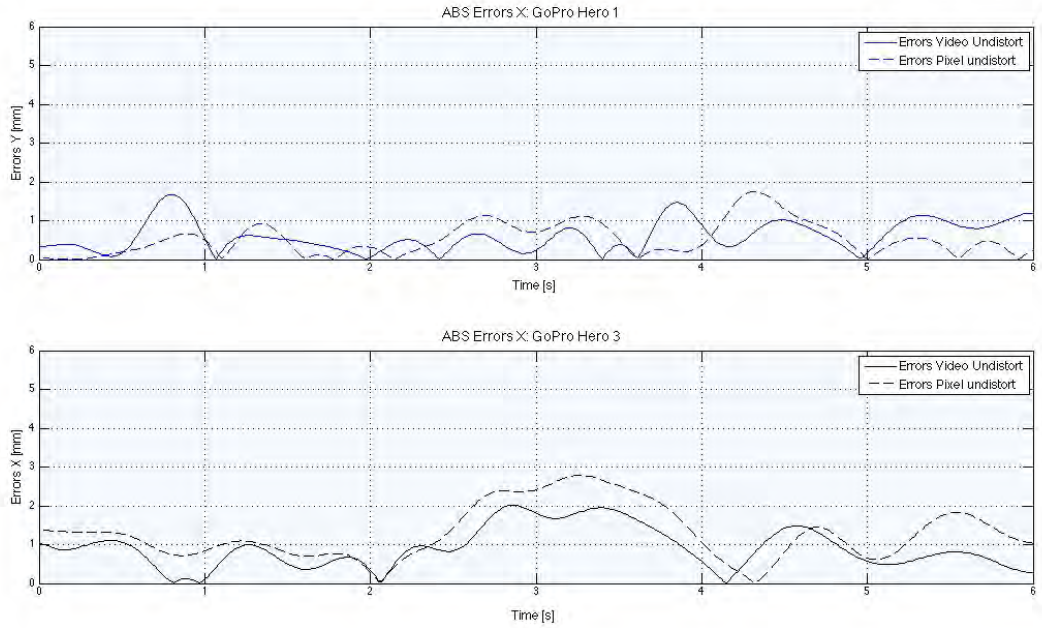


Figura 3.10: Confronto fra gli errori ottenuti con i due metodi di correzione della distorsione (valore assoluto degli errori, traslazione asse X).

Asse Y

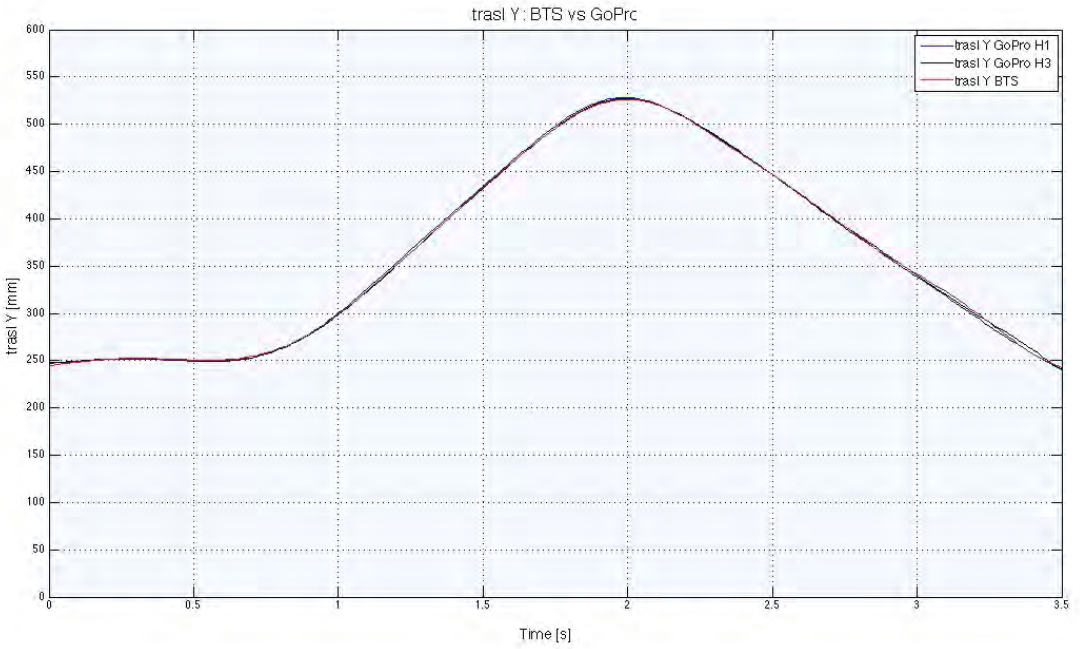


Figura 3.11: Traslazione lungo l'asse Y: confronto fra i risultati ottenuti dal *BTS motion capture system* e i risultati delle due telecamere.

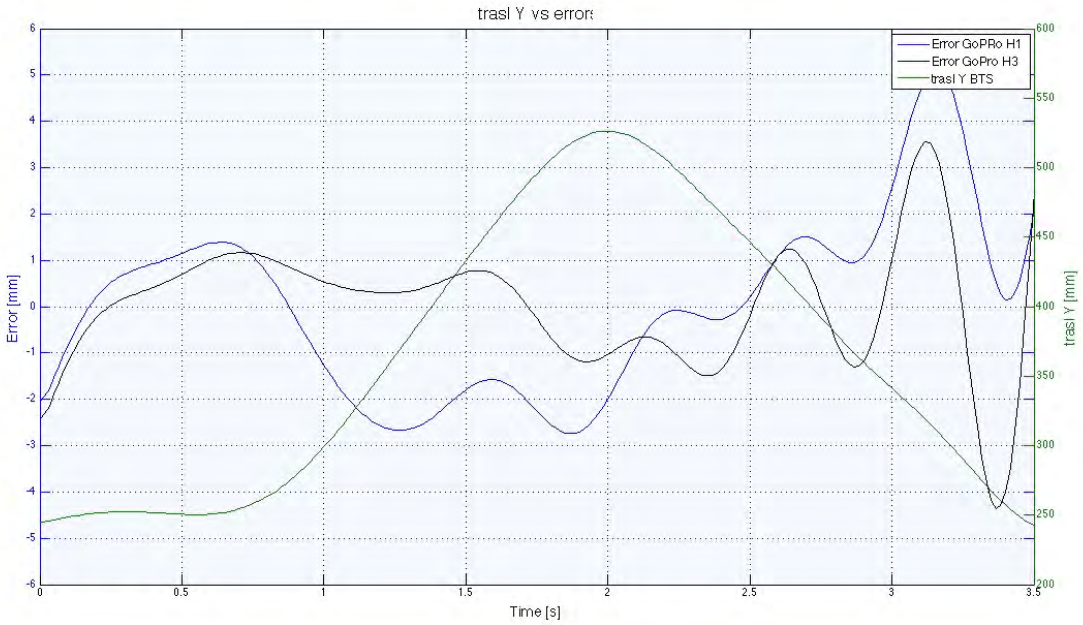


Figura 3.12: Traslazione lungo l'asse Y ed errori sulla misura ottenuta dalle telecamere.

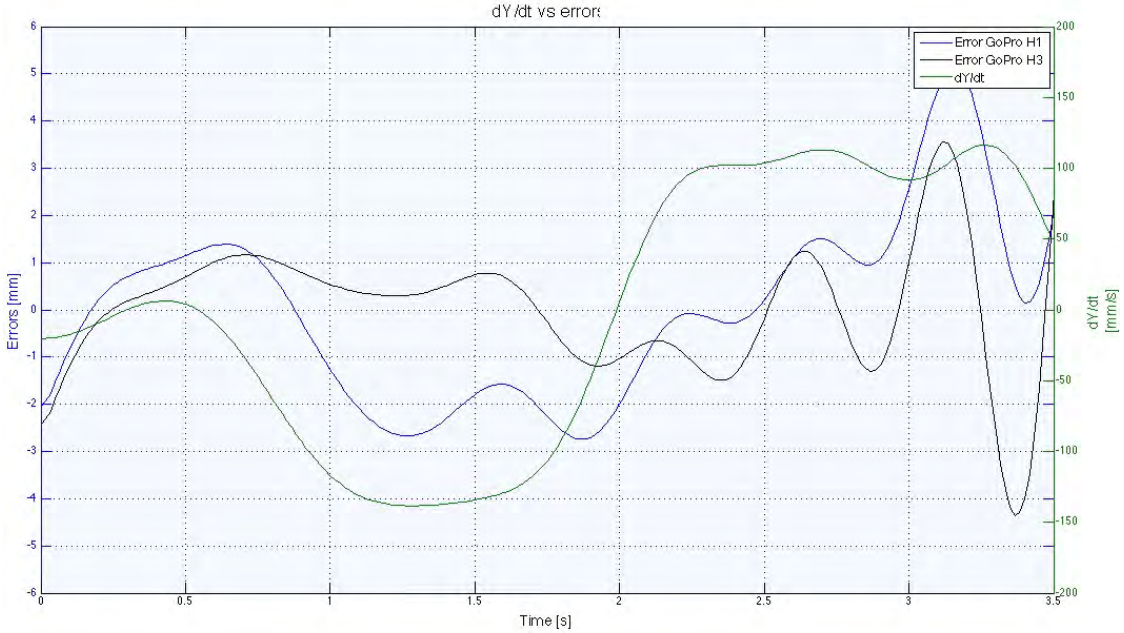


Figura 3.13: Velocità di traslazione lungo l'asse Y ed errori sulla misura ottenuta dalle telecamere.

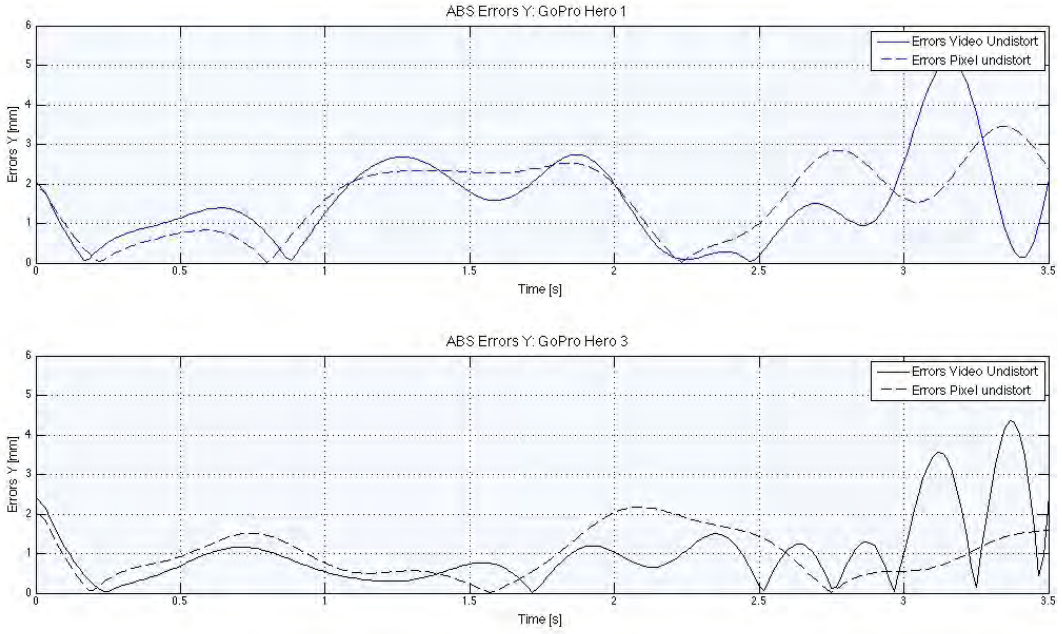


Figura 3.14: Confronto fra gli errori ottenuti con i due metodi di correzione della distorsione (valore assoluto degli errori, traslazione asse Y).

Asse Z

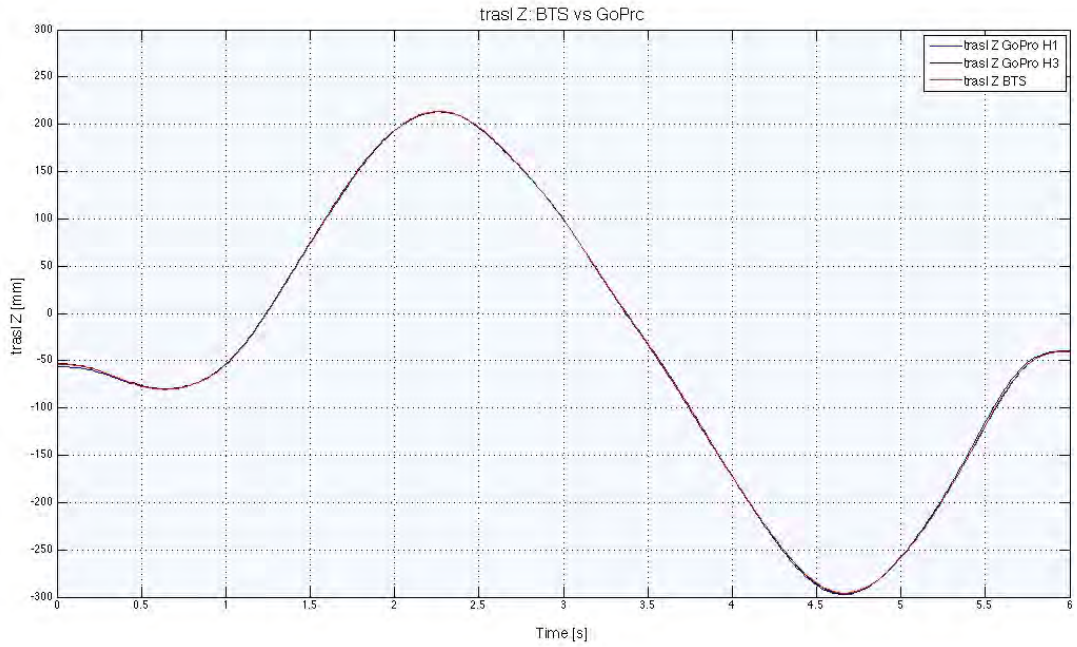


Figura 3.15: Traslazione lungo l'asse Z: confronto fra i risultati ottenuti dal *BTS motion capture system* e i risultati delle due telecamere.

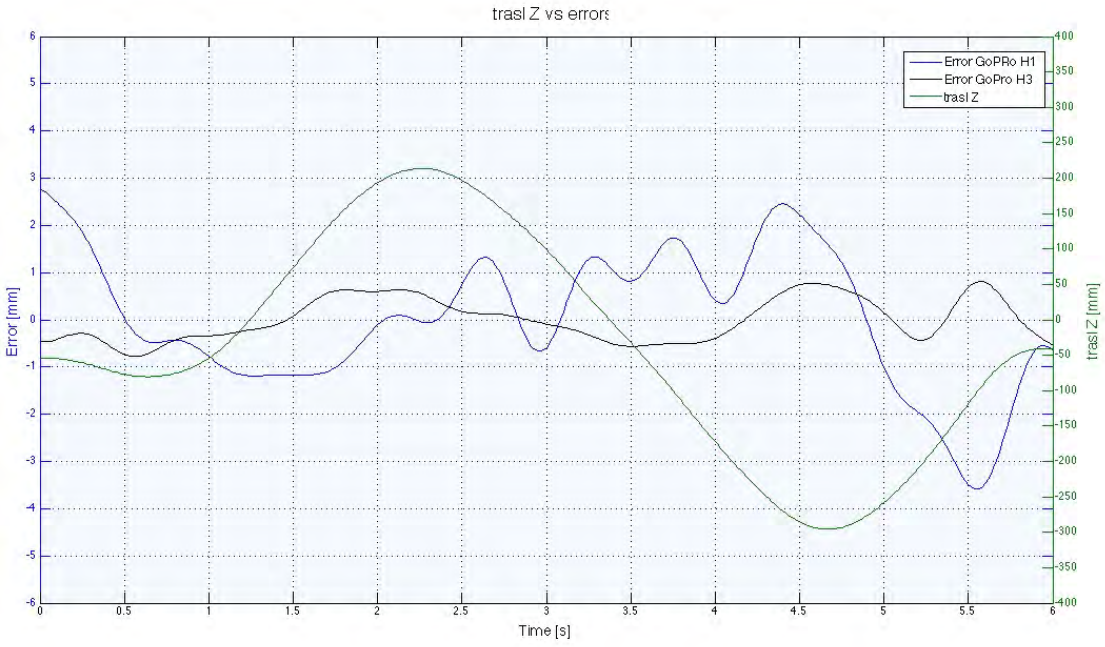


Figura 3.16: Traslazione lungo l'asse Z ed errori sulla misura ottenuta dalle telecamere.

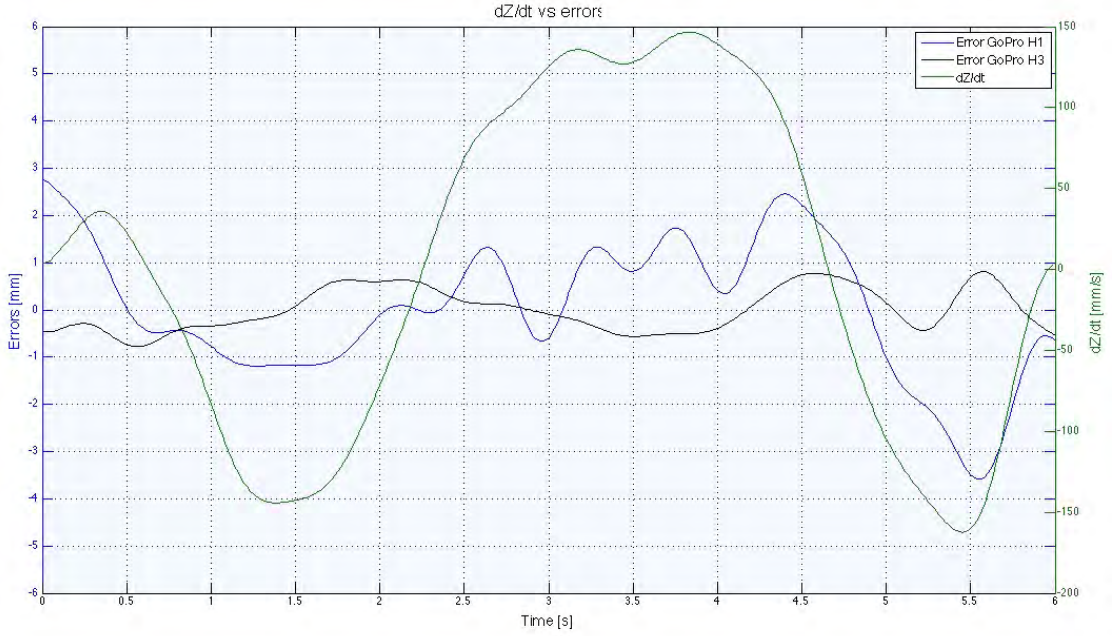


Figura 3.17: Velocità di traslazione lungo l'asse Z ed errori sulla misura ottenuta dalle telecamere.

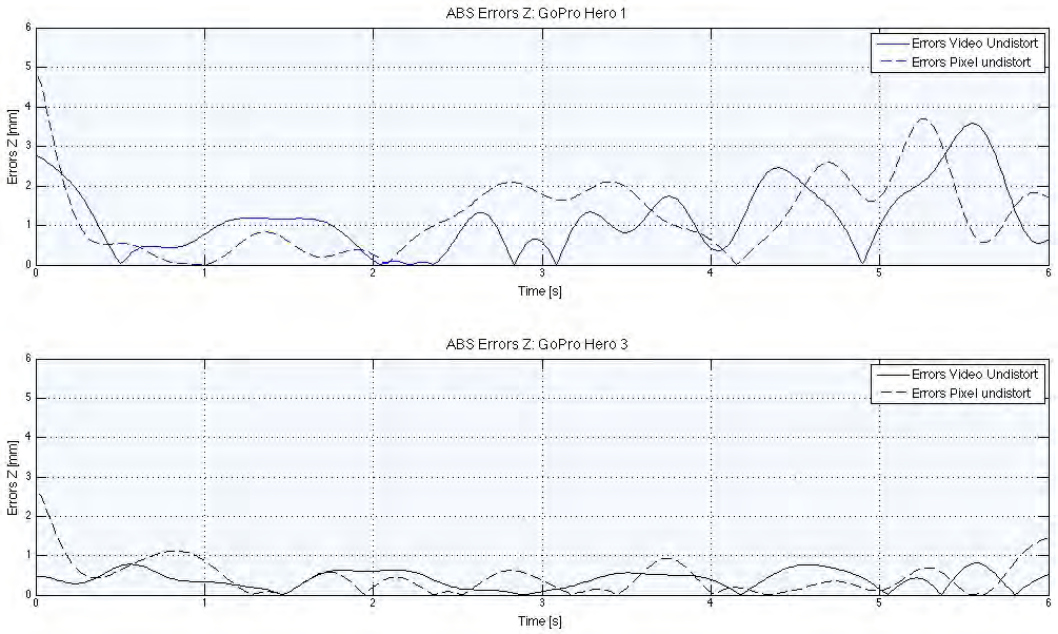


Figura 3.18: Confronto fra gli errori ottenuti con i due metodi di correzione della distorsione (valore assoluto degli errori, traslazione asse Z).

3.4.2 Rotazioni

Asse X

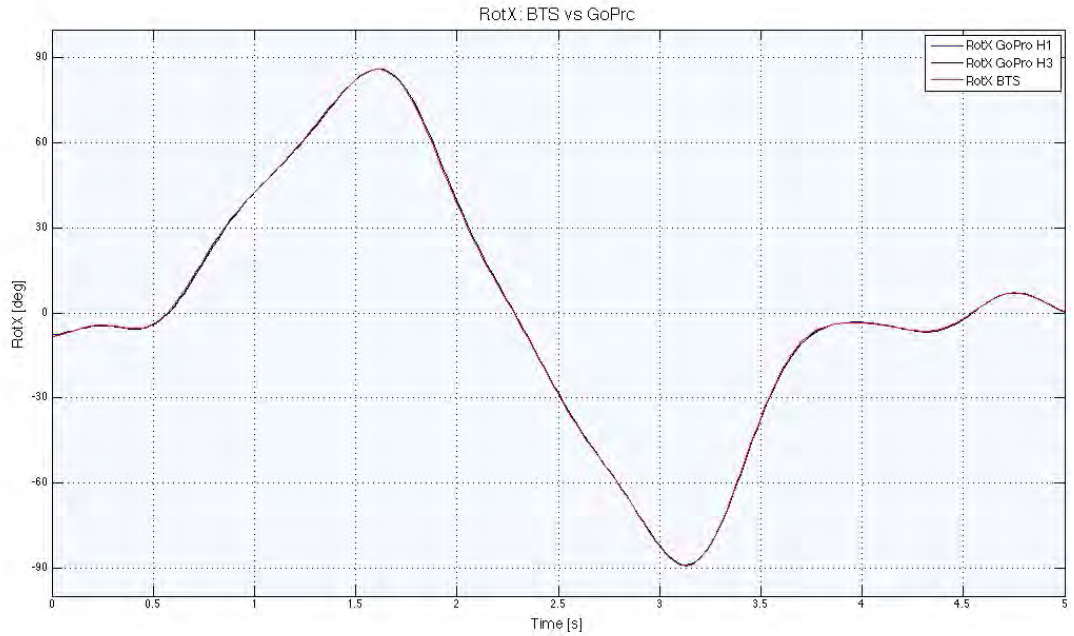


Figura 3.19: Rotazione lungo l'asse X: confronto fra i risultati ottenuti dal *BTS motion capture system* e i risultati delle due telecamere.

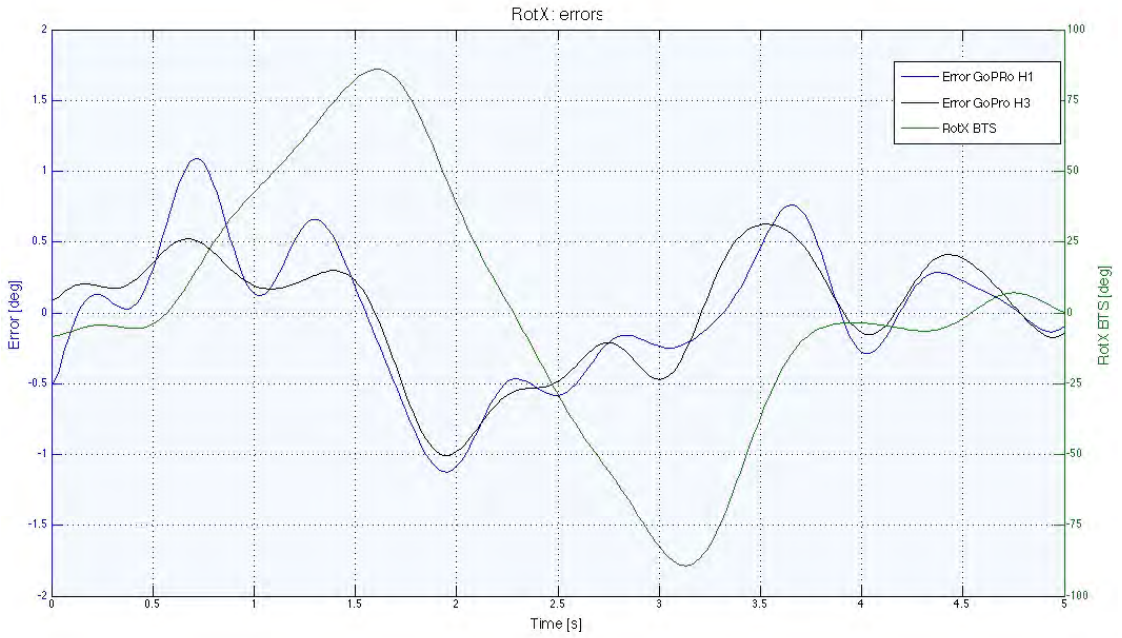


Figura 3.20: Rotazione lungo l'asse X ed errori sulla misura ottenuta dalle telecamere.

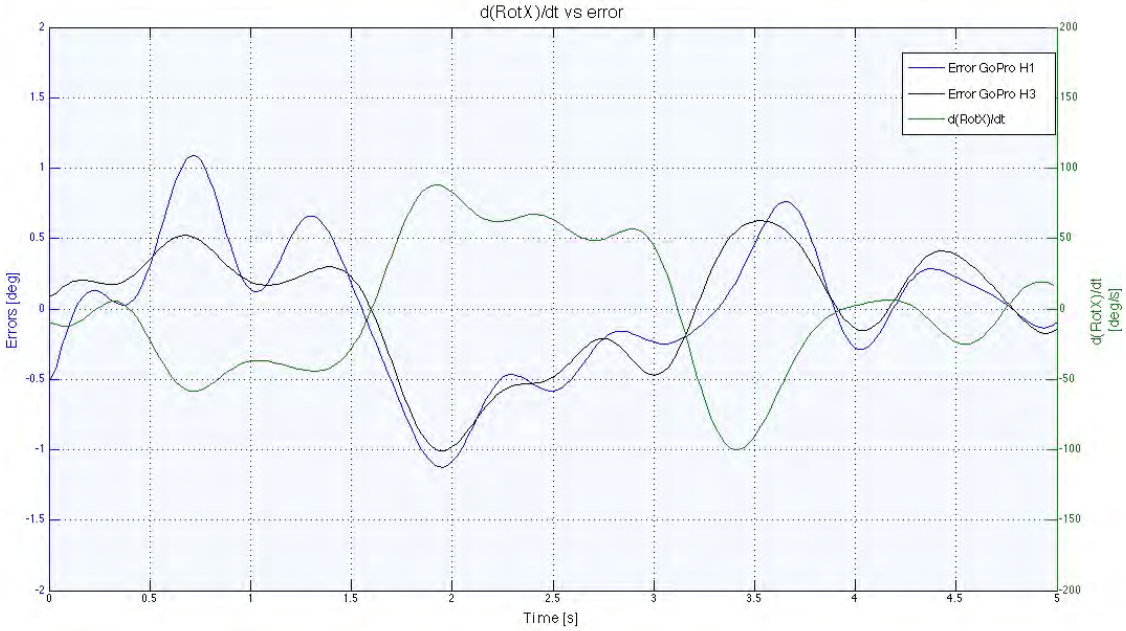


Figura 3.21: Velocità di rotazione lungo l'asse X ed errori sulla misura ottenuta dalle telecamere.

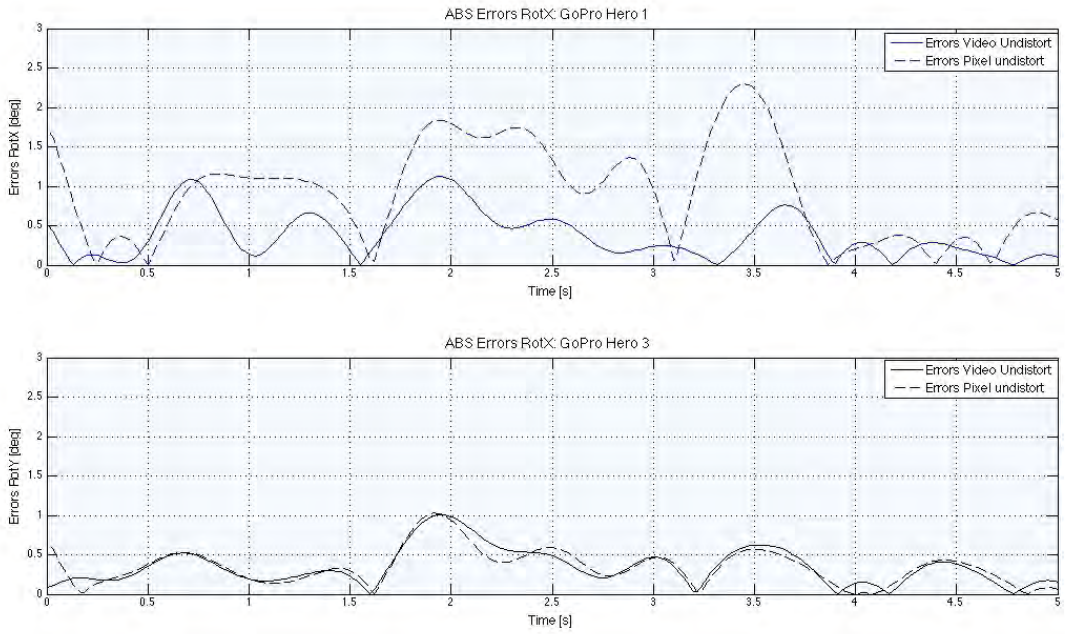


Figura 3.22: Confronto fra gli errori ottenuti con i due metodi di correzione della distorsione (valore assoluto degli errori, rotazione asse X).

Asse Y

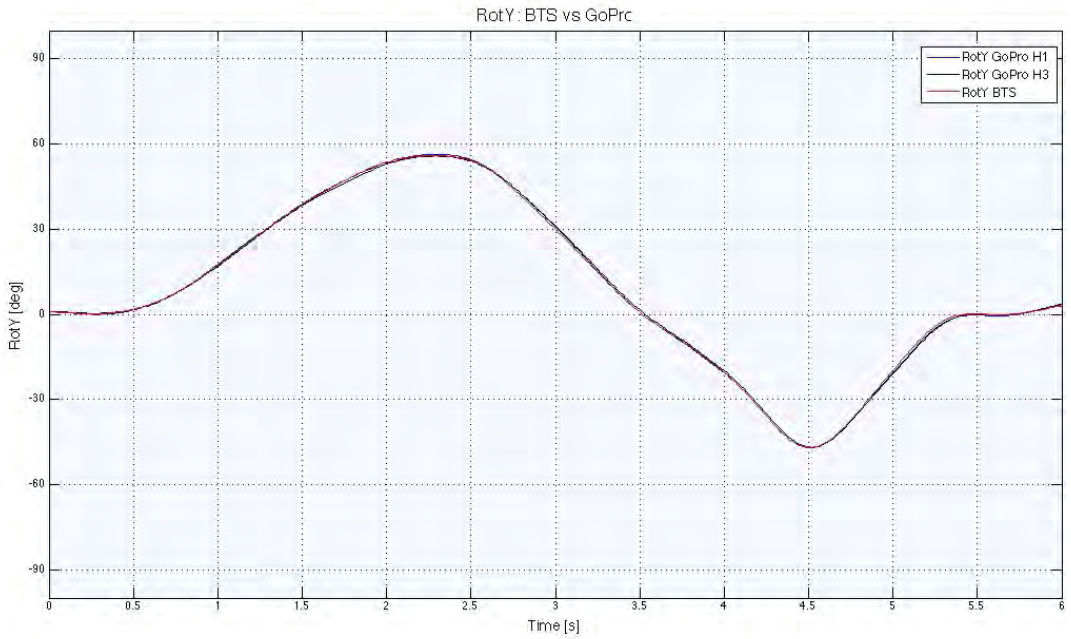


Figura 3.23: Rotazione lungo l'asse Y: confronto fra i risultati ottenuti dal *BTS motion capture system* e i risultati delle due telecamere.

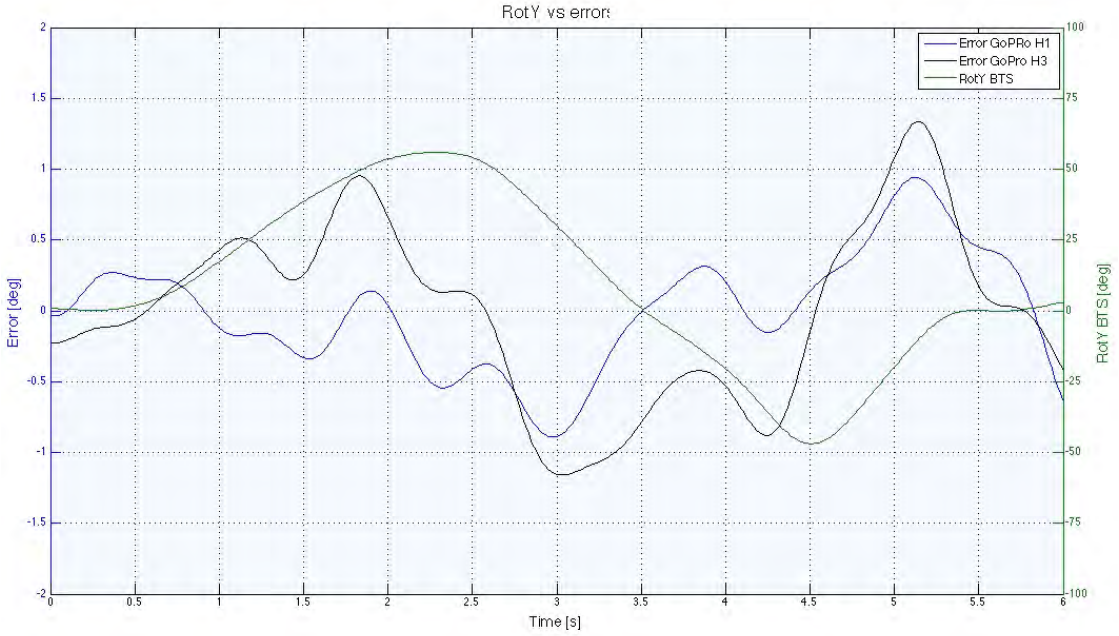


Figura 3.24: Rotazione lungo l'asse Y ed errori sulla misura ottenuta dalle telecamere.

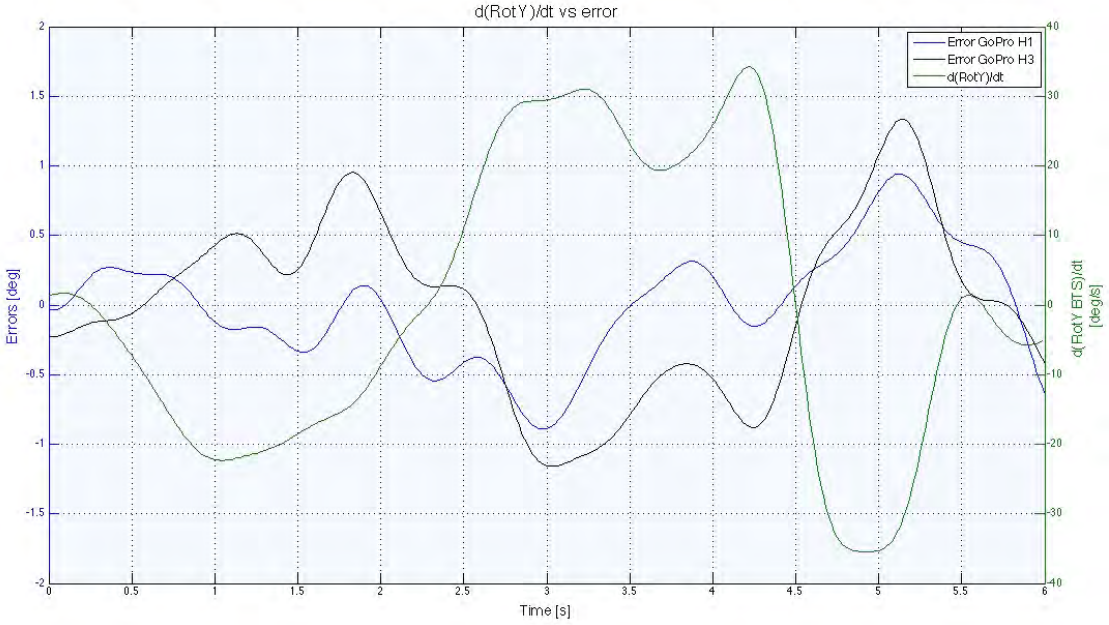


Figura 3.25: Velocità di rotazione lungo l'asse Y ed errori sulla misura ottenuta dalle telecamere.

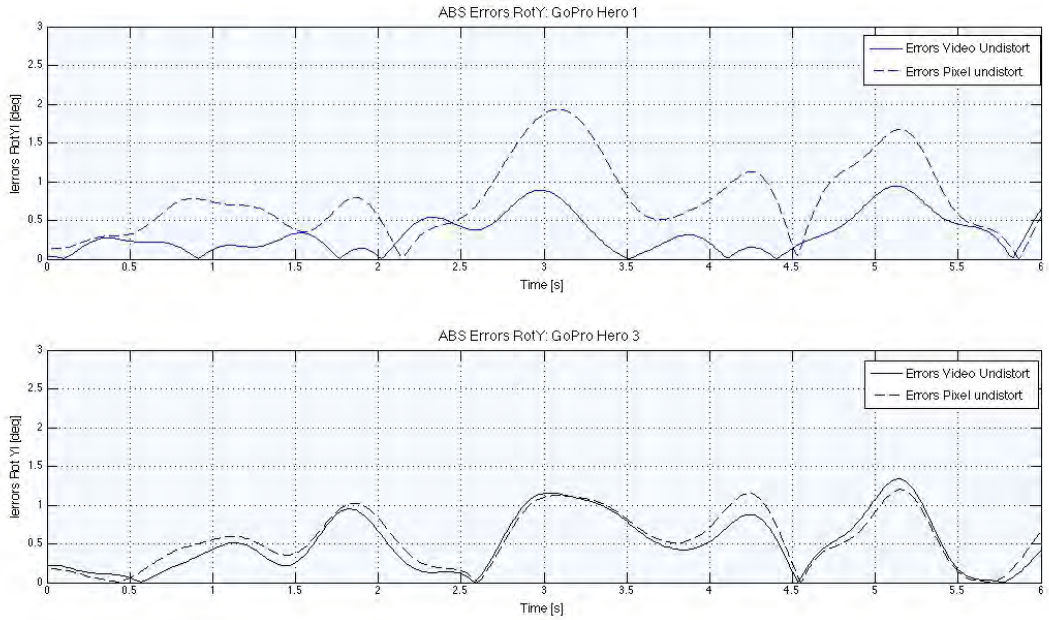


Figura 3.26: Confronto fra gli errori ottenuti con i due metodi di correzione della distorsione (valore assoluto degli errori, rotazione asse Y).

Asse Z

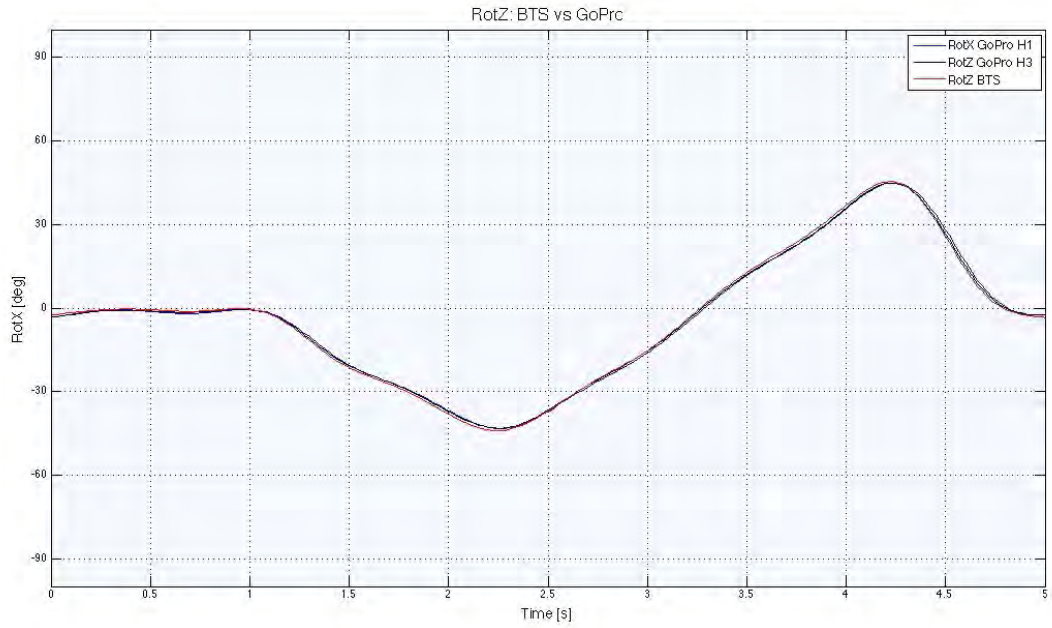


Figura 3.27: Rotazione lungo l'asse Z: confronto fra i risultati ottenuti dal *BTS motion capture system* e i risultati delle due telecamere.

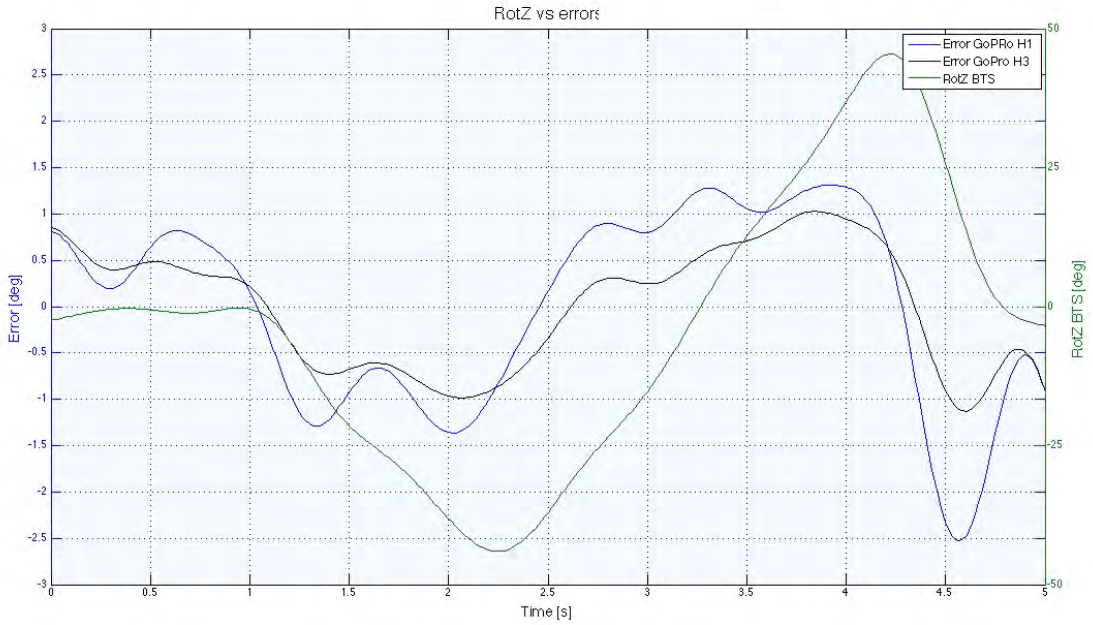


Figura 3.28: Rotazione lungo l'asse Z ed errori sulla misura ottenuta dalle telecamere.

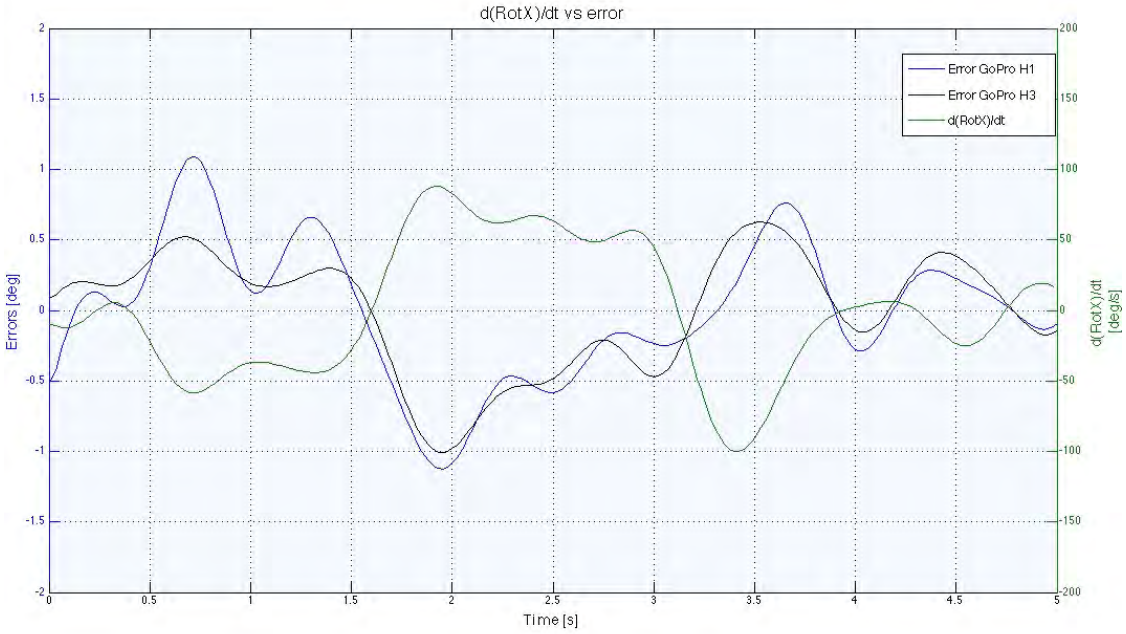


Figura 3.29: Velocità di rotazione lungo l'asse Z ed errori sulla misura ottenuta dalle telecamere.

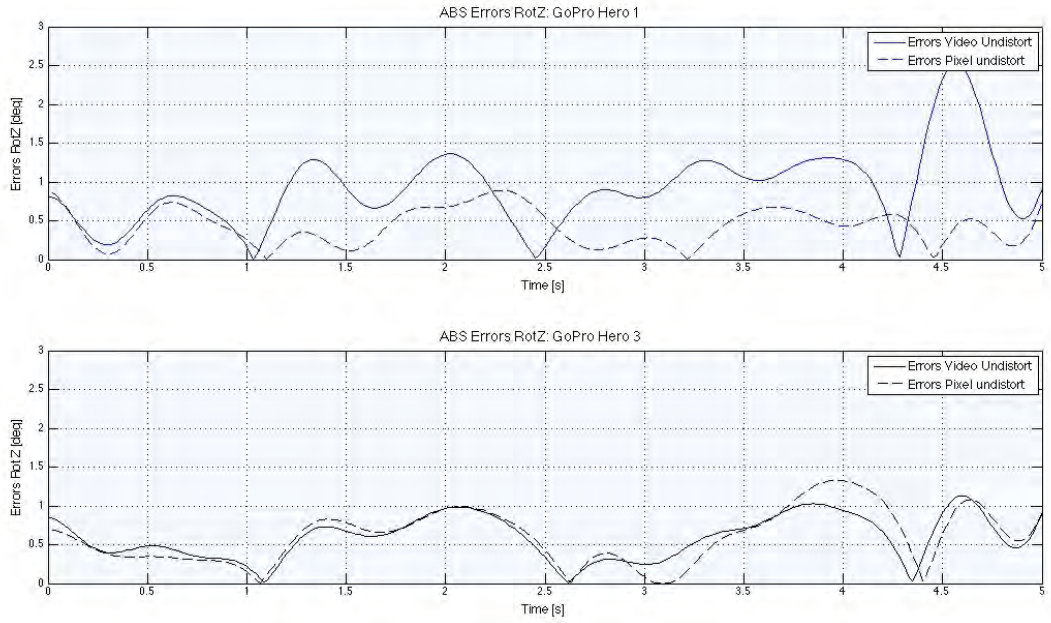


Figura 3.30: Confronto fra gli errori ottenuti con i due metodi di correzione della distorsione (valore assoluto degli errori, rotazione asse Z).

Tabella 3.1: Errori medi traslazioni e rotazioni.

	GoPro Hero1 960p @30*fps*		**GoPro Hero3+ 1440p @48*fps***	
	Video-Und.	*Pixel-Und.*	*Video-Und.*	*Pixel-Und.*
Err. X [mm]	0.61	0.53	0.91	1.29
Err. Y [mm]	1.53	1.63	1.01	0.99
Err. Z [mm]	1.17	1.25	0.40	0.46
Err. RotX [deg]	0.39	0.92	0.35	0.35
Err. RotY [deg]	0.33	0.77	0.51	0.55
Err. RotZ [deg]	0.91	0.42	0.59	0.61

Considerazioni sui risultati delle prove.

Dai grafici relativi alle traslazioni (da figura 3.7 a figura 3.18) si nota che l'errore oscilla circa attorno al valore $\pm 3.5mm$ per la *GoPro Hero1*, mentre per quanto riguarda la *GoPRo Hero3+* l'errore oscilla circa attorno al valore $\pm 2mm$.

Per quanto riguarda l'andamento dell'errore con la velocità, si nota che esiste una dipendenza dell'errore da quest'ultima. Ovvero in corrispondenza dei picchi di velocità l'errore tende al sul massimo.
Questa tendenza è si verifica nel caso delle traslazioni ma è ancora più evidente nel caso delle rotazioni. Infatti, per quest'ultime, l'andamento dell'errore segue quasi esattamente l'andamento della velocità di rotazione, però con il segno opposto.

Nei grafici è riportato anche il confronto fra gli errori valutati correggendo la distorsione all'intero video (*Video Undistort*) e correggendo la distorsione solamente alle coordinate dei pixel (*Pixel Undistort*). Si nota che la differenza tra i due metodi è minima: mediamente l'errore valutato con il metodo *Pixel Undistort* è leggermente più grande, anche se in alcuni tratti risulta essere addirittura minore dell'errore ottenuto con il metodo *Video Undistort*.

In tabella 3.2 sono riportati gli errori medi per ogni grado di libertà di entrambe le telecamere. Si può notare che la telecamera *GoPro Hero3+* è mediamente più accurata anche se la differenza con meno performante *GoPro Hero1* è minima. Questo risultato è dovuto probabilmente alla sua maggiore risoluzione rispetto alla risoluzione della *Hero1* (1440p *vs* 960p) e anche al frame rate più elevato (48 fps *vs* 30 fps).

3.4.3 Test con movimenti combinati

In questi test, a differenza dei precedenti, i movimenti effettuati sono combinati: ovvero il target è stato mosso non solo attraverso un particolare grado di libertà ma in modo casuale all'interno di un volume entro il quale il pilota si può muovere durante la guida. La velocità di spostamento in queste prove è stata molto più elevata rispetto ai test precedenti. I risultati sono mostrati nelle figure che seguono.

Traslazioni

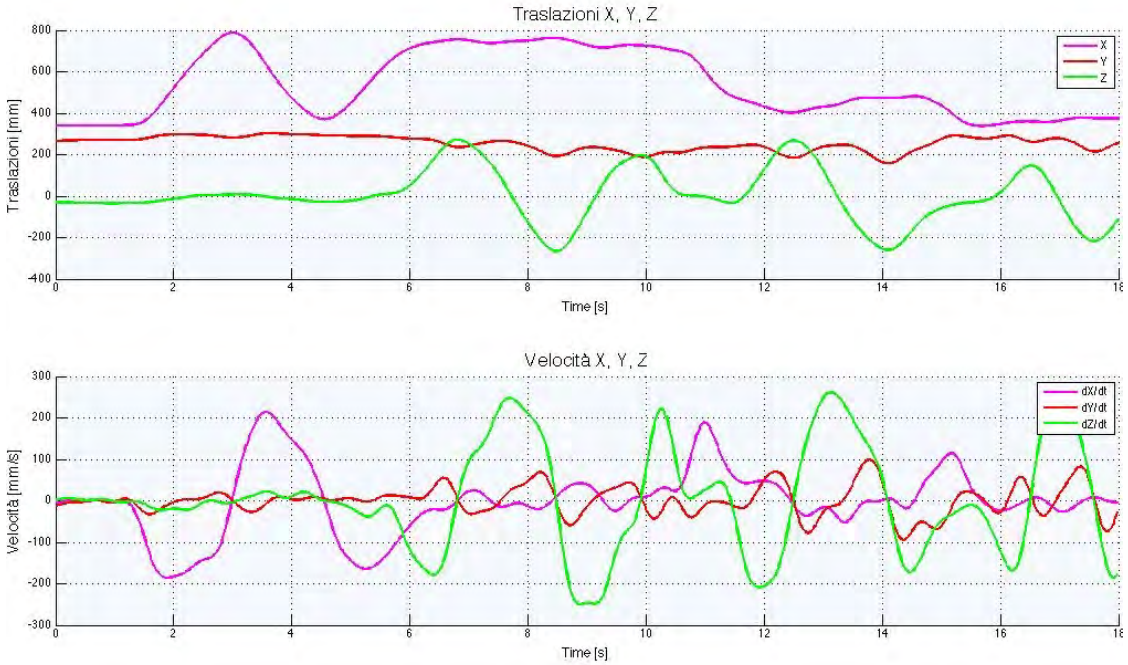


Figura 3.31: Traslazioni e velocità di traslazione lungo gli assi X, Y, Z.

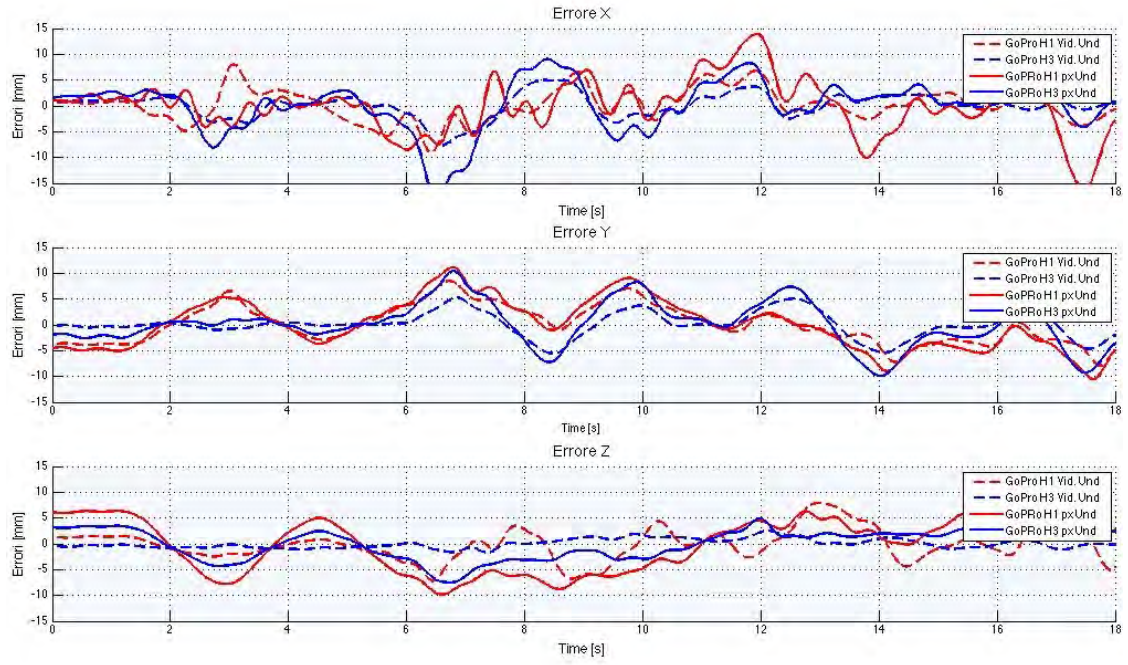


Figura 3.32: Errori telecamere.

Rotazioni

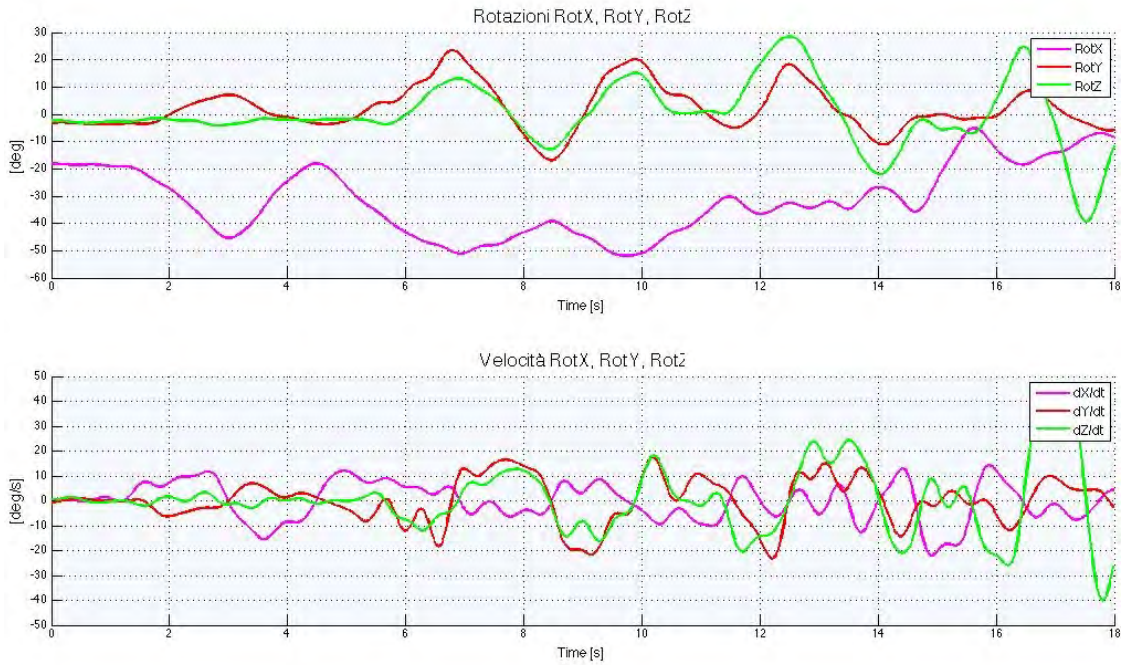


Figura 3.33: Rotazioni e velocità di rotazione attorno agli assi X, Y, Z.

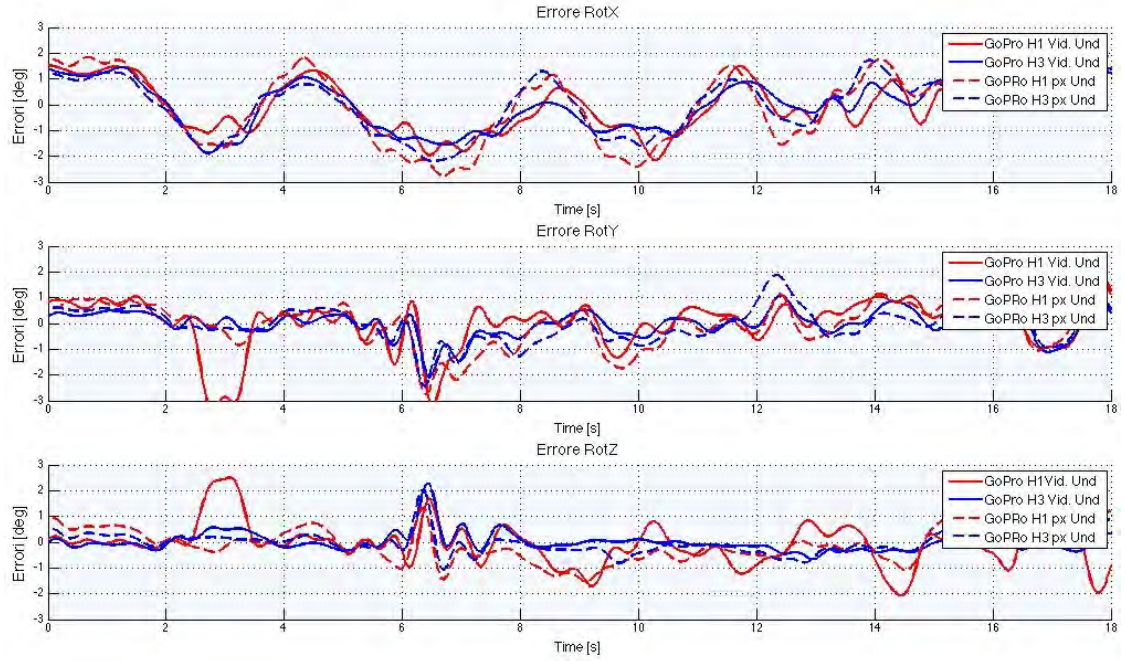


Figura 3.34: Errori telecamere.

Tabella 3.2: Errori medi traslazioni e rotazioni: test movimenti combinati.

	GoPro Hero1 960p @30*fps*		**GoPro Hero3+ 1440p @48*fps***	
	Video-Und.	*Pixel-Und.*	*Video-Und.*	*Pixel-Und.*
Err. X [mm]	2.41	3.75	1.78	3.49
Err. Y [mm]	3.08	3.97	1.71	3.23
Err. Z [mm]	2.51	4.07	0.74	2.55
Err. RotX [deg]	0.85	1.18	0.80	0.91
Err. RotY [deg]	0.68	0.71	0.39	0.57
Err. RotZ [deg]	0.65	0.62	0.24	0.33

Considerazioni sui risultati della prova.

Dai grafici relativi alle traslazioni (da figura 3.31 a figura 3.34) si nota che ora l'errore è più elevato rispetto alle prove precenti, soprattutto per le traslazioni. Questo probabilmente è dovuto in primis alle velocità di traslazione nettamente più elevate dove, nel caso delle traslazioni, si raggiungono picchi di quasi 300$[mm/s]$ rispetto ai 150$[mm/s]$ registrati nei test precedenti. In quest o caso inoltre si nota che la differenza tra le due telecamere è più marcata: la *GoPro Hero3+* si comporta meglio della versione *Hero1*, probabilmente a velocità più elevate un maggior frame rate fa la differenza. In tabella 3.2 sono riportati gli errori medi per ogni grado di libertà di entrambe le telecamere.

Parte II

Test su strada con una motocicletta strumentata

Capitolo 4

Setup strumentale

In questo capitolo verrà descritta la strumentazione installata sulla motocicletta impiegata nelle prove sperimentali. Questa strumentazione ha l'obiettivo di analizzare il comportamento del pilota durante la guida e l'interazione pilota-motocicletta. I sensori installati permettono infatti di misurare:

- forze e coppie scambiate tra il pilota e la moto;
- l'angolo di rollio;
- l'angolo di sterzo;
- la posizione del pilota rispetto alla motocicletta.

La motocicletta impiegata è una *Aprilia Mana 850*, mostrata in figura 4.1, dotata di cambio automatico. La cilindrata del motore è di $839.3\,cc$ e la potenza massima dichiarata di $56\,kW$ a $8000\,rpm$.

Una motocicletta dotata di cambio automatico permette al pilota di concentrarsi esclusivamente sulla guida del veicolo ed inoltre non è presente l'azione di cambiata esercitata dal pilota che, nel caso di una motocicletta con il cambio manuale, implicherebbe l'applicazione di forze sulla moto che non sono dovute alla pura azione di controllo del veicolo. Per altro queste forze disturberebbero il segnale ottenuto nei sensori, soprattutto nelle pedaline.

Figura 4.1: Motocicletta Aprilia Mana 850.

Tabella 4.1: Data Logger: caratteristiche.

Model	$2D^{®}$ Data Logger - id. 124
Memory	$512\,Mb$
Sampling frequency	from $6.25\,Hz$ *to* $3.2\,kHz$
№channels	256
Analogical channels	16
Digital Channel IC	8

4.1 Data Logger

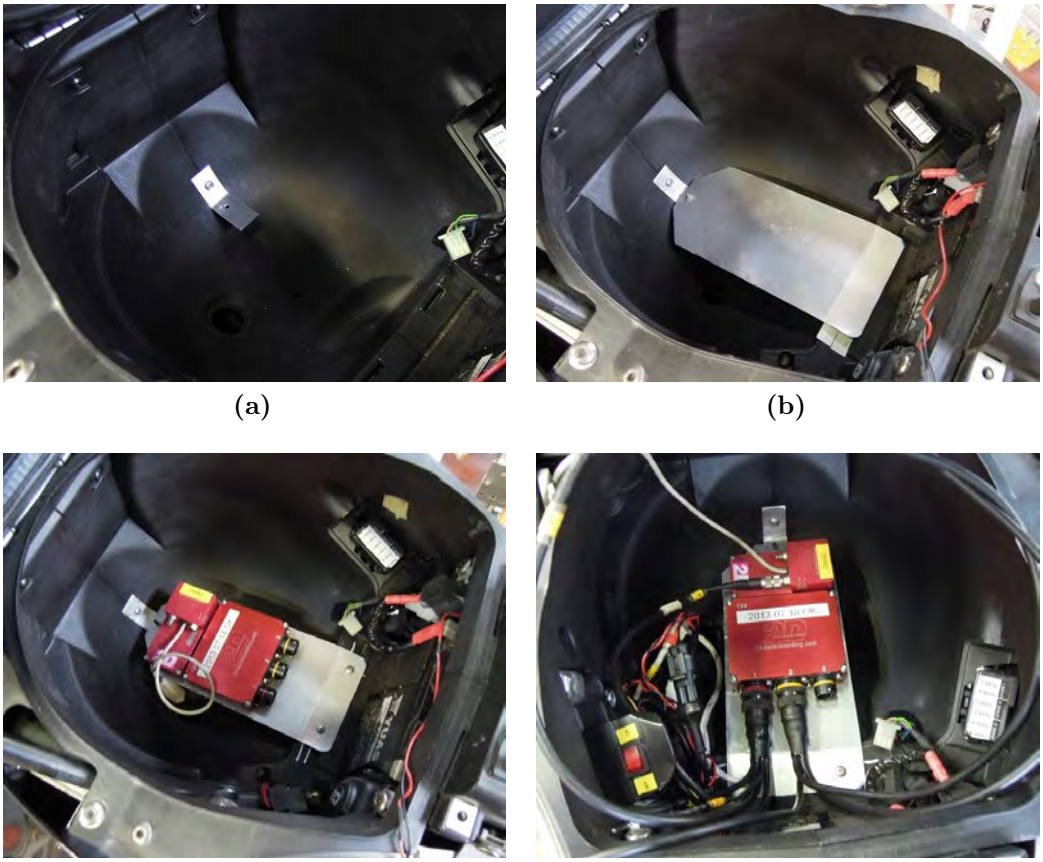
Il *Data Logger* (figura 4.2) è un componente hardware che permette di organizzare e gestire la strumentazione installata nella motocicletta e registrare i dati provenienti dai sensori. La tabella 4.1 ne riporta le caratteristiche tecniche. Il *Logger* può interfacciarsi al software di gestione tramite porta seriale Ethernet o porta USB. Nel nostro caso il collegamentro con il pc avviene tramite porta Ethernet. Nella figura 4.3 è rappresentata la sequenza dell'installazione del Logger e della piattaforma inerziale all'interno del vano casco della moto, mentre in figura 4.4 è rappresentata la mappatura dei sensori collegati al Logger.

Figura 4.2: 2D® Data Logger

4.2 Piattaforma inerziale

La piattaforma inerziale $2D^{®}$ (figura 4.5) permette di misurare:

- accelerazione longitudinale A_x;
- accelerazione laterale A_y;
- accelerazione verticale A_z;
- velocità angolare G_x;



(a) **(b)**

(c) **(d)**

Figura 4.3: Installazione del 2D® Data Logger.

- velocità angolare G_y;
- velocità angolare G_z;

con una risoluzione di $0.02\,[m/s^2]$ ed un fondo scala di $\pm 6\,textg$.

Inoltre se dotato di antenna *GPS* (figura 4.5), questo strumento permette di misurare e ricostruire il tracciato percorso. L'antenna *GPS* acquisisce ad una frequenza di $6\,Hz$ i seguenti canali:

- latitudine;
- lungitudine;
- velocità assoluta sulla superficie terrestre $[km/h]$;
- accelerazione longitudinale assoluta (riferita alla traiettoria ricostruita) $[m/s^2]$;
- accelerazione laterale assoluta (riferita alla traiettoria ricostruita) $[m/s^2]$;

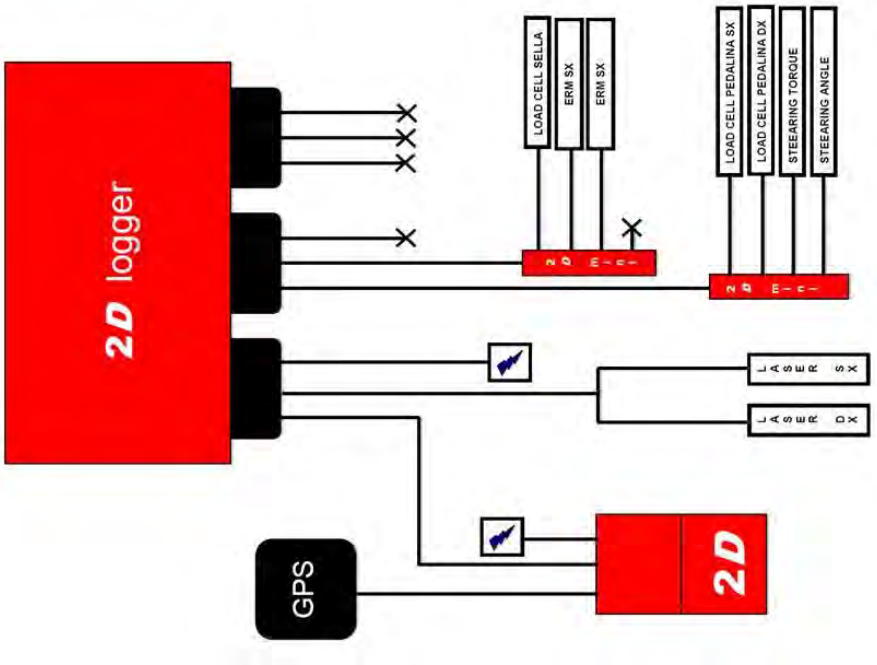


Figura 4.4: Mappatura dei sensori

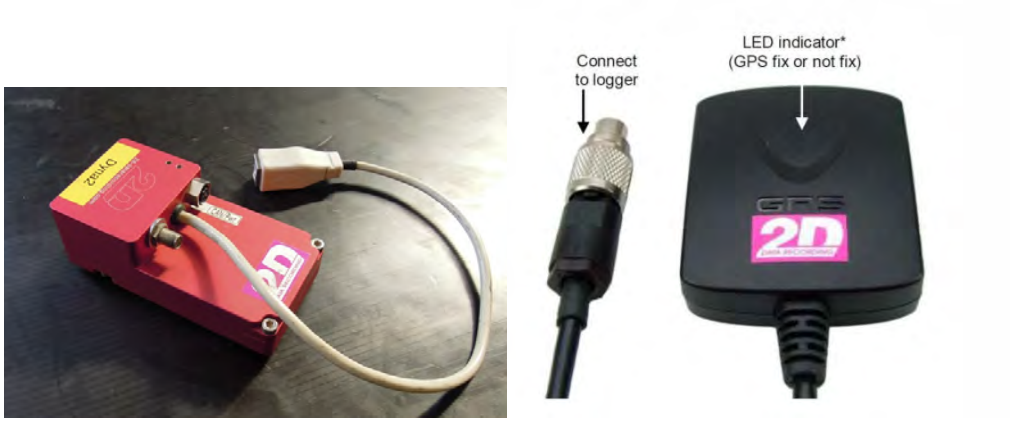
4.3 Sensore di coppia allo sterzo e di coppia al manubrio

Il sensore di coppia allo sterzo (figura 4.6) è stato realizzato dal gruppo di ricerca MDRG *Motorcycle dynamics research group*. La coppia allo sterzo viene rilevata direttamente nel punto di fissaggio del manubrio. Il sensore risulta essere sensibile sia ai bassi valori di coppia, tipici delle manovre stazionarie in percorrenza di curva, che agli alti valori. tipici di manovre di emergenza o di cambio improvviso di direzione. Il sensore misura valori positivi per momenti applicati in senso orario, osservando la moto dall'alto verso il basso. Infine la taratura di questo sensore risultava già tarato in un precedente lavoro di tesi.

Il sensore di coppia applicata al manubrio (figura 4.6) misura il momento ortogonale all'asse di sterzo applicato dal pilota . La coppia è positiva se applicata il senso orario.

4.4 Sensore per la misura dell'angolo di sterzo

La misura dell'angolo di sterzo è effettuata tramite un potenzionetro rotativo magnetico $2D^{®}$ ad alta risoluzione (figura 4.7). In tabella 4.2 sono riassunte le sue caratteristiche. Il sensore misura valori positivi ruotando lo sterzo in senso orario (rotazioni verso il lato destro della moto).



(a) (b)

Figura 4.5: Piattaforma inerziale e sensore GPS.

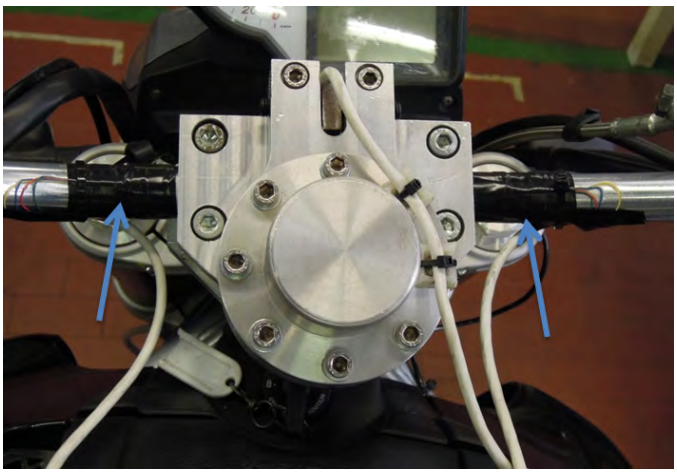


Figura 4.6: Sensore di coppia allo sterzo e sensori di coppia al manubrio (indicati dalle frecce azzurre).

4.5 Celle di carico sulle pedane

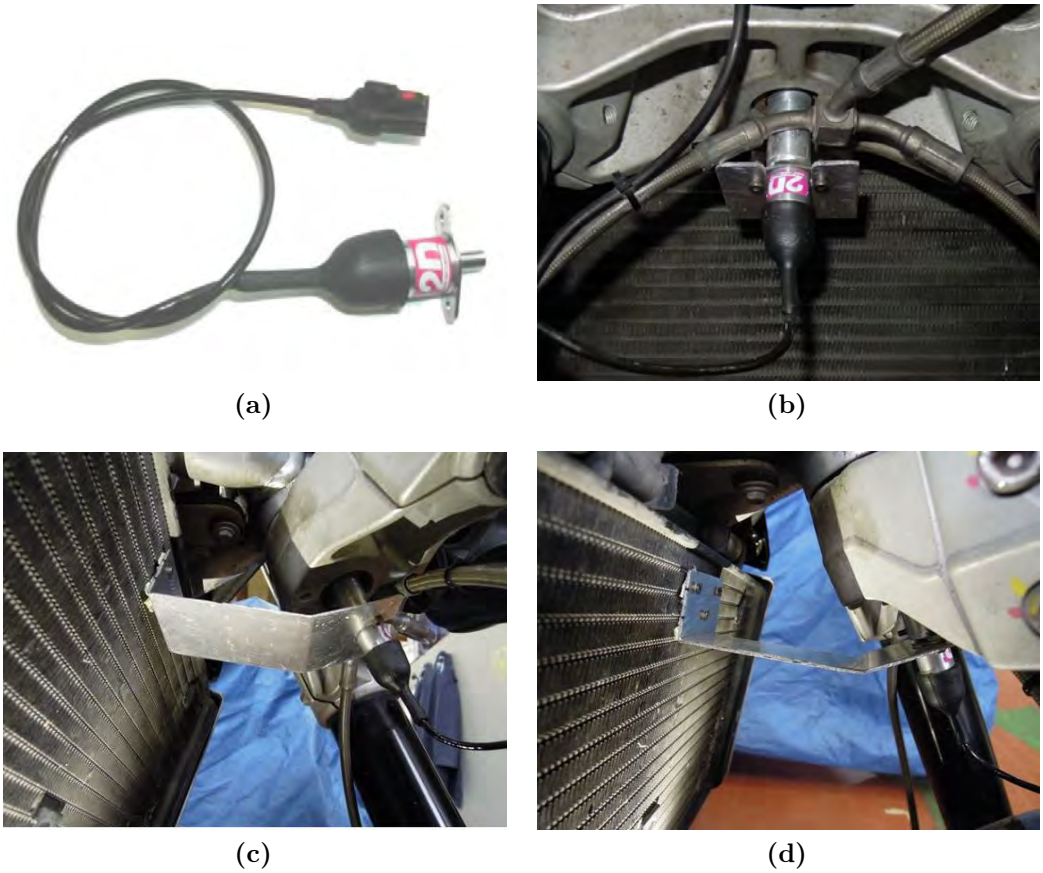
Come per il sensore di coppia allo sterzo, anche questi due sensori (figura 4.8) sono stati interamente progettati ed assemblati dal gruppo di ricerca *MDRG*. La strumentazione consiste in due celle di carico per la misura delle forze normali applicate dal pilota sulle pedane.

4.6 Sensore di coppia alla sella

Il sensore per la misura della coppia alla sella (figura 4.9) è stato realizzato dall'ing. Giolo del gruppo di ricerca *MDRG*. La sella originale è stata sostituita con una nuova sella montata su un supporto in grado ruotare attorno all'asse longitudinale della motocicletta. Questo supporto a sua volta è collegato ad una cella di carico che misura il momento applicato in direzione longitudinale, è in grado

Output range [V]	$0 - 5$
Output range @ lower resolution [°]	$\pm 40°$
Max range of linearity [°]	$\pm 40°$
Resolution	??

Tabella 4.2: Potenziometro rotativo: caratteristiche.



(a) **(b)** **(c)** **(d)**

Figura 4.7: Sensore angolo di sterzo.

quindi di misurare la coppia sulla sella dovuta al rollio del pilota. Il momento è positivo se il pilota rolla verso destra.

4.7 Sensori ottici per la misura della distanza

I sensori installati (figura 4.10) sono il modello *E-shock® RAY*. Si tratta di sensori laser a triangolazione per la misura di distanze appositamente studiati per applicazioni motociclistiche. In tabella 4.3 sono riassunte le caratteristiche principali.

(a) (b)

Figura 4.8: Sensore pedane.

Figura 4.9: Sensore di coppia allo sella.

4.8 Telecamera

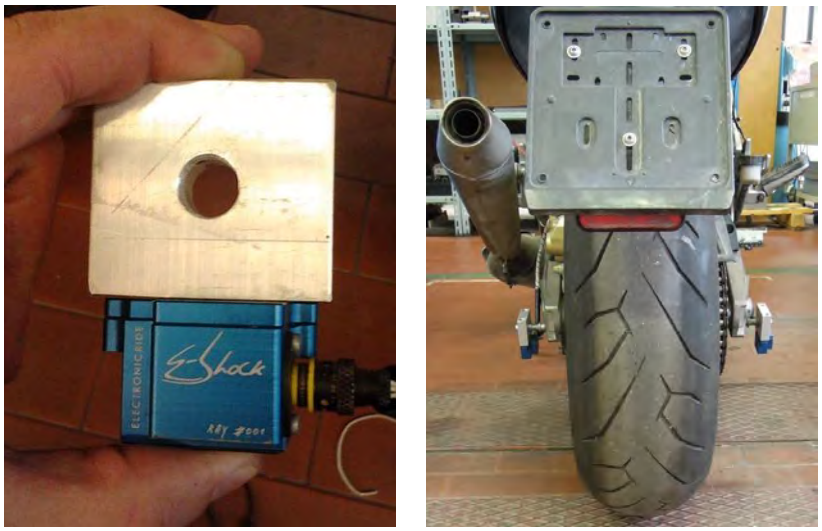
La telecamera per la misura della posizione del pilota installata sulla moto è la *GoPro Hero HD* ed è stata posizionata nel codone della moto (figura 4.11) a *soli* 25 *cm* dal pilota però, grazie all'ampio fov di questa camera (170°) il pilota è ripreso interamente.

4.9 Software di gestione dei dati

I dati delle misurazioni ottenuti dalle centraline $2D°$ vengono gestiti dal software *2D WinARace* appositamente creato per questo scopo. Tramite questo software è stato possibile visualizzare i dati acquisiti per ogni canale/sensore, selezionare i tratti di interesse ed esportare i dati in formato di testo da poter essere gestiti in MatLab. Questo software inoltre ricostruisce il tracciato percorso con i dati ottenuti dal *GPS* (se disponibili).

Tabella 4.3: Caratteristiche dei sensori laser *E-shock® RAY* per misura di distanze.

Measurement range]	$from\,100\,mm\,to\,1000\,mm$
Laser Power	$< 5\,mW$
Wawe lenght	$7801, nm$
Temperature range	$from\ -10℃\,to\ +70℃$



(a) **(b)**

Figura 4.10: Sensori laser.

Figura 4.11: Telecamera installata sul codone della moto.

Capitolo 5

Prove sperimentali

In questo capitolo saranno elaborate le prove effettuate con la motocicletta strumentata vista nel capitolo precedente. In particolare si valuterà se è possibile determinare la coppia di rollio applicata dal pilota considerando il solo spostamento laterale del pilota rispetto alla motocicletta. Questa coppia sarà confrontata con la coppia totale misurata dai sensori installati sulla motocicletta e sarà calcolato il coefficiente di correlazione tra le due coppie.

5.1 Calcolo della coppia generata dal pilota

La figura 5.1 mostra il modello implementato per il calcolo della coppia generata dallo spostamento laterale del pilota rispetto al piano della motocicletta, spostamento che viene quantificato dall'angolo α ovvero l'angolo fra la retta passante per il punto di contatto e il baricentro del pilota e la retta passante per il punto di contatto e il baricentro della moto (che assumiamo appartenere al piano di simmetria della moto). In questo modello sono presenti le seguenti semplificazioni:

- il sistema è considerato in equilibrio;
- si assume che la direzione della risultante $\vec{R}$ passi per il punto di contatto della ruota posteriore.

In condizioni di equilibrio, la risultante R della forza centrifuga e della forza peso attraversa la linea che congiunge i punti di contatto dei pneumatici sul piano stradale. Questa linea giace sul piano del motocliclo se le ruote hanno spessore nullo e l'angolo di sterzata è molto piccolo. Questa condizione di equilibrio corrisponde alla percorrenza di una curva a raggio costante ed a velocità costante, inoltre considerare nullo lo spessore del pneumatico comporta avere un angolo di rollio leggermente più piccolo dell'angolo di rollio effettivo.

La coppia generata dallo spostamento laterale del pilota è calcolata rispetto al *punto-sella* (P_S in figura 5.1) ed è data dalla seguente:

$$\vec{M}_p = \vec{b} \times \vec{R} \tag{5.1}$$

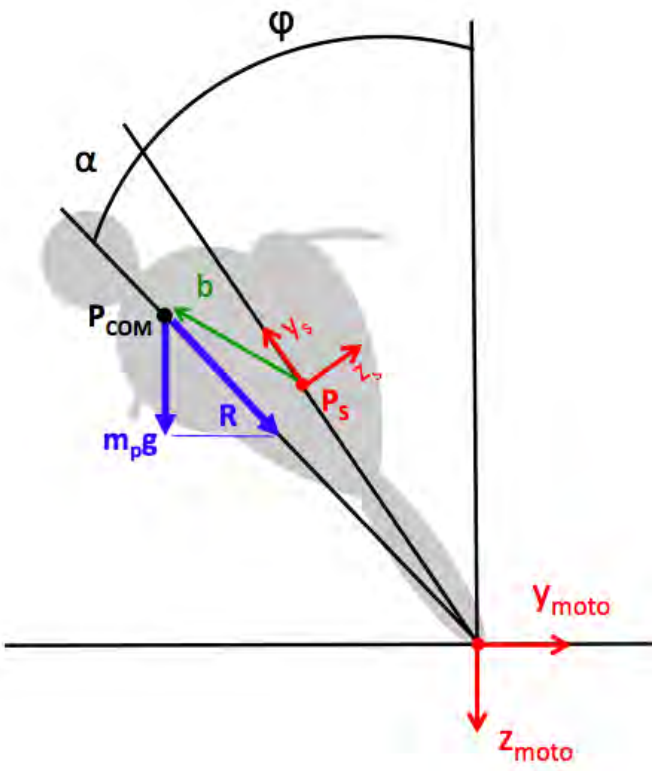


Figura 5.1: Modello per il calcolo della coppia del pilota.

Si ha che:

$$R = \frac{m_P \cdot \mathrm{g}}{\cos(\varphi + \alpha)} \tag{5.2a}$$

$$b = \big(z_s - y_s \tan\alpha\big) \cdot \cos\alpha \tag{5.2b}$$

l'angolo α è piccolo quindi si può considerare $\cos\alpha \approx 1$, si ottiene:

$$b \simeq \big(z_s - y_s \tan\alpha\big) \tag{5.2c}$$

la coppia è quindi:

$$M_p = \big(z_s - y_s \tan\alpha\big) \cdot \frac{m_P \cdot \mathrm{g}}{\cos(\varphi + \alpha)} \tag{5.3}$$

y_s e z_s sono le coordinate del baricentro del pilota rispetto al sistema di riferimento centrato sul *punto-sella* (figura 5.2)).

La posizione del pilota ${}^{Tm}\mathbf{T}_{Tp}$ che si ottiene dal sistema di *pose-estimation* è espressa rispetto al sistema di riferimento del *target-moto* (figura 5.2). É necessario perciò trasformare questo sistema di riferimento nel sistema di riferimento centrato nel *punto-sella* per ottenere la posizione del pilota espressa rispetto ad esso ed orientare inoltre il sistema di riferimento del *taget pilot* come il sistema di riferimento del *punto-sella*.

La trasformazione si può scrivere:

$${}^{S}[\mathbf{T}]_{Tps} = {}^{S}[\mathbf{T}]_{Tm} \cdot {}^{Tm}[\mathbf{T}]_{Tp} \cdot {}^{Tp}[\mathbf{T}]_{Tps} \tag{5.4}$$

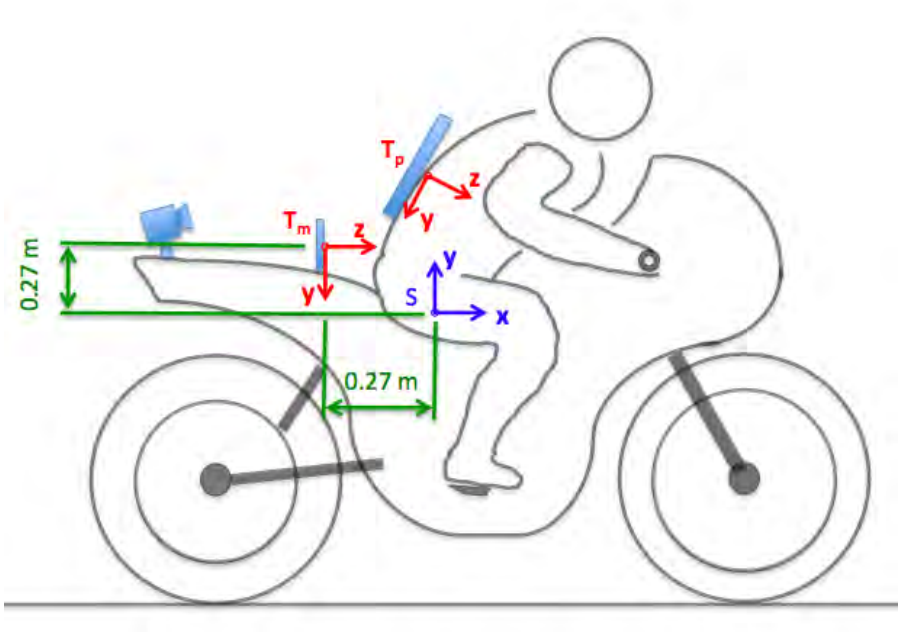


Figura 5.2: Sistemi di riferimento: target-moto, target-pilota e punto sella.

con:

$$ {}^{S}[\mathbf{T}]_{Tm} = T_{trasl}\big(-0.270, 0.270, 0\big)\cdot\Big[T_{roty}\big(\pi/2\big)\cdot T_{rotz}\big(\pi\big)\Big]\cdot T_{rotx}\Big(-7°\frac{\pi}{180°}\Big) \tag{5.5} $$

$$ {}^{Tp}[\mathbf{T}]_{Tps} = \Big[T_{roty}\big(\pi/2\big)\cdot T_{rotz}\big(\pi\big)\Big]^{-1} \tag{5.6} $$

dalla matrice ${}^{S}[\mathbf{T}]_{Tps}$ si ricavano le coordinate $(x_p, y_p, z_p$ del pilota rispetto al sistema di riferimento centrato nel *punto-sella*.

La formula 5.3 necessita del valore di α, φ ed m_P che verranno determinati nei paragrafi che seguono.

5.1.1 Calcolo dell'angolo di rollio

L'angolo di rollio è calcolato tramite i sensori di distanza installati sulla moto. In riferimento alla figura 5.3 si può scrivere che:

$$ \begin{cases} d_1 &= h + L_1 \tan\varphi \\ d_2 &= h - L_2 \tan\varphi + \Delta z \end{cases} \tag{5.7} $$

quindi, ricavando $\tan\varphi$:

$$ \begin{cases} \tan\varphi &= \frac{d_1 - h}{L_1} \\ \tan\varphi &= \frac{-d_2 + h + \Delta z}{L_2} \end{cases} \tag{5.8} $$

e sommando le due equazioni si ottiene:

$$ \tan\varphi = \frac{L_2\big(d_1 - h\big) + L_1\big(h + \Delta z - d_2\big)}{2L_1L_2} \tag{5.9} $$

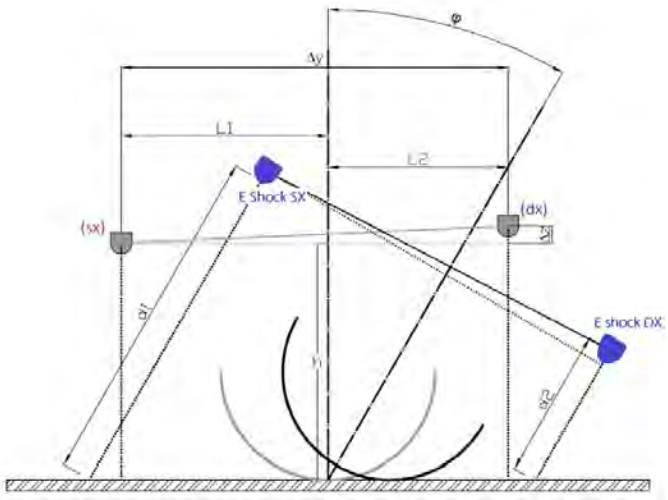


Figura 5.3: Calcolo dell'angolo di rollio.

Tabella 5.1: Distanze sensori laser.

L_1	204.5	mm
L_2	204.5	mm
h	210	mm
d_1	210	mm
d_2	220	mm
Δz	10	mm

quindi l'angolo di rollio φ è:

$$\varphi = \arctan \frac{L_2(d_1 - h) + L_1(h + \Delta z - d_2)}{2L_1L_2} \tag{5.10}$$

dove:

L_1 è la distanza del sensore ottico di sinistra rispetto al piano di mezzeria della motocicletta;

L_2 è la distanza del sensore ottico di destra rispetto al piano di mezzeria della motocicletta;

h altezza dal terreno della linea orizzontale passante per il sensore di sinistra;

d_1 distanza dal sensore di sinistra al terreno in [mm];

d_2 distanza dal sensore di destra al terreno in [mm];

Δz distanza verticale tra i due laser, ovvero la differenza di misura tra i due sensori per angolo di rollio nullo.

I dati delle misure effettuate sulla motocicletta sono riportati in tabella 5.1.

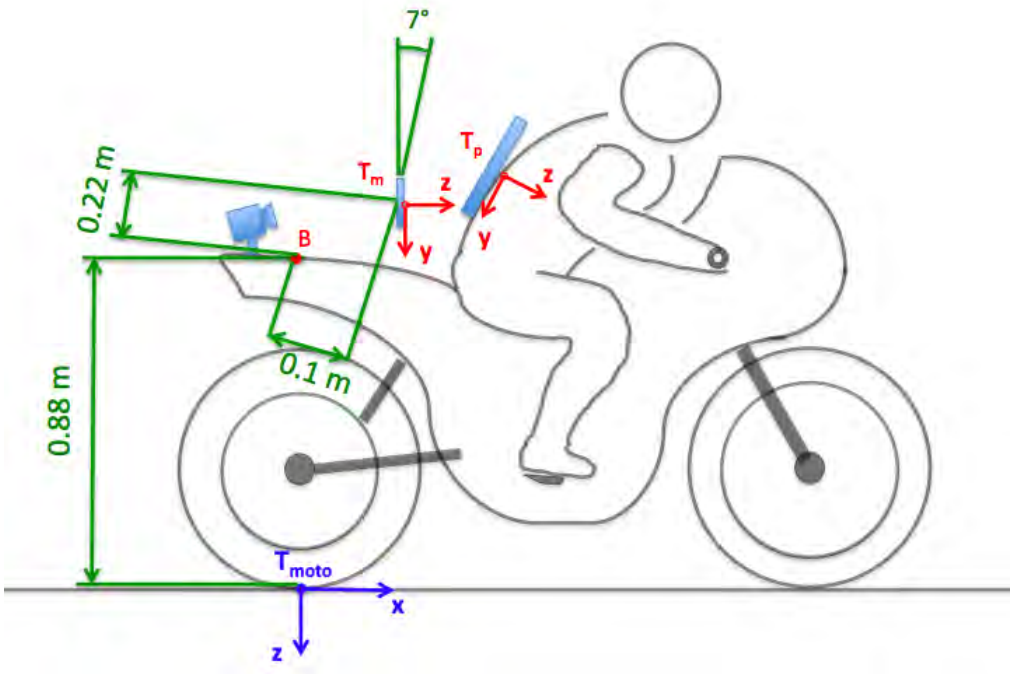


Figura 5.4: Sistemi di riferimento: target-moto, target-pilota e sistema di riferimento motocicletta.

5.1.2 Calcolo dell'angolo α

In riferimento alla figura 5.1, l'angolo α è dato dalla seguente:

$$\alpha = \arctan\left(\frac{y_p}{-z_p}\right) \tag{5.11}$$

dove y_p e z_p sono le coordinate del baricentro del pilota rispetto al sistema di riferimento della motocicletta.

Anche in questo caso la posizione del pilota ${}^{Tm}\mathbf{T}_{Tp}$ che si ottiene dal sistema di *pose-estimation* è espressa rispetto al sistema di riferimento del *target-moto* (figura), quindi come prima, è necessario trasformare rispetto al sistema di riferimento della motocicletta (figura 5.4) per ottenere la posizione del pilota espressa rispetto ad esso ed orientare inoltre il sistema di riferimento del *taget pilot* come il sistema di riferimento della moto. La trasformazione si può scrivere:

$${}^{moto}[\mathbf{T}]_{Tp_M} = {}^{moto}[\mathbf{T}]_{Tm} \cdot {}^{Tm}[\mathbf{T}]_{Tp} \cdot {}^{Tp}[\mathbf{T}]_{Tp_M} \tag{5.12}$$

con:

$${}^{moto}[\mathbf{T}]_{Tm} = {}^{moto}[\mathbf{T}]_{cam} \cdot {}^{cam}[\mathbf{T}]_{B} \cdot {}^{B}[\mathbf{T}]_{Tm} \tag{5.13}$$

$$\begin{aligned}
{}^{moto}[\mathbf{T}]_{cam} &= T_{roty}(\pi/2) \cdot T_{rotz}(\pi/2) && \text{(5.14a)}\\
{}^{cam}[\mathbf{T}]_{B} &= T_{trasl}(0, -0.880, 0) \cdot T_{rotx}\left(-7° \frac{\pi}{180°}\right) && \text{(5.14b)}\\
{}^{B}[\mathbf{T}]_{Tm} &= T_{trasl}(0, -0.220, 0.100) && \text{(5.14c)}\\
{}^{Tp}[\mathbf{T}]_{Tp_M} &= {}^{moto}[\mathbf{T}]^{-1}_{cam} && \text{(5.14d)}
\end{aligned}$$

La matrice ${}^{S}[\mathbf{T}]_{Tps}$ contiene le coordinate del pilota rispetto al sistema di riferimento della motocicletta.

5.1.3 Modellizzazione del pilota secondo Dempster

Il pilota è stato modelizzato secondo i parametri del modello di *Dempster*. Questo modello prevede la suddivisione del corpo in *segmenti* collegati da *articolazioni*. Dempster, tramite i risultati ottenuti da uno studio effettuato su numerosi soggetti, ha definito lunghezze, masse e baricentri dei *segmenti* normalizzati rispetto all'altezza e massa totale del corpo. In tabella 5.2 sono riportati i valori, in percentuale rispetto alla massa e altezza totale.

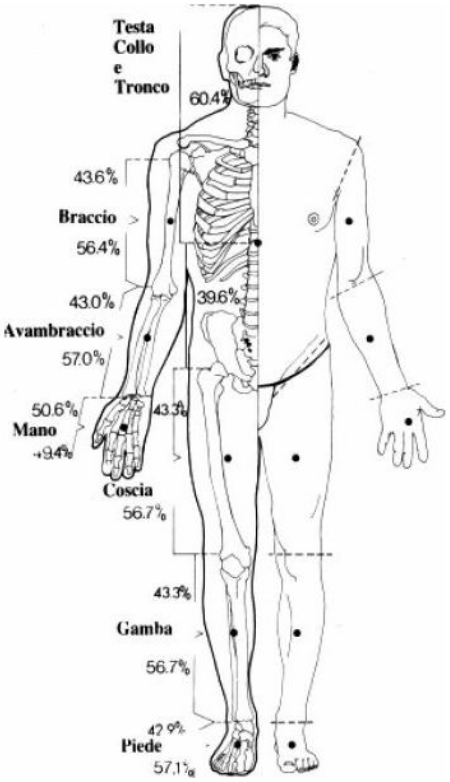


Figura 5.5: *Segmentazione secondo Dempster.*

Tenendo presente lo scopo del calcolo, cioè stimare la massa del pilota m_P da introdurre nella formula 5.3, si può considerare in prima approssimazione i segmenti *busto* e *testa+collo* mobili rispetto alla moto mentre gambe e braccia si considerano fermi. Quindi, dati peso e altezza del pilota, tramite i parametri di Dempster è possibile calcolare il la massa e la posizione del baricentro per il modello del pilota, ora dato solamente dalle parti mobili: *busto + testa+collo* dato che le parti che rimangono ferme rispetto alla moto non danno momento.
Le caratteristiche del pilota sono:

- $m_{tot} = 70\,kg$;
- $H_{tot} = 1,7\,m$.

Si ottiene quindi:

$$m_P = 41.3\,kg$$
$$pilot_{COM} = \big(0.011; 0.000; 0.306\big)\,[m].$$

dove $pilot_{COM}$ sono le coordinate del baricentro *(COM)* rispetto al bacino. La testa è stata considerata leggermente inclinata in avanti rispetto al piano frontale del tronco.

É necessario che le misure di posizione siano *riferite al baricentro del pilota*; il sistema di *pose-estimation* però,calcola la posizione del centro del target. Quindi, riferendoci allo schema di figura 5.6, è necessario effettuare la seguente trasformazione:

$$^{Tm}\mathbf{T}_{Tp} = {}^{Tm}[\mathbf{T}]_{Tt} \cdot T_{trasl}\big(D_X, D_Y, D_Z\big) \tag{5.15}$$

dove:

- $D_X = 0.021\,m$ rappresenta lo scostamento in direzione x del centro del target rispetto al piano sagittale del pilota ed è stato determinato estraendo dai test dei tratti di rettilineo nei quali il pilota era ben centrato e rimaneva fermo sulla motocicletta;

Tabella 5.2: Parametri di Dempster: massa, lunghezza e posizione dei segmenti.

Segmenti	$\frac{m_{link}}{m_{tot}}\%$	$\frac{L_{link}}{H_{tot}}\%$	$\frac{L_{COM}}{L_{link}}\%$
Mano	0.6	4.0	50.6
Avambraccio	1.6	15.4	43.0
Braccio	2.7	17.2	43.6
Testa e collo	7.9	17.7	56.7
Tronco	51.1	29.9	46.7
Coscia	9.7	24.7	43.3
Gamba	4.5	23.3	43.3
Piede	1.4	15.2	26.4

- $D_Y = zt - zb$, con:
  - $zt = \left(h_{tr} - h_t\right)$, $h_t = 0.24\,m$ (misurato durante i test);
  - $zb = z_{COM}$.
- $D_Z = 0.16 + x_{COM}$.

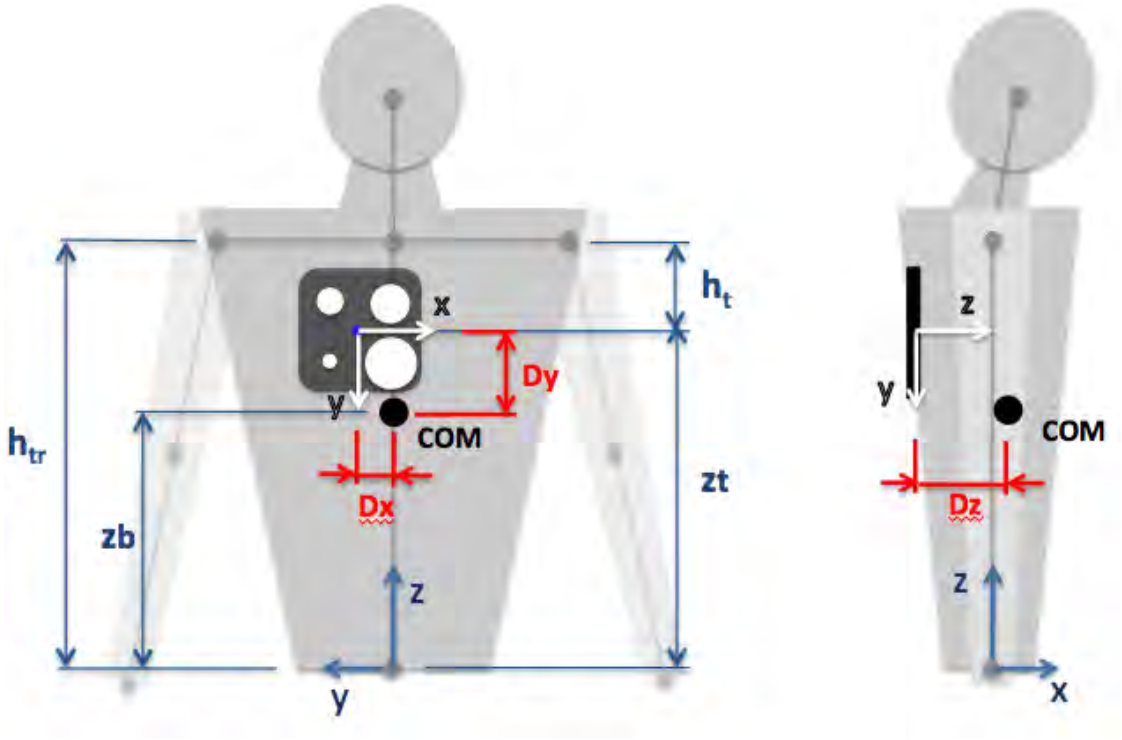


Figura 5.6: Modello del pilota.

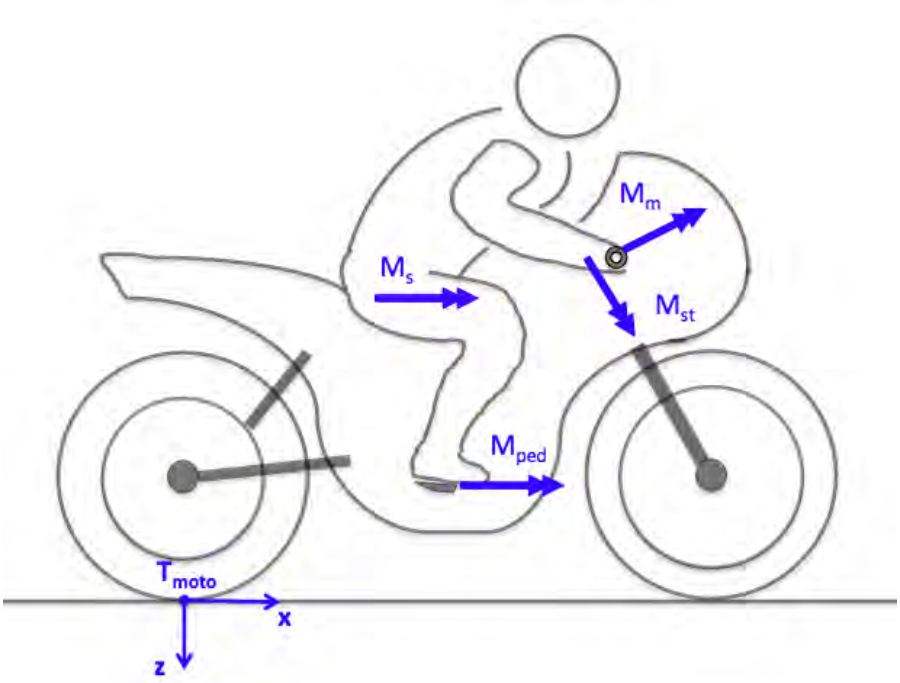


Figura 5.7: Coppie rilevate sulla motocicletta.

5.2 Calcolo della coppia misurata dai sensori

La coppia totale misurata dai sensori è data dalla somma dei contributi delle componenti in direzione x delle varie coppie (figura 5.7). Quindi:

$$M_{2D} = M_{ped} + M_s + M_m \cdot \cos\epsilon + M_{st} \cdot \sin\epsilon \tag{5.16}$$

dove:

- M_{ped} è la coppia applicata dal pilota sulle pedaline e vale:

$$M_{ped} = \left(F_{pedDX} - F_{pedSX}\right) \cdot \frac{d}{2}$$

  con d si è indicata la distanza fra le due pedaline;

- M_s è la coppia applicata sulla sella;
- M_m è la coppia applicata sul manubrio;
- M_{st} è la coppia allo sterzo;
- ϵ è l'angolo di inclinazione dello sterzo ($\epsilon = 25.5°$).

5.3 Risultati delle prove sperimentali

Le prove sono state effettuate in un tratto di strada pubblica che presenta delle ottime curve con una buona visibilità. Il tratto (lunghezza 1.6 km) è rappresentato in figura 5.8, le prove sono state suddivise in due parti[1]:

andata con senso di percorrenza da A a B (2 volte);

ritorno senso di percorrenza da B ad A (2 volte).

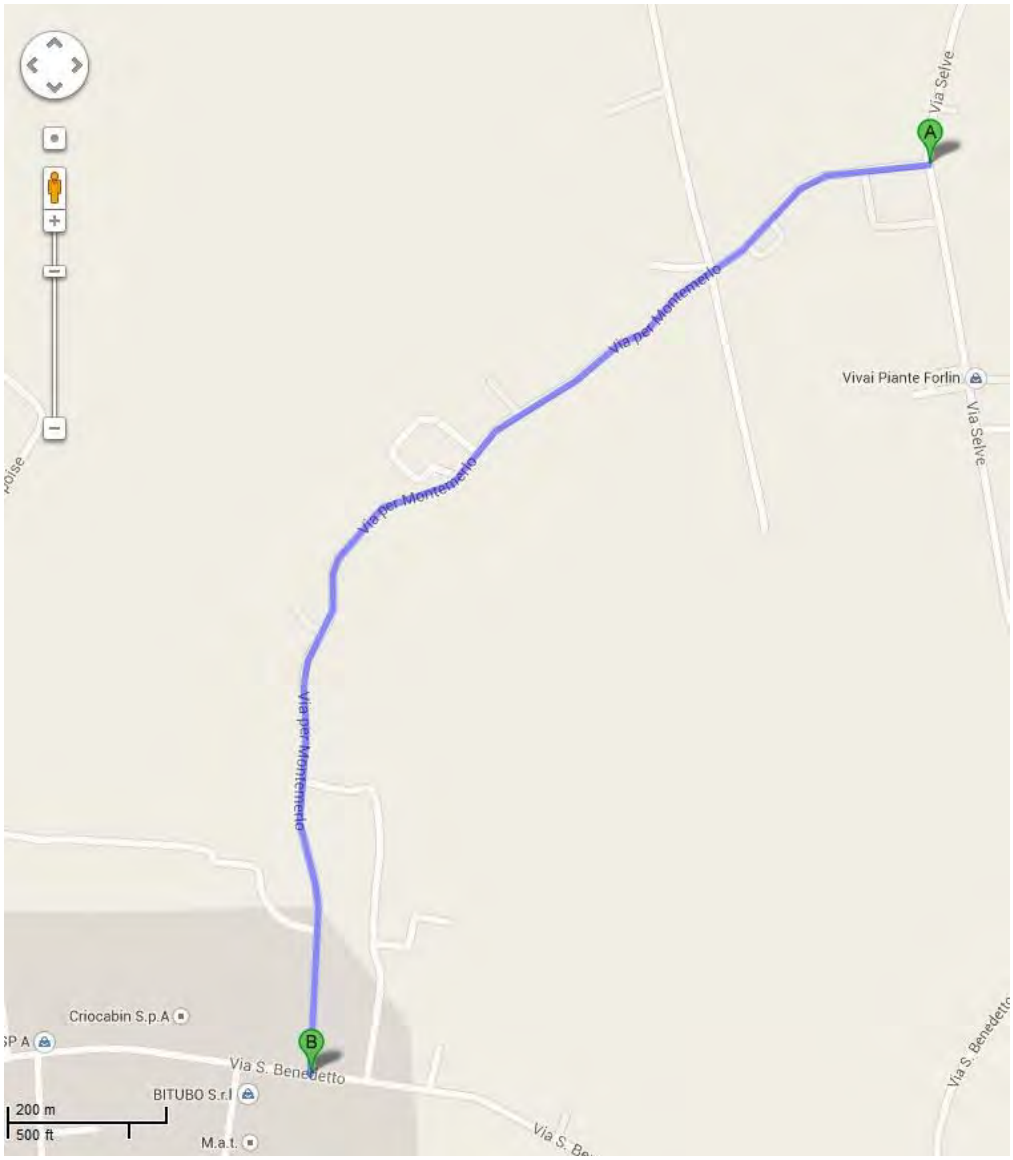


Figura 5.8: Tratto di strada percorso durante i test. *Fonte*: Google Maps.

I dati del sistema di *pose-estimation* e della telecamera sono stati sincronizzati tramite l'algoritmo di *cross-correlation* incrociando il canale della coppia sulla sella

[1]Entrambe le prove non sono riferite all'intero percorso di figura 5.8 ma ad un tratto più corto (1.2 km) a causa del traffico locale che ha ostacolato in parte la guida.

M_s con la coppia ottenuta dal sistema di pose estimation M_p. Questo perchè, in linea teorica, lo sfasamento fra queste due coppie ci si aspetta sia nullo dato che la coppia alla sella risente in modo diretto dello spostamento del pilota. Infine i dati sono stati filtrati a $2\,Hz$ tramite un filtro passa basso di ordine 10.

I risultati dei test sono riportati dalla figura 5.9 alla 5.12, con il seguente ordine:

andata 1 in figura 5.9

andata 2 in figura 5.10

ritorno 1 in figura 5.11

ritorno 2 in figura 5.12

In ogni figura è riportato rispettivamente:

1. un grafico che riporta l'andamento della velocità di avanzamento in $[km/h]$ e l'accelerazione longitudinale a_x in $[m/s^2]$, per il tratto percorso;

2. un grafico che riporta l'andamento dell'angolo di rollio φ in gradi e l'andamento dell'angolo α del pilota sempre in gradi, per il tratto percorso;

3. un grafico che riporta il confronto tra la coppia totale misurata dai sensori (eq. 5.16) indicata nei grafici con Coppia 2Dtot e la coppia misurata dal *pose-estimation* (eq. 5.3) indicata con Coppia Pilota, per il tratto percorso;

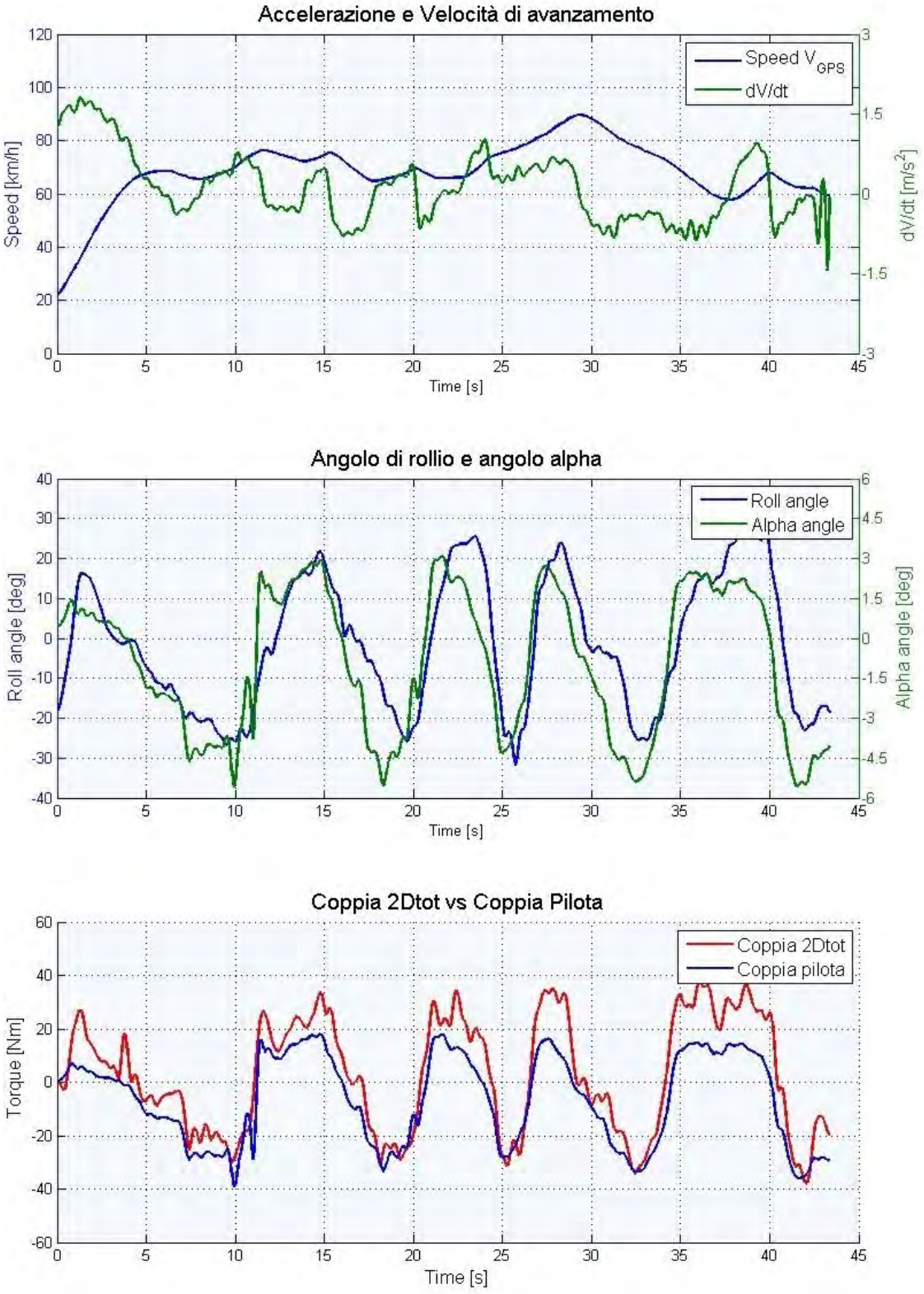


Figura 5.9: Andata 1

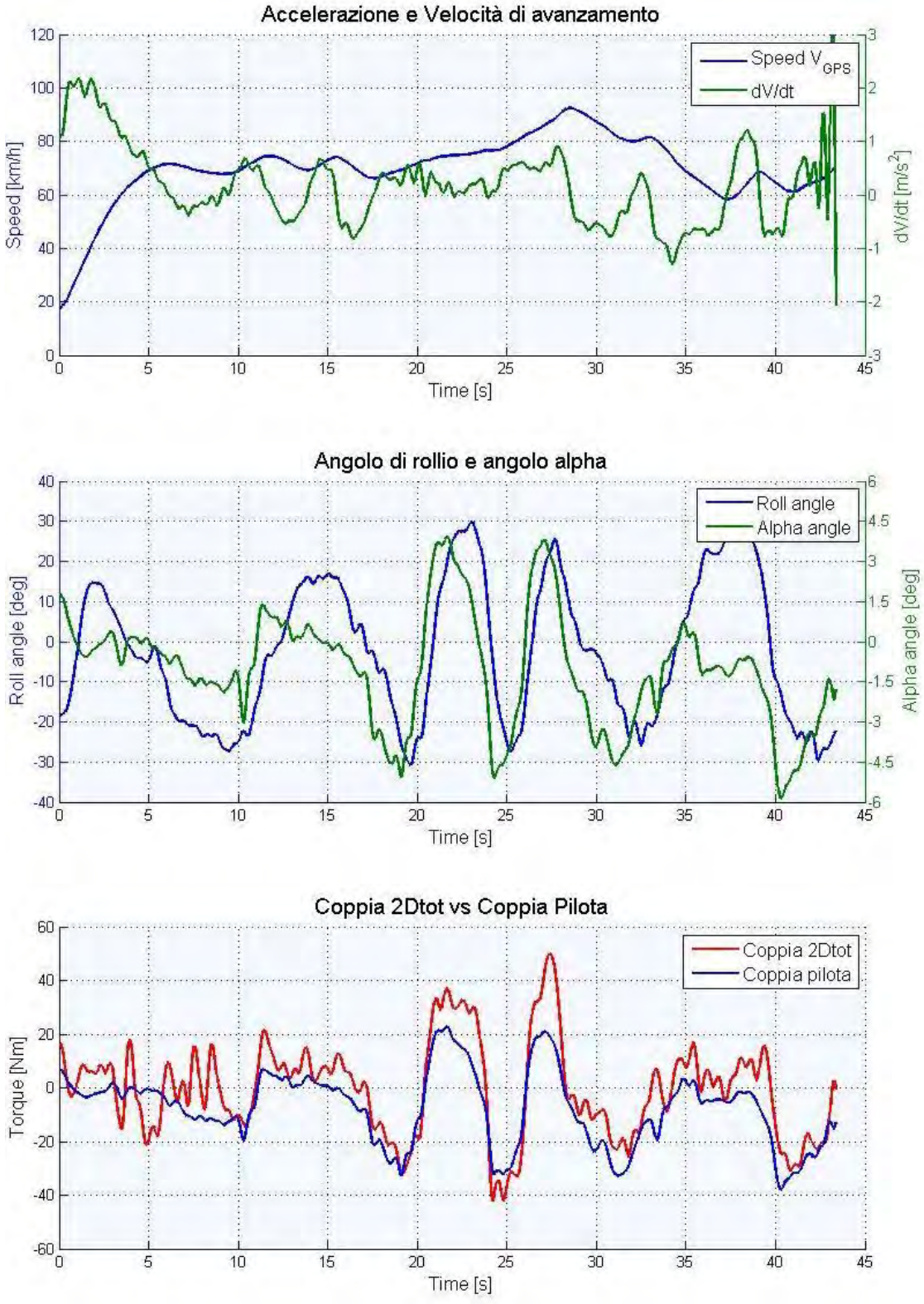


Figura 5.10: Andata 2

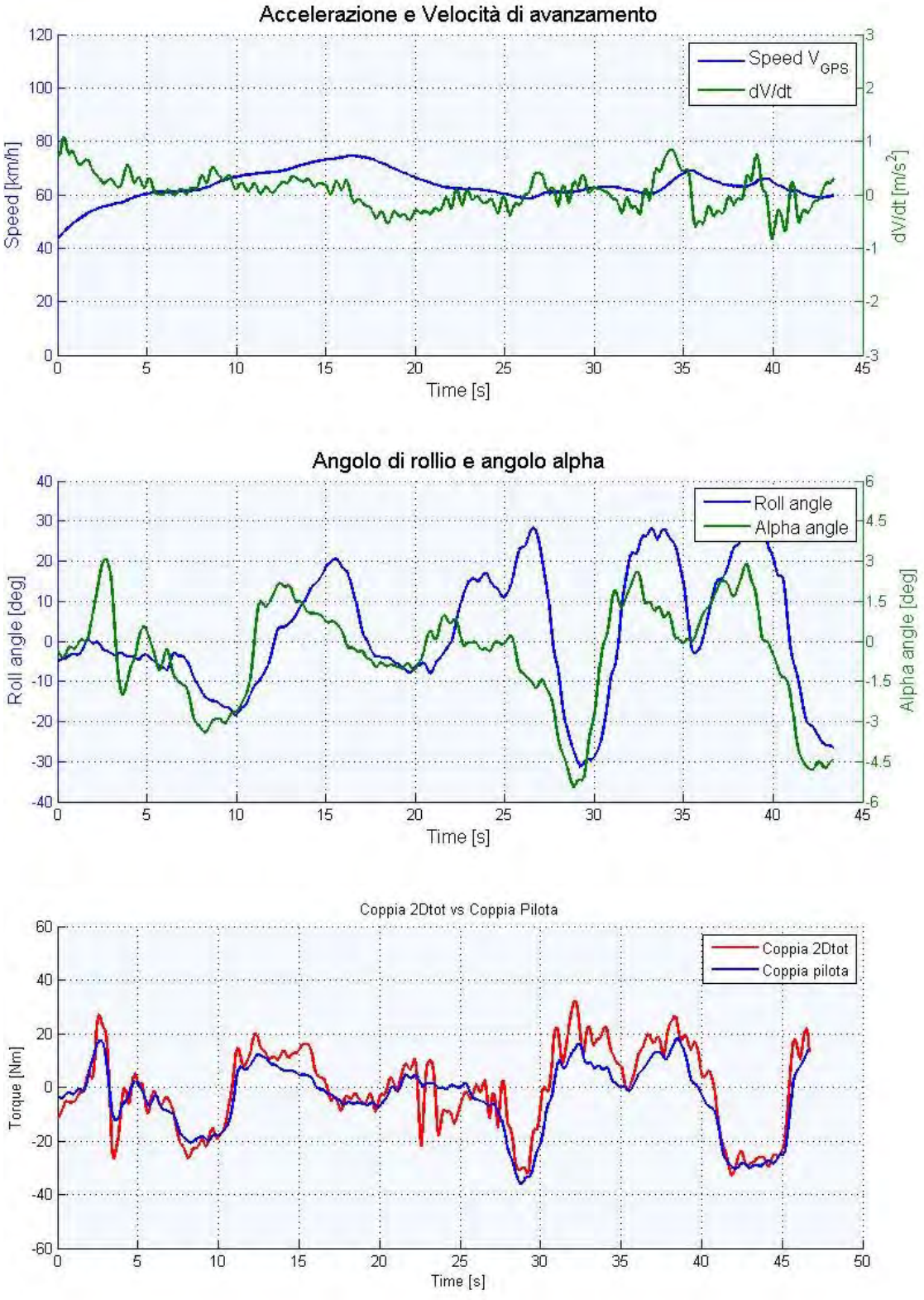


Figura 5.11: Ritorno 1

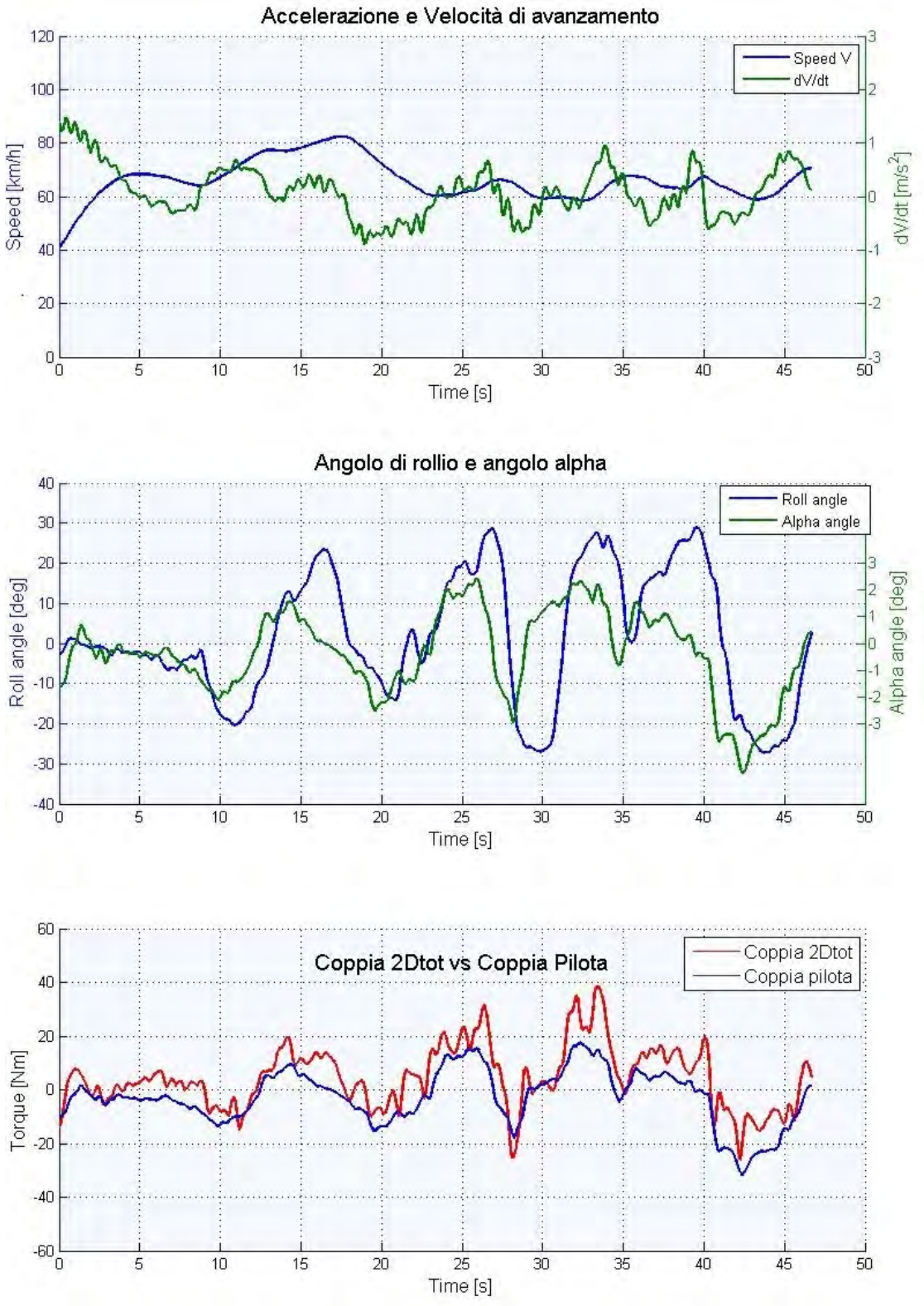


Figura 5.12: Ritorno 2

Tabella 5.3: Correlazione fra le coppie misurate con le due metodologie e correlazione fra φ e α.

	Cross Correlation Coefficients $\{C_P \, vs \, C_{2D}\}$	$\{\varphi \, vs \, \alpha\}$	*Lags* $\{\varphi \, vs \, \alpha\}$ [s]
Andata 1	0.842	0.886	0.53
Andata 2	0.802	0.718	0.73
Ritorno 1	0.893	0.759	0.90
Ritorno 2	0.694	0.792	1.16

Analisi dei risultati

Dai grafici si nota subito una buona correlazione fra l'andamento della *Coppia Pilota* e la coppia misurata dai sensori. In tabella 5.3 sono riportati i valori degli indici di correlazione, calcolati con l'algoritmo *Cross Correlation*.
I valori trovati fra la coppia calcolata con la 5.3 e la coppia ottenuta dai sensori (tabella 5.3) evidenziano che esiste un'ottima correlazione per le prove di *Andata1, Andata2* e *Ritorno1* mentre nel caso della prova *Ritorno2* la correlazione è leggermente più bassa.

Oltre ai valori di correlazione fra le coppie, in tabella 5.3 è riportato anche l'indice di correlazione e il ritardo (*lag*) fra l'angolo di rollio della motocicletta φ e l'angolo α del pilota.
I dati indicano che l'angolo di rollio e l'angolo α sono ben correlati in tutte quattro le prove inoltre l'angolo α risulta sempre in anticipo rispetto a ϕ. Questo dimostra come il pilota agisca spostando il proprio baricentro verso la direzione vuole che la moto rolli.
Quindi, se il pilota vuole far rollare la moto a destra, sposta il suo corpo verso destra. Questo che si traduce in una coppia di rollio che fa rollare la moto, e il ritardo dipenderà dall'entità della coppia applicata oltre che dall'inerzia della moto.

Osservando il valore del *lag*, fra le prove di *Andata* e di *Ritorno* si nota che quest'ultimo varia anche del 100% il che sta ad indicare che il pilota in ha guidato in modo nettamente diverso tra le due prove. Infatti dai grafici delle prove in *Andata*, dove il *lag* è inferiore rispetto al *Ritorno*, si può osservare che le coppie applicate sono mediamente più elevate rispetto ai tratti di ritorno il che significa che il pilota ha guidato con uno stile più aggressivo cercando di far rollare la moto rapidamente, di conseguenza ne risulta una coppia applicata più elevata e uno sfasamento basso tra φ e α.
Nel tratto di *Ritorno* il *lag* è doppio rispetto all'*andata* e anche la coppia è nettamente minore, il quindi in questo caso il pilota ha guidato con uno stile meno aggressivo, questo fatto si può apprezzare anche dall'entità dell'angolo α nettamente inferiore nel tratto di *ritorno* rispetto all'*andata*.

In tabella 5.4 sono riportati i coefficienti di correlazione tra l'angolo di rollio ottenuto confrontando le due prove di *andata* e le due prove di *ritorno*. Lo stesso si è fatto per l'angolo α. La correlazione fra l'andamento dell'angolo di rollio indica quanto *"simili"* sono le traiettorie seguite dalla moto (1 = traiettorie identiche),

Tabella 5.4: Confronto tra le prove ripetute.

	Cross Correlation Coefficient	
	Andata1 *vs* Andata2	Ritorno1 *vs* Ritorno2
φ	0,892	0,923
α	0,797	0,607

Tabella 5.5: Correlazione fra le coppie misurate 2.

	Cross Correlation Coefficients		
	C_P *vs* C_{2Dtot}	C_P *vs* $C_{2Dnosteer}$	C_P *vs* C_{sella}
Andata 1	0.842	0.895	0.970
Andata 2	0.802	0.817	0.952
Ritorno 1	0.893	0.840	0.961
Ritorno 2	0.694	0.787	0.930

mentre la correlazione fra l'andamento dell'angolo α indica quanto differente è stato lo stile di guida fra una prova e l'altra. Come si può notare l'angolo di rollio è ben correlato, mentre vi è una scarsa correlazione nell'angolo α nel caso delle prove di *ritorno*: in questo caso quindi il pilota ha adottato due stili diversi tra la prova *Ritorno1* e *Ritorno2*.

Nei grafici che seguono (da figura 5.13 a figura5.16) sono riportati gli andamenti delle coppie misurate dai sensori (prese singolarmente) per le prove effettuate.

Come si può notare la coppia alla sella e la coppia alla pedane sono le componenti che influiscono maggiormente sulla coppia totale.

La coppia allo sterzo si mantiene sempre al di sotto dei 10 Nm in tutte e quattro le prove, e si nota che il suo andamento è molto simile tra le prove ripetute. Un contributo maggiore lo da invece la coppia al manubrio che in certi tratti raggiunge i 20 Nm.

É chiaro che il modello usato per valutare la coppia applicata dal pilota M_p considerando solo lo spostamento laterale del pilota, è molto semplificato e le componenti della coppia che sono applicate allo sterzo, che il pilota applica usando la forza delle braccia non sono conteggiate correttamente. Questo modello inoltre non tiene conto degli effetti dinamici che invece sono presenti durante la guida.

Correlando infatti la coppia M_p con la coppia data dai sensori senza le componenti allo sterzo si ottiene una correlazione mediamente più elevata rispetto ai dati ottenuti precedentemente. Mentre correlando la coppia M_p con la sola coppia misurata alla sella si ottiengono coefficienti di correlazione elevatissimi, prossimi all'unità. Questo era abbastanza prevedibile in quanto lo spostamento laterale del pilota si trasmette perfettamente alla sella, mentre per quanto riguarda le azioni sul manubrio e in percentuale minore sulle pedane, la dinamica è più complessa.

In tabella 5.5 sono riportati i coefficienti di correlazione tra la coppia M_p e la coppia senza le componenti sullo sterzo e tra M_p e la coppia alla sella.

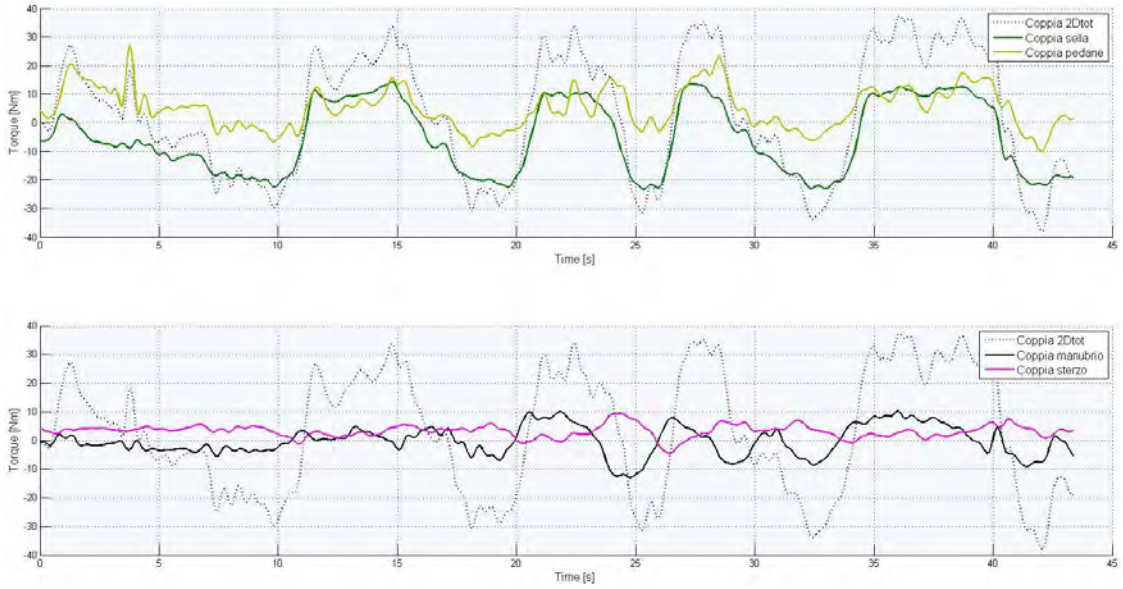


Figura 5.13: Andata1: coppie rilevate dai sensori.

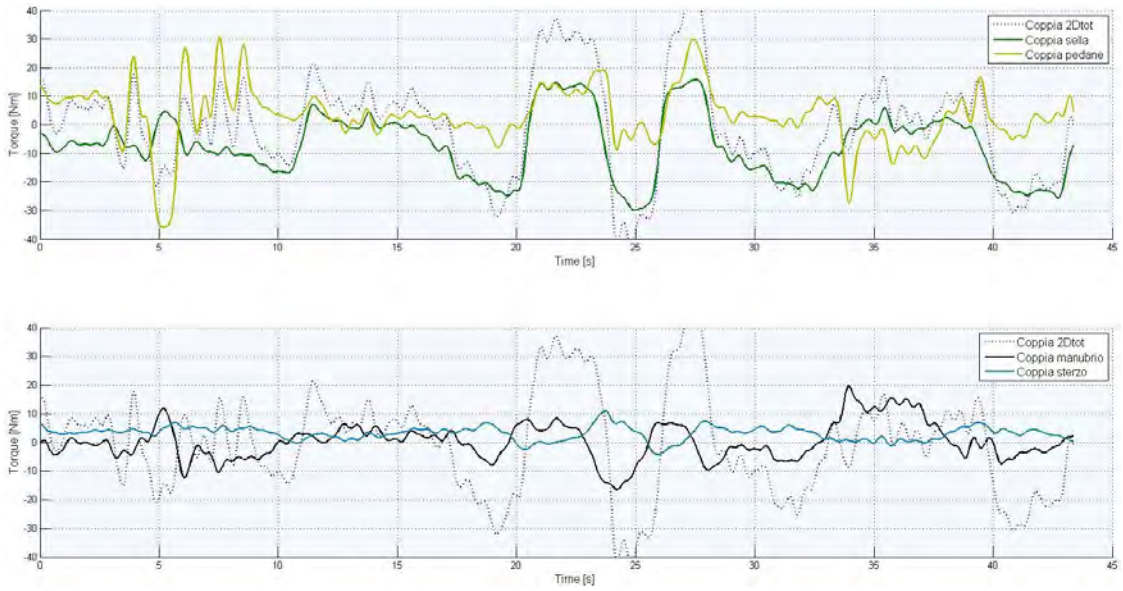


Figura 5.14: Andata2: coppie rilevate dai sensori.

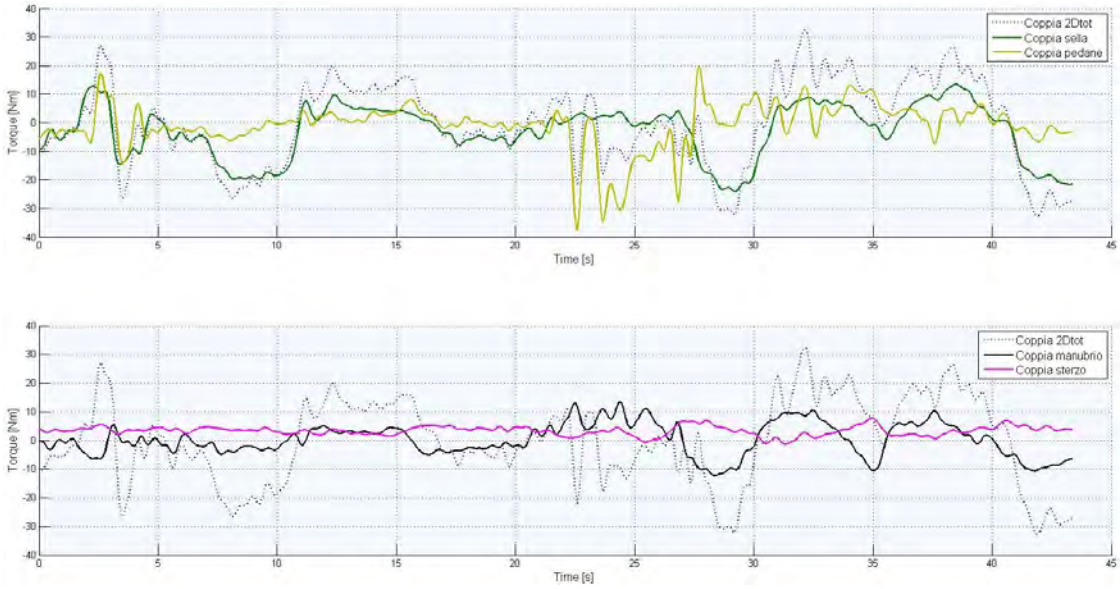


Figura 5.15: Ritorno1: coppie rilevate dai sensori.

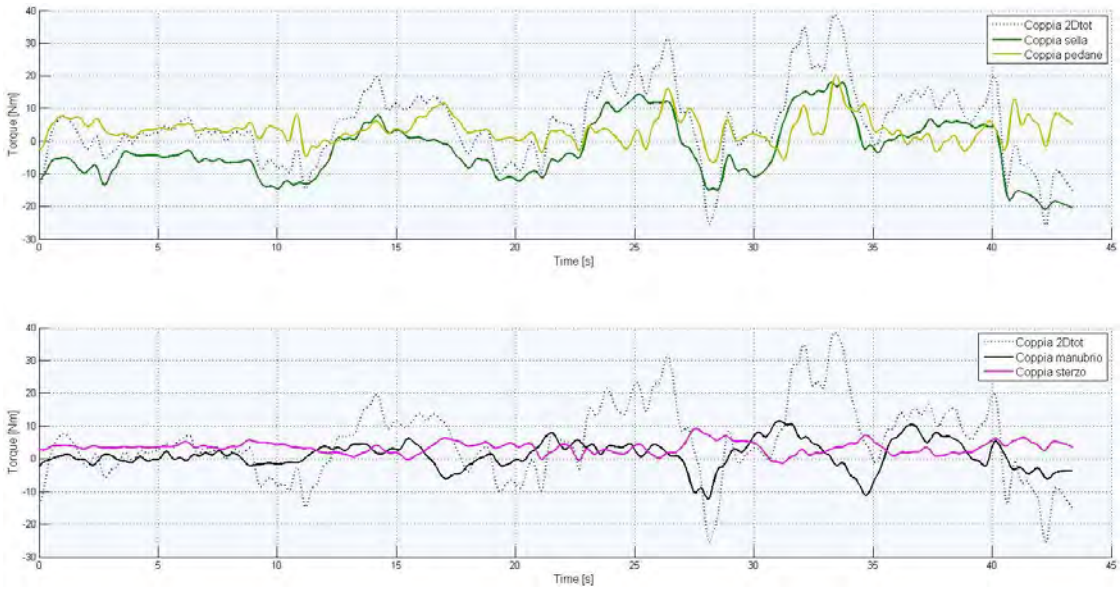


Figura 5.16: Ritorno2: coppie rilevate dai sensori.

Conclusioni

In questo lavoro è stato realizzato un metodo per la stima della posizione del pilota durante la guida di una motocicletta. Il metodo è basato sull'algoritmo di pose estimation che permette di ottenere la posizione e l'orientamento di un oggetto di dimensioni note rispetto a una camera calibrata. L'algoritomo richiede l'individuazione di almeno quattro punti dell'oggetto di cui si vuole conoscere la posizione. Questo oggetto è rappresentato da un target che sarà posizionato sulla schiena del pilota. É stato quindi scritto un codice in linguaggio Matlab che effettua l'analisi del filmato, individua i punti del target e stima la sua posizione. Questo codice è stato ottimizzato affinchè i tempi di analisi siano i più ridotti possibile. É raggiunto un risultato di $2,5$ minuti di analisi per minuto di filmato.

Successivamente è stata effettuata una procedura di calibrazione nella quale sono emersi dei buoni risultati in termini di errore di posizione. Questi risultati sono dovuti principalmente al fatto che il metodo utilizza un target fisso tramite il quale è possibile svincolarsi dal sistema di riferimento della telecamera della quale non è noto l'orientamento.

Infine il metodo è stato utilizzato in alcune prove sperimentali dove è stato messo a punto un modello che traduce lo spostamento laterale del pilota rispetto alla moto in coppia applicata alla moto stessa ed è emersa una discreta correlazione tra la coppia misurata dai sensori e la coppia ottenuta dal modello.

In conclusione a questo lavoro è emerso che il metodo di *pose-estimation* messo a punto ha superato ogni aspettativa iniziale per quanto riguarda l'accuratezza della misura, in quanto dalla calibrazione è emerso un valore dell'errore medio al di sotto del grado per quanto riguarda le rotazioni e sotto i due millimetri per quanto riguarda le traslazioni. Inoltre il codice realizzato elabora i filmati con tempi molto ristretti: questo ha permesso di analizzare i test in un tempo molto breve.

Il modello utilizzato per la stima della coppia di rollio applicata alla motocicletta è molto semplificato, non sono state considerate infatti le rotazioni del busto del pilota. Il modello che comprende tutti e sei i *gld* del busto del pilota è molto complesso e richiede uno studio approfondito.

Appendice A

Codice Matlab

A.1 Video Processing

```
% POSE-ESTIMATION OF A MOTORCYCLE RIDER USING A CAMERA
% Video processing:
%                - loading video;
%                - blob analysis target-pilot;
%                - blob analysis target-moto.
%
% A. BRAI 2014
% mail to: alberto.brai85@libero.it

clearvars %-except video
close all
disp('                                    ')
disp('VIDEO PROCESSING:');
disp('          - loading video;');
disp('          - blob analysis of a rider target;');
disp('          - blob analysis of a moto target.');
disp('  ');


%% LOADING VIDEO

videoName = input('Insert video name with suffix (.avi,...
                         .mp4): ','s');
video = VideoReader(videoName)

% video parameter

[start_frame, end_frame, n_frames, skip_frames] = ...
             videoParameter(video);
```

```
%% TARGET PILOT:
% Crop images at target size

disp('---> IDENTIFY THE RIDER TARGET...');
disp('  ');
[ crop win ] = click_and_crop(read(video, start_frame));

disp('OK...');
disp('Running blob analysis...');


% TARGET PILOT: Blob analysis

shape = [0.20 1];
area = [200 20000];
wbar = waitbar(0,'Processing video...');
    i = 1; f = 0; centers_old = 0;

       for k = start_frame : skip_frames : end_frame
            [centers(:,:,i) newcrop win shape area f k] = ...
                  blobs_00(read(video, k), shape, area, crop, ...
                        win, f, centers_old, k, i);
            centers_old = centers(:,:,i);
            crop = newcrop;
            waitbar(i / n_frames, wbar, sprintf('Processing frame ...
                  %i / %i ...', i , n_frames))
            i = i+1;
        end

% correct camera lens distortion

var01 = 0;
var01 = input('Do you want to correct lens distortion? ( [ ]=Yes, ...
        other=No) ','s');

if isempty(var01) == 1;
    [centers_und] = undistortPixel(centers);
    centers = centers_und;
end

close(wbar)
save('centers_pilot.mat', 'centers')
disp('  ');
disp('DONE!');
disp('  ');
disp('The coordinates of the centers for each frame of Rider  ...
          Target were saved on the current folder, on file: ...
```

```
                  "centers_pilot.mat"');
disp('  ');
disp('...');
disp('  ');


%% TARGET MOTO

var2 = 0;
var3 = 0;
var2 = input('Do you want to analyze even the target moto (fixed ...
          target)? ( [ ]=Yes, other=No) ','s');

if isempty(var2) == 1;
     var3 = input('The Target Moto it is on the same video of Rider ...
             Target? ( [ ]=Yes, other=No) ','s');

      if isempty(var3) == 0;
          videoName = input('Insert video name with suffix (.avi, ...
             .mp4):  ','s');
          video = VideoReader(videoName)
          [start_frame, end_frame, n_frames, skip_frames] = ...
                 videoParameter(video)
      end

    % Crop images at target size

    disp('---> IDENTIFY THE TARGET-MOTO...');
    disp('  ');
    [ crop win ] = click_and_crop(read(video, start_frame));
    disp('OK...');
    disp('Running blob analysis...');

    % Blob analysis

    n_frames = 50;
    end_frameM = start_frame + n_frames - 1;

       if end_frameM >= video.NumberOfFrames
               end_frameM = video.NumberOfFrames
               n_frames = (end_frameM-start_frame)
       end

    shape = [0.6 1];
    area = [150 20000];
    wbar = waitbar(0,'Processing video...');
        i = 1; f = 0;
```

```
            for k = start_frame : end_frameM
                [centers_moto(:,:,i) f win crop i] = ...
                    blobs_01(read(video, k), shape, area, crop, ...
                           win, f, video, k, i);
                waitbar(i / n_frames, wbar, sprintf('Processing frame ...
                    %i / %i ...',i , n_frames))
              i = i+1;
            end


   % correct camera lens distortion

   var01 = 0;
   var01 = input('Do you want to correct lens distortion? ...
           ( [ ]=Yes, other=No) ','s');
   if isempty(var01) == 1;
  [centers_und] = undistortPixel(centers_moto);
   centers_moto = centers_und;
   end


   close(wbar)
   save('centers_moto.mat', 'centers_moto')
   disp('  ');
   disp('DONE!');
   disp('  ');
   disp('The coordinates of the centers of each frame were saved ...
           on the current folder, on file: ');
   disp('                  - "centers_pilot.mat" for Rider Target');
   disp('                  - "centers_moto.mat" for Moto Target');
   disp('  ');
   disp('Complete! :)');

else
  disp('Complete! :)');
end

%% message

disp('  ');
disp('For correct any ambiguity problems, that it is possible to ...
     occurred during the blob analysis, running: ...
        "ambiguity_control.m"');
disp('   ');
disp('Instead, for pose estimation, running: ');
disp('            - "pose_est_pilot.m" for rider pose estimation;');
disp('            - "pose est_moto.m"  for moto pose estimation;');
```

```
disp('  ');
disp('Bye!');
disp('A. Brai 2014.');
```

A.2 pose_est_moto

```
% POSE-ESTIMATION OF A MOTORCYCLE RIDER USING A CAMERA
% pose estimation of a target moto
%
% A. BRAI 2014
% mail to: alberto.brai85@libero.it


% Loading parameters

 cam = gopro(); % load GoPro Hero 1 parameter from camera calibration
% cam = goproH3_1440p();
 load('centers_moto.mat'); % load centers from blob analysis


%% Pose estimation:

clear T
n_frames = size(centers_moto,3);
T   = zeros(4, 4, n_frames);
T_old = T(:,:,1); % memorize the previous pose matrix

for k = 1:n_frames
      % attempt to estimate target's pose
      try T(:,:,k) = cam.estpose(target_moto, centers_moto(:,:,k));
      catch  % on pose estimation failure...
          disp(['Warning: pose estimation failed at ', num2str(k)]);
          T(:,:,k) = T_old; % set the matrix equal to the previous
      end
      T_old = T(:,:,k); % memorize the previous pose matrix
end

T_moto = median(T, 3); % calcola la posizione media


% save data
save('T_moto.mat', 'T_moto')
```

A.3 pose_est_pilot

```
% POSE-ESTIMATION OF A MOTORCYCLE RIDER USING A CAMERA
```

```
% pose estimation of a target pilot
%
% A. BRAI 2014
% mail to: alberto.brai85@libero.it


% Loading parameters
 cam = gopro(); % load GoPro Hero 1 parameter from camera calibration
% cam = goproH3_1440p();
 load('centers_pilot.mat'); % load centers from blob analysis


%% Pose estimation:

 n_frames = size(centers,3);
 T_pilot   = zeros(4, 4, n_frames);
T_old = T_pilot(:,:,1); % memorize the previous pose matrix

for k = 1:n_frames
      % attempt to estimate target's pose
      try T_pilot(:,:,k) = cam.estpose(target_pilot, centers(:,:,k));
      catch  % on pose estimation failure...
          disp(['Warning: pose estimation failed at ', num2str(k)]);
          T_pilot(:,:,k) = T_old; % set the matrix equal to the previous
      end
      T_old = T_pilot(:,:,k); % memorize the previous pose matrix
end


% save the variable
save('T_pilot.mat', 'T_pilot')
```

A.4 rel_pos_TmTp

```
% POSE-ESTIMATION OF A MOTORCYCLE RIDER USING A CAMERA
% Relative position TARGET-MOTO - TARGET-PILOT
%
% A. BRAI 2014
% mail to: alberto.brai85@libero.it

dim = size(T_pilot,3);

for k = 1:dim
   T_motopilot(:,:,k) = inv(T_moto)*T_pilot(:,:,k);
end
```

Bibliografia

[1] Jean-Yves Bouguet, *Camera Calibration Toolbox for Matlab*, http://www.vision.caltech.edu/bouguetj/calib_doc.

[2] Nicola Petrone (2013), Dispense del corso *Sport engineering and rehabilitation devices*.

[3] Peter Corke (2011), *Robotics, Vision and Control*, Springler.

[4] Peter Corke (2012), *Machine Vision Toolbox*.

[5] Vianello Alberto (2011), *Studio teorico e Sperimentale per la determinazione dell'angolo di rollio di una motocicletta in curva*.

[6] V. Lepetit, F.M. Noguer, P. Fua (2009), *EPnP: An Accurate O(n) Solution to the PnP Problem*.

[7] Vittore Cossalter (2008), *Motorcycle Dynamics*.