UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

LAUREA MAGISTRALE IN INGEGNERIA AEROSPAZIALE

VISIONE STEREOSCOPICA PER OPERAZIONI DI PROSSIMITÀ TRA SATELLITI

Stereoscopic vision for spacecraft proximity operations

Relatore: Prof. Enrico Lorenzini
Correlatore: Ing. Mattia Mazzucato

Laureando: ALBERTO GUZZO
Matricola: 1080630

Anno Accademico 2016-2017

Ai miei genitori, le mie solide fondamenta.

Indice

Elenco delle figure

Elenco delle tabelle

Abstract

The "SPARTANS" project is a simulation environment for space activities developed in the CISAS (Centro Interdipartimentale Studi ed Attività Spaziali) laboratories of the University of Padova with the aim of becoming a reference center for the study of satellite formation flight. It is currently composed of two satellite units with three rotational degrees of freedom and free to translate on a low-friction plane.
The aim of this thesis is to develop a vision system based on the output of a stereocamera, to be integrated into the attitude module of a micro satellite and able to determine the position and the attitude in terms of Euler's angles, of another cooperating micro satellite.
A set of fiducial markers is created to be applied to a target satellite and a suitable software is developed to recognize the markers in the stereo images. Through the mathematical modeling of the stereocamera it is calculated the spatial positions of the markers in a camera reference, and by these, knowing the position of the markers in the target reference, are estimated position and attitude.
Virtual and laboratory tests are carried out obtaining interesting results both in terms of precision and computational speed and the software is finally validated by means of a Montecarlo analysis.

Introduzione

Ci si trova in un contesto storico dove lo sviluppo dell'attività spaziale è entrato con vigore ormai in molteplici ambiti siano essi scientifici, industriali, militari o commerciali e la sua espansione non rallenta, grazie anche ai reparti di ricerca ed innovazione di ognuno di questi settori. In concomitanza con la nascita di nuove tecnologie ed un uso sempre più frequente del settore aerospaziale, ci si trova però ad aver necessità di abbaterne i costi, poichè aumentarne l'uso significa imbattersi in sempre più situazioni che richiedono un dispendio di energie e finanze. Fra queste i possibili guasti o i danneggiamenti esterni ad opera ad esempio di un sempre più sovraintasato ambiente orbitale terrestre.
Questo porta a fare considerazioni su due ambiti in ascesa:

1. la ricerca di soluzioni costruttive e compensative differenti;
2. l'affinamento dello studio e della pianificazione di strategie e manovre di controllo.

Per il primo ambito una delle soluzioni di sempre maggior interesse è il volo in formazione di satelliti cooperanti e non, in fase di studio e sviluppo poichè presentano soluzioni costruttive di dimensioni ridotte che implicano costi ridotti, tempi di produzione più brevi, la possibilità di sostituire componenti in modo più semplice e non da meno la possibilità di reciproco controllo tra satelliti di una stessa formazione.

Strettamente legato al primo è il secondo ambito. Si fa più importante poter studiare con minuzia le vare fasi di controllo in una possibile missione spaziale o esaminare tecnologie e strategie da utilizzare in differenti manovre spaziali come ad esempio randevouz autonomi, docking o semplici ispezioni orbitali di controllo. Per fare questo in modo efficace, efficiente e a budget ridotti, numerosi istituzioni e centri di ricerca stanno sviluppando ambienti per simulazioni in microgravità in cui è possibile testare in sicurezza, in modo controllato ed abbattendo i costi, le diverse situazioni d'interesse spaziale. Tra questi si posso citare l'*experimental free flying platform*

"PINOCCHIO" ad opera dell'Università di Roma La Sapienza [1], il *Synchronized Position Hold, Engage, and Reorient Experimental Satellites "SPHERES"* ad opera dei ricercatori del MIT[2] o il *Formation Control Testbed "FCT"* sviluppato da JPL-Caltech[3].

In questo scenario, nasce nei laboratori del CISAS (Centro Interdipartimentale Studi ed Attività Spaziali) e del DII (Dipartimento di Ingegneria Industriale) dell'Università degli Studi di Padova il progetto "SPARTANS", sviluppato dal Professor Enrico Lorenzini in collaborazione con un'équipe di studenti e dottorandi. Il progetto ha l'intento di creare un centro di riferimento per lo studio di strategie e algoritmi di controllo per il volo in formazione. Nella pratica l'ambiente di simulazione è attualmente composto da due unità indipendenti, di cui una in fase di completamento, che simulano due micro satelliti in formazione e le quali sono dotate di cinque degrees of freedom (DOF) di cui tre rotazionali e due traslazionali, le unità sono infatti libere di traslare su un piano a basso attrito. Ognuna di esse è dotata di un proprio modulo d'assetto e di un equipaggiamento interno tra cui un'ADCS (Attitude Determination and Control Subsystem), una IMU (Inertial Measurement Unit) ed un sistema propulsivo.

Obbiettivi della tesi

Il contributo che questo lavoro di tesi si offre di portare al progetto "SPARTANS" è quello di implementare un sistema di visione basato sull'output di una stereocamera, da integrare nell'equipaggiamento interno di un'unità satellitare e che permetta al modulo d'assetto della stessa, di stimare la posa e l'assetto relativi di un'altra unità in formazione. Nella pratica questo si traduce in:

- progettare e produrre un sistema di markers da applicare su un possibile satellite *target*;
- sviluppare un software che rielaborando le immagini provenienti dalla stereocamera individui tali markers;
- sviluppare un software che riesca a stimare la posizione spaziale dei markers rispetto ad un riferimento scelto;

- sviluppare infine un software che tramite la posizione dei markers stimi la posa e l'assetto relativi del sistema di riferimento scelto per il possibile target rispetto al sistema di riferimento scelto per la stereocamera.

La trattazione si sviluppa allora in un primo capitolo dove vengono descritte la basi teoriche e i modelli matematici su cui si basa l'intero processo di stima. Viene poi analizzato nel capitolo secondo il set up sperimentale tra cui lo schema costruttivo di un'unità satellitare ed i markers pensati per questo lavoro. Si passa poi ad esporre nel terzo capitolo com'è stato pensato e sviluppato il software in ogni sua componente e si presentao infine nel quarto ed ultimo capitolo i risultati ottenuti in due fasi di test del software sviluppato, una prima in ambiente virtuale e in seguito sull'effettivo simulatore nei laboratori dell'Università di Padova.

Capitolo 1

Background Teorico

Per comprendere le fondamenta alla base del lavoro svolto e le motivazioni dietro a determinate scelte progettuali, è necessario esplicitare gli strumenti di cui è stato fatto uso. Questo primo capito presenta infatti i modelli matematici e le basi teoriche utilizzate nello sviluppo del sistema di visione e del processo per la stima di posa e assetto. In particolare:

1. Essendo lo scopo ultimo di questo lavoro di tesi per l'appunto la stima d'assesso, la prima parte del lavoro tratta alcuni basilari concetti di cinematica e determinazione dei parametri d'assetto, utili anche per la trattazione degli argomenti successivi;

2. Viene poi esplicitato e descritto per passi sequenziali cos'è e come viene modellizzato un sistema di visione;

3. Si passa successivamente a spiegare come viene sfruttato il sistema di visione e quindi il concetto e l'utilità dei *fiducial markers*;

4. Si conclude il capitolo spiegando l'algoritmo scelto alla base del riconoscimento dei markers stessi.

Il tutto viene successivamente accorpato nel *capitolo 3* dove viene illustrato com'è stato sviluppato il software e come sono state usate tali informazioni nel farlo.

1.1 Richiami di cinematica e calcolo dell'assetto

In questo paragrafo si rivedono alcune nozioni di base della cinematica relativa e il concetto di assetto. Con una prima definizione, determinare posa e assetto significa stimare la posizione e l'orientamento di un determinato sistema di riferimento rispetto ad un altro prefissato, cercando di calcolare la trasformazione che istante per istante farebbe combaciare i due sistemi. Riprendendo alcuni semplici concetti [4], dati due sistemi di riferimento ortonormali (*Figura 1.1*), il moto rigido con il quale possiamo idealmente sovrapporre una terna sull'altra è dato da due componenti, una traslazione ed una rotazione.

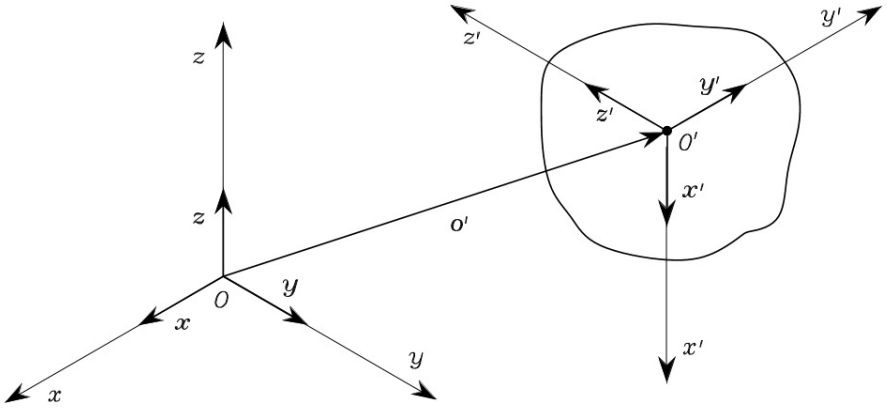


Figura 1.1: Relazione sistemi di riferimento

Un primo moto è atto a far combaciare i centri delle due terne, annullando la posizione relativa di una rispetto all'altra. Se consideriamo come terna principale O, la terna O' si trova in posizione (O'_x, O'_y, O'_z) rispetto ad O (*Figura 1.1*) e quindi si ha un primo moto di traslazione dato dal vettore

$$T = [-O'_x, -O'_y, -O'_z] \tag{1.1}$$

detti poi $\mathbf{x}, \mathbf{y}, \mathbf{z}$ i versori della terna principale, i versori di O' hanno componenti

$$\mathbf{x}' = x'_x\mathbf{x} + x'_y\mathbf{y} + x'_z\mathbf{z} \tag{1.2}$$

$$\mathbf{y}' = y'_x\mathbf{x} + y'_y\mathbf{y} + y'_z\mathbf{z} \tag{1.3}$$

$$\mathbf{z}' = z'_x\mathbf{x} + z'_y\mathbf{y} + z'_z\mathbf{z} \tag{1.4}$$

rispetto alla terna di riferimento O, e tali componenti sono anche i termini di una matrice $R \in [3\times3]$ detta *matrice di rotazione*. Detta matrice contiene le informazioni

necessarie ad annullare la rotazione angolare tra le due terne ed è quindi il secondo moto rigido dopo quello di traslazione dell'origine della terna. La matrice R gode di tre importanti proprietà:

1. Detta R^{-1} la matrice di rotazione inversa, ovvero la matrice che contiene le componenti dei i versori della terna O rispetto alla terna O', vale

$$R^T = R^{-1} \tag{1.5}$$

2. Essendo una matrice ortogonale (conseguenza logica del fatto che le sue componenti sono i versori di una terna ortonormale), vale

$$RR^T = I \tag{1.6}$$

dove I è la matrice identità. Vale quindi anche

$$RR^{-1} = I \tag{1.7}$$

3. Vale sempre $det(R) = 1$

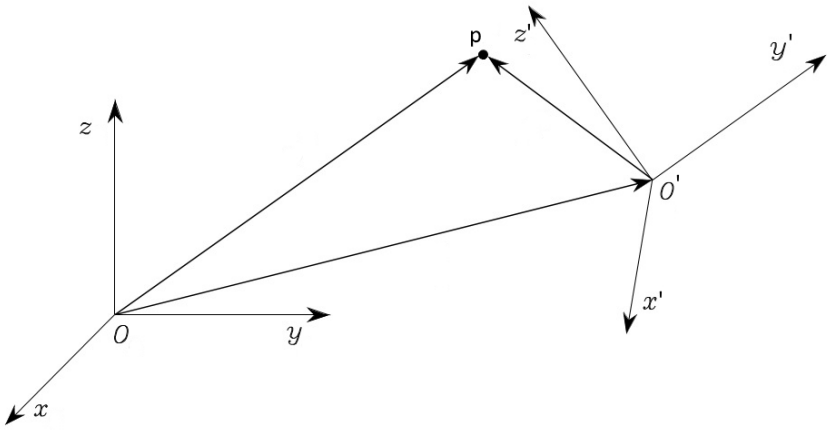


Figura 1.2: Il punto p nei due sistemi di riferimento

La trasformazione composta dalla traslazione del vettore T e dalla rotazione tramite la matrice R permette inoltre di poter esprimere un generico punto p rispetto al riferimento O conoscendo le sue coordinate rispetto ad O', e viceversa. In particolare, se indichiamo con $\{\mathbf{X}\} = [x, y, z]$ le coordinate di p rispetto ad O e con $\{\mathbf{X}'\} = [x', y', z']$ quelle rispetto ad O' (*Figura 1.2*), vale

$$\{\mathbf{X}\} = [R]\{\mathbf{X}'\} + \{T\} \tag{1.8}$$

o viceversa,

$$\{\mathbf{X}'\} = [R^{-1}]\{\mathbf{X}\} - \{T\} \tag{1.9}$$

oppure, indicando con T^{-1} il vettore di traslazione inverso contenente le coordinate del centro della terna O rispetto alla terna O', coincidente con $-T$,

$$\{\mathbf{X}'\} = [R^{-1}]\{\mathbf{X}\} + \{T^{-1}\} \tag{1.10}$$

Torna infine utile riscrivere le equazioni 1.8 e 1.10 in una forma compatta attraverso l'uso di coordinate omogenee, ovvero accorpando l'intera trasformazione di rotazione e traslazione in un'unica matrice $\mathbf{T} \in [4 \times 4]$ detta matrice di trasformazione. Si ottengono

$$\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = [\mathbf{T}] \begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} \tag{1.11}$$

$$\begin{bmatrix} \mathbf{X}' \\ 1 \end{bmatrix} = \left[\mathbf{T}^{-1}\right] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} R^{-1} & T^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \tag{1.12}$$

Ricapitolando, la matrice di rotazione R:

- fornisce l'orientamento di una terna rispetto ad un'altra, in particolare i suoi vettori colonna sono i coseni direttori degli assi della terna ruotata rispetto alla terna di partenza;
- rappresenta assieme al vettore di traslazione T una trasformazione di coordinate che mette in relazione le coordinate di uno stesso punto in due terne differenti;
- è l'operatore che consente di ruotare un vettore in una stessa terna di coordinate.

Esistono diversi modi per parametrizzare tale matrice. I due più utilizzati sono gli *angoli di Eulero* e i *quaternioni*.

1.1.1 Angoli di Eulero

La prima parametrizzazione è quella attraverso gli angoli di Eulero. Questa si basa sul fatto di poter vedere l'intero moto rotazionale necessario a far sovrapporre i versori delle due terne O e O', come frutto di una serie di rotazioni elementari attorno agli assi della terna O' in modo da farli combaciare con gli assi della terna principale O. Esistono tre rotazioni elementari.

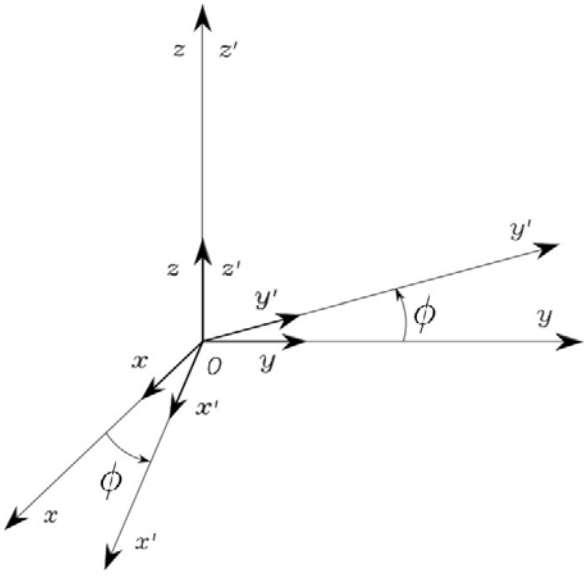


Figura 1.3: Rotazione attorno all'asse Z di un angolo ϕ

Se ad esempio si considera la rotazione attorno al solo asse z di un certo angolo ϕ come in *Figura 1.3*, la matrice di rotazione, introdotta come la matrice le cui componenti sono le componenti dei versori della terna ruotata rispetto alla terna presa come riferimento principale, risulta una matrice R_z tale che

$$R_z = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.13}$$

allo stesso modo una rotazione elementare attorno all'asse y di un certo angolo θ ha matrice di rotazione R_y pari a

$$R_y = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \tag{1.14}$$

ed una rotazione attorno ad x di un angolo ψ,

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & -sin\psi \\ 0 & sin\psi & cos\psi \end{bmatrix} \tag{1.15}$$

Esistono molteplci combinazioni delle tre rotazioni elementari usate in vari ambiti per ricostruire la matrice di rotazione finale, combinazioni date dalle diverse sequenze di rotazioni attorno agli assi, due tra le più comnui ad esempio sono ZYZ o

ZYX. Consideriamo in particolare la seconda di queste in quanto la più utilizzata in ambito aerospaziale, poichè da questa è possibile ricavare come parametri d'assetto gli angoli ψ, θ, ϕ intesi rispettivamente come angoli di *roll*,*pitch* e *yaw*, gli *angoli di Eulero*. La combinazione nell'ordine delle tre rotazioni elementari

$$R_z R_y R_x = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & -sin\psi \\ 0 & sin\psi & cos\psi \end{bmatrix} \tag{1.16}$$

da la matrice di rotazione

$$R = \begin{bmatrix} cos\phi cos\theta & sin\phi cos\theta & -sin\theta \\ cos\phi sin\theta sin\psi - sin\phi cos\psi & sin\phi sin\theta sin\psi + cos\phi cos\psi & cos\theta sin\psi \\ cos\phi sin\theta cos\psi + sin\phi sin\psi & sin\phi sin\theta cos\psi - cos\phi sin\psi & cos\theta cos\psi \end{bmatrix} \tag{1.17}$$

Volendo risolvere il problema inverso, ovvero data una generica matrice di rotazione R

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \tag{1.18}$$

da questa estrarre gli angoli di Eulero relativi ad una stessa combinazione ZYX, si possono calcolare ψ, θ, ϕ tramite

$$\phi = arctan\left(\frac{r_{1,2}}{r_{1,1}}\right) \tag{1.19}$$

$$\theta = arcsin\left(-r_{1,3}\right) \tag{1.20}$$

$$\psi = arctan\left(\frac{r_{2,3}}{r_{3,3}}\right) \tag{1.21}$$

Questa parametrizzazione, seppur usata comunemente, può però in alcune configurazioni particolari portare a delle singolarità cinematiche ed è per questo che viene illustrata un'ulteriore parametrizzazione della matrice R, ovvero quella attraverso i *quaternioni*, che invece non presenta lo stesso problema.

1.1.2 Quaternioni

I quaternioni nascono nel 1843 da Hamilton nel tentativo di trovare un sistema di numeri che generalizzasse allo spazio tridimensionale i numeri complessi ed il loro

significato di operatori di rotazione nel piano. Si indicano solitamente con la lettera q. Un quaternione è un elemento a quattro dimensioni nello spazio $\{1, i, j, k\}$ e definito dalla combinazione lineare

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{1.22}$$

dove q_0 è detta la sua parte reale, mentre le componenti (q_1, q_2, q_3) la sua parte vettoriale. Si definisce inoltre il *quaternione unitario* come quel particolare quaternione la cui norma è unitaria, ovvero

$$\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{1.23}$$

Esso può essere ulteriormente scritto come

$$q = (q_0, \{q_i\}_1^3) = cos(\frac{\alpha}{2}) + \{q_i\}_1^3 sin(\frac{\alpha}{2}) \tag{1.24}$$

con α angolo di rotazione attorno ad un asse definito dal vettore $\{q_i\}_1^3$. Ogni quaternione unitario rappresenta infatti una rotazione nello spazio tridimensionale, così come ogni numero complesso unitario rappresenta una rotazione nel piano. La relazione che lega questi parametri ad un moto rotazionale deriva dal fatto che in un atto di moto rigido è sempre possibile individuare un asse di istantanea rotazione e l'atto di moto può essere ricondotto alla rotazione attorno a tale asse, detto *asse di Mozzi*. Si ha che una rotazione rigida nello spazio cartesiano tridimensionale può venire rappresentata dalla matrice di rotazione R, che ha la proprietà di essere ortonormale e a determinante unitario positivo, $det(R) = +1$. Si può allora associare ad ogni matrice di rotazione un quaternione unitario e viceversa. Con questa parametrizzazione la matrice di rotazione è data da

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{1.25}$$

Volendo comunque riferirsi agli angoli di Eulero come parametri d'assetto, è possibile da una generica matrice di rotazione R parametrizzata tramite quaternioni estrarre gli angoli di Roll, Pitch e Yaw tramite

$$\phi = arctan\left(\frac{2(q_2 q_3 + q_0 q_1)}{1 - 2(q_1^2 + q_2^2)}\right) \tag{1.26}$$

$$\theta = arcsin\left(-2(q_1 q_3 - q_0 q_2)\right) \tag{1.27}$$

$$\psi = arctan\left(\frac{2(q_1 q_2 + q_0 q_3)}{1 - 2(q_2^2 + q_3^2)}\right) \tag{1.28}$$

1.2 Sistema di visione

1.2.1 Introduzione al sistema di visione

Il seguente paragrafo presenta i modelli matematici alla base del sistema di visione. Il concetto è simile a quanto fatto dall'insieme occhi-cervello di un essere umano: la luce riflette sugli oggetti ed arriva alla retina del bulbo oculare che è in grado di recepirne ed apprezzarne le variazioni, l'informazione viene poi mandata al cervello che la rielabora per ricostruire l'ambiente. Senza la pretesa di ricreare un sistema tanto elaborato quanto quello occhi-cervello, il modello si rifà a questo metodo.
É necessario dare una prima definizione di sistema di visione, ovvero un assieme formato macroscopicamente da due componenti, un *hardware* (gli "occhi") ed un *software* (il "cervello"):

1. La componente *hardware* del sistema è composta da una o più videocamere con il compito di catturare e digitalizzare l'immagine dell'ambiente antistante; una videocamera è a sua volta composta da due parti principali:
   - Il gruppo ottico, formato da una o più lenti con il compito di convogliare la luce verso il componente ricettivo;
   - un piano sensore con il compito di catturare la luce derivante dal gruppo ottico e trasformarla in un'informazione digitale.
2. La componente *software* è invece composta da un elaboratore che si occupa di rielaborare le immagini sotto forma di dati digitali e da esse estrapolare le informazioni ricercate.

Generalmente il piano sensore di una videocamera coincide con un CCD (Charged-Coupled Device) o con un CMOS (Complementary Metal-Oxide-Semiconductor). La differenza tra i due sta nel tipo di uscita generata, in particolare analogica per il primo e digitale per il secondo, ma in entrambe i casi si tratta di una griglia di materiale semiconduttore capace di accumulare una carica elettrica proporzionale all'intensità della radiazione elettromagnetica che lo colpisce e restituire l'informazione sotto forma di una differenza di potenziale discretizzata. Nei paragrafi a seguire si cerca allora di descrivere il processo attravero il quale è possibile modellizzare una videocamera, a partire dalla comprensione di cosa sia e di come si formi un'immagine, all'analisi matematica di un gruppo ottico, ovvero la lente, ed infine alla modellizzazione analitica di una camera ideale e come poter usare tale modello nell'utilizzo di una camera reale.

1.2.2 Formazione e rappresentazione dell'immagine

Concettualmente, quello che si propone di fare il sistema di visione è svolgere il processo inverso della formazione dell'immagine, cercare cioè di risalire a come un raggio luminoso, una volta riflesso su di un oggetto sia arrivato al sensore della videocamera, e da queste informazioni ricreare la geometria e la posizione dell'oggetto stesso. La prima cosa che abbiamo bisogno di definire allora è cos'è e come si forma un'immagine.
Per il software, il "cervello", un'immagine coincide con un array bidimensionale (ossia una matrice) contenente i livelli di intensità luminosa I, detta anche Irradianza, rilevata dal sensore grazie al gruppo ottico che inquadra l'ambiente antistante. L'irradianza è definita come la quantità di energia che cade su di una specifica porzione d'area del sensore ed ha quindi unità di misura $\frac{W}{m^2}$. Nello specifico, per il sensore CCD (o CMOS) di una videocamera, l'unità d'area considerata è la cella formata dal semiconduttore che corrisponde ad un pixel. Ognuno di questi pixel corrisponde virtualmente ad una componente (i, j) (riga,colonna) della matrice immagine vista dall'elaboratore, ovvero ogni componente di tale matrice contiene l'informazione d'Irradianza relativa alla quantità di luce caduta su di uno specifico pixel. L'immagine viene quindi catturata dalla lente della videocamera e proiettata in 2D sul piano del sensore Ω, rappresentabile come un piano cartesiano in cui le coordinate sono date per l'appunto in pixel. In particolare, per convenzione l'origine del piano è posta nell'angolo in alto a sinistra del sensore con semiasse positivo delle ascisse verso destra e semiasse positivo delle ordinate verso il basso. Il livello di intensità luminosa I, dipende da diversi fattori, come la forma degli oggetti, le proprietà riflettenti dei materiali, le condizioni d'illuminazione delle sorgenti e il tempo di esposizione. Si ha una relazione del tipo

$$I : \Omega \subset R^2 \longrightarrow R_+; (x, y) \longrightarrow I(x, y) \tag{1.29}$$

Generalmente un'immagine a colori è la combinazione di tre di queste matrici, una per ognuno dei tre colori primari rosso, verde e blu (*Figura 1.1*), e contenenti le informazioni di Irradianza relative ad ognuno di questi colori, informazioni che rielaborate insieme danno l'informazione iniziale sul colore del singolo pixel e quindi nel complesso, dell'immagine.
Per un'immagine in scala di grigi invece vi è una sola matrice con le informazioni sull'intensità luminosa. L'Irradianza I, legata all'energia trasportata da un'onda elettromagnetica, è nella realtà una grandezza continua ad ampio spettro e necessita

Figura 1.4: Un'immagine è formata da tre matrici contenenti le informazioni di Irradianza in ognuno dei tre colori primari RGB e attraverso le quali è possibile risalire al colore finale dell'immagine

perciò di venire discretizzata, per convenzione prende infatti un valore tra 0 e 255, dove in particolare, per l'immagine in scala di grigi, lo 0 corrisponde al nero assoluto ed il 255 al bianco assoluto. La *Figura 1.5* mostra un esempio di matrice immagine contenente i livelli di Irradianza per pixel, ovvero come viene vista un'immagine da un elaboratore.

188	186	188	187	168	130	101	99	110	113	112	107	117	140	153	153	156	158	156	153
189	189	188	181	163	135	109	104	113	113	110	109	117	134	147	152	156	163	160	156
190	190	188	176	159	139	115	106	114	123	114	111	119	130	141	154	165	160	156	151
190	188	188	175	158	139	114	103	113	126	112	113	127	133	137	151	165	156	152	145
191	185	189	177	158	138	110	99	112	119	107	115	137	140	135	144	157	163	158	150
193	183	178	164	148	134	118	112	119	117	118	106	122	139	140	152	154	160	155	147
185	181	178	165	149	135	121	116	124	120	122	109	123	139	141	154	156	159	154	147
175	176	176	163	145	131	120	118	125	123	125	112	124	139	142	155	158	158	155	148
170	170	172	159	137	123	116	114	119	122	126	113	123	137	141	156	158	159	157	150
171	171	173	157	131	119	116	113	114	118	125	113	122	135	140	155	156	160	160	152
174	175	176	156	128	120	121	118	113	112	123	114	122	135	141	155	155	158	159	152
176	174	174	151	123	119	126	121	112	108	122	115	123	137	143	156	155	152	155	150
175	169	168	144	117	117	127	122	109	106	122	116	125	139	145	158	156	147	152	148
179	179	180	155	127	121	118	109	107	113	125	133	130	129	139	153	161	148	155	157
176	183	181	153	122	115	113	106	105	109	123	132	131	131	140	151	157	149	156	159
180	181	177	147	115	110	111	107	107	105	120	132	133	133	141	150	154	148	155	157
181	174	170	141	113	111	115	112	113	105	119	130	132	134	144	153	156	148	152	151
180	172	168	140	114	114	118	113	112	107	119	128	130	134	146	157	162	153	153	148
186	176	171	142	114	114	116	110	108	104	116	125	128	134	148	161	165	159	157	149
185	178	171	138	109	110	114	110	109	97	110	121	127	136	150	160	163	158	156	150

Figura 1.5: Immagine rappresentata da una matrice di livelli di intensità I

Chiarito questo, si va a spiegare nei paragrafi a seguire, passo dopo passo i modelli matematici che tramite semplificazioni e adattamenti permettono di trovare la relazione matematica che lega la luce riflessa su di un generico punto dell'ambiente antistante la videocamera, al relativo effetto che questo genera sul piano del sensore. Per fare questo si parte dal modello di *lente sottile* [8]-[9], per la modellizzazione di un sistema ottico, passando per il modello *pinhole* che si utilizza come modello

prospettico ed arrivando infine ad illustrare le caratteristiche di un'oggetto *camera*, descritto tramite la *camera ideale* integrata dai parametri necessari a poter adattare tale modello ad una videocamera reale. Si descrive infine cos'è una Stereocamera e si analizzano i motivi matematico-geometrici per i quali nel processo di ricostruzione diretta ci si avvale di una stereocamera invece che di una singola monocamera.

1.2.3 Lente sottile

A "convogliare" la luce proveniente dall'ambiente circostante al sensore della videocamera, c'è innanzituto un sistema ottico, ovvero una o un insieme di lenti che si occupa di deviare in modo controllato la direzione di propagazione dei raggi luminosi. Per un analisi semplificata, ma sensata per lo studio in questione, vengono ignorati gli effetti di riflessione e diffrazione delle lenti e si prende in considerazione la sola rifrazione. Si considera allora il più semplice modello ottico possibile, quello della *lente sottile* (*Figura 1.6*), un modello matematico definito da:

- un asse, chiamato *asse ottico*;
- un piano perpendicolare all'asse ottico, detto *piano focale*;
- l'intersezione tra piano focale ed asse ottico, chiamata *centro ottico*.

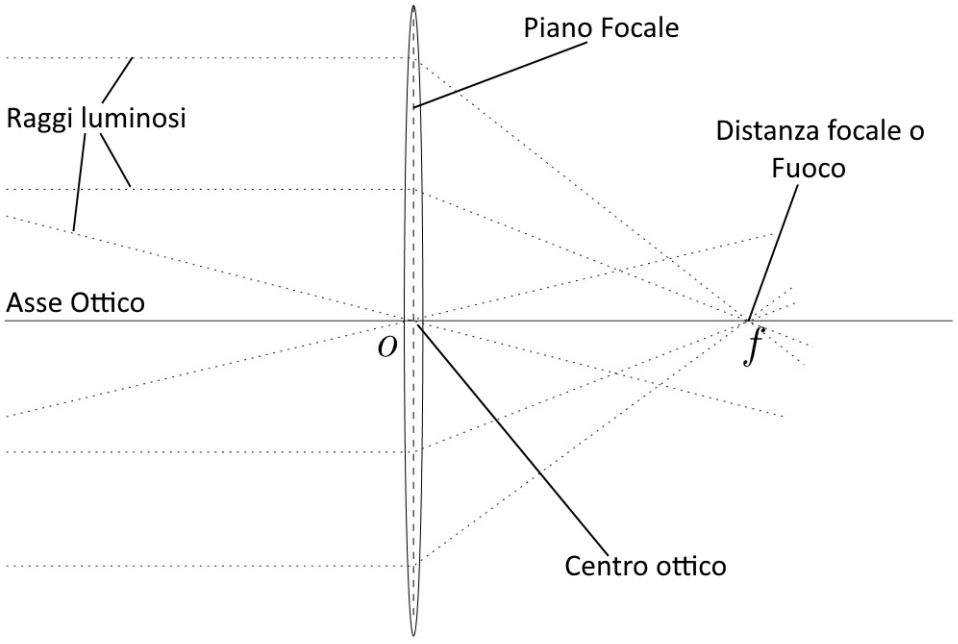


Figura 1.6: I raggi paralleli all'asse ottico intersecano nel fuoco

La lente sottile è caratterzzata da un primo importante parametro, la *lunghezza focale f*, e da due proprietà:

- tutti i raggi che intersecano il piano focale arrivando parallelamente all'assse ottico, vengono deflessi ed intersecano quest'ultimo ad una distanza dal centro ottico pari alla lunghezza focale in punto chiamato fuoco;
- tutti i raggi che intersecano il piano focale nel centro ottico non vengono deflessi.

Consideriamo ora un punto p nel semipiano frontale alla lente, non sull'asse ottico e a distanza Z dal piano focale (*Figura 1.7*).

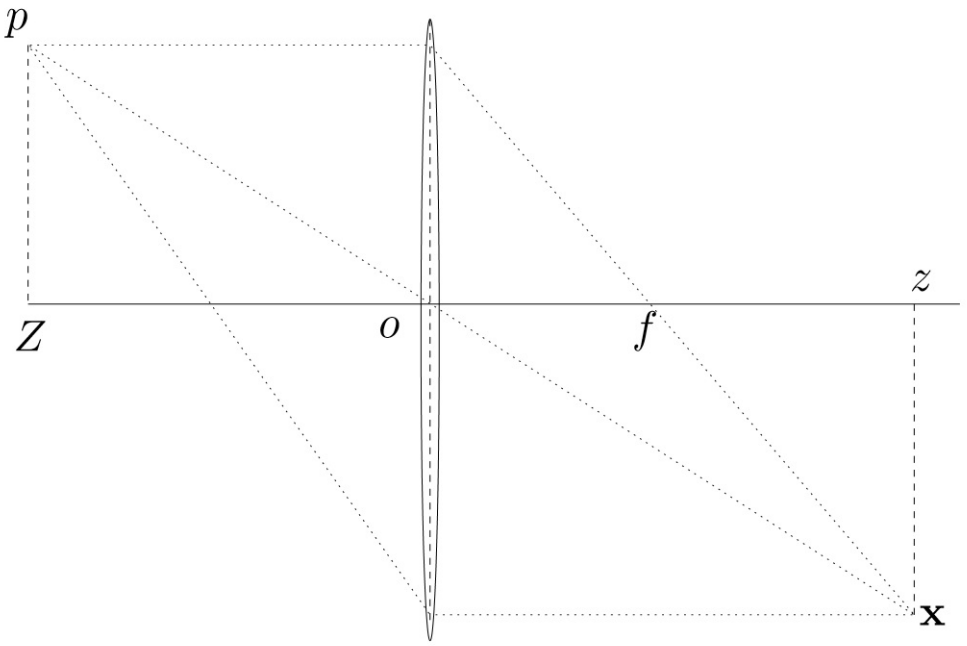


Figura 1.7: Relazione fondamentale lente sottile, x è l'immaine del punto p

Se tracciamo da esso due raggi, uno parallelo all'asse ottico ed uno diretto invece al centro ottico, il primo una volta intersecata la lente viene deflesso e interseca per il fuoco, mentre il secondo rimane non deflesso. Si chiama **x** il punto in cui i due raggi si incontrano nel semipiano opposto rispetto alla lente e z la proiezione di **x** sull'asse ottico ovvero la sua distanza dal piano focale. Attraverso le similitudini tra i triangoli si può arrivare a scrivere l'equazione fondamentale della lente sottile:

$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{f} \tag{1.30}$$

x è chiamata *immagine* del punto p.

L'irradianza I in un punto **x** è allora l'integrale dell'energia proveniente dalla regione contenuta nel cono che ha vertice in **x** stesso, cono determinato a sua volta dalla geometria della lente. Questo ci permette di nominare un'altra caratteristica propria della lente, il *campo di vista* θ, ovvero l'angolo di apetura massimo in cui il fuoco può vedere la scena antistante. Per tale angolo, detto D il diametro della lente, vale:

$$\tan\theta = \frac{D}{2f} \tag{1.31}$$

1.2.4 Modello pin-hole

Se ora si considera il modello della lente sottile, a cui però si riduce idealmente a zero l'apertura, quello che si ottiene è che tutti i raggi devono passare forzatamente per il centro ottico e rimanere quindi non deflessi. Ciò implica che gli unici punti a dare un contributo all'Irradianza del punto **x** sono tutti e soli i punti giacenti sulla retta passante per p e il centro ottico (*Figura 1.8*).
Se consideriamo $\mathbf{X} = [X, Y, Z]^T$ come le coordinate del punto p in un sistema di riferimento centrato nel centro ottico e il cui asse Z sta sull'asse ottico, si ha secondo le leggi della *prospettiva ideale* che:

$$x = -\frac{X}{Z}f \tag{1.32}$$

$$y = -\frac{Y}{Z}f \tag{1.33}$$

con (x, y) le coordinate del punto **x** su di un piano a distanza f dal centro ottico, detto *piano immagine*.
Questo prende il nome di *modello ideale pin-hole* ovvero il modello prospettico che permette in prima battuta e sotto le semplificazioni viste, di legare le coordinate di un punto p alle coordinate della sua *immagine* sul piano immagine. Si possono notare i segni negativi nelle equazioni 1.32 e 1.33 delle coordinate sul piano focale che stanno ad indicare come un'immagine reale viene ricostruita capovolta sul piano immagine; se si vuole eliminare questo effetto si può pensare di assegnare a (x,y) (-x,-y), che corrisponde ad avere il piano immagine in $z = f$ invece che in $z = -f$, ovvero frontale al centro ottico.

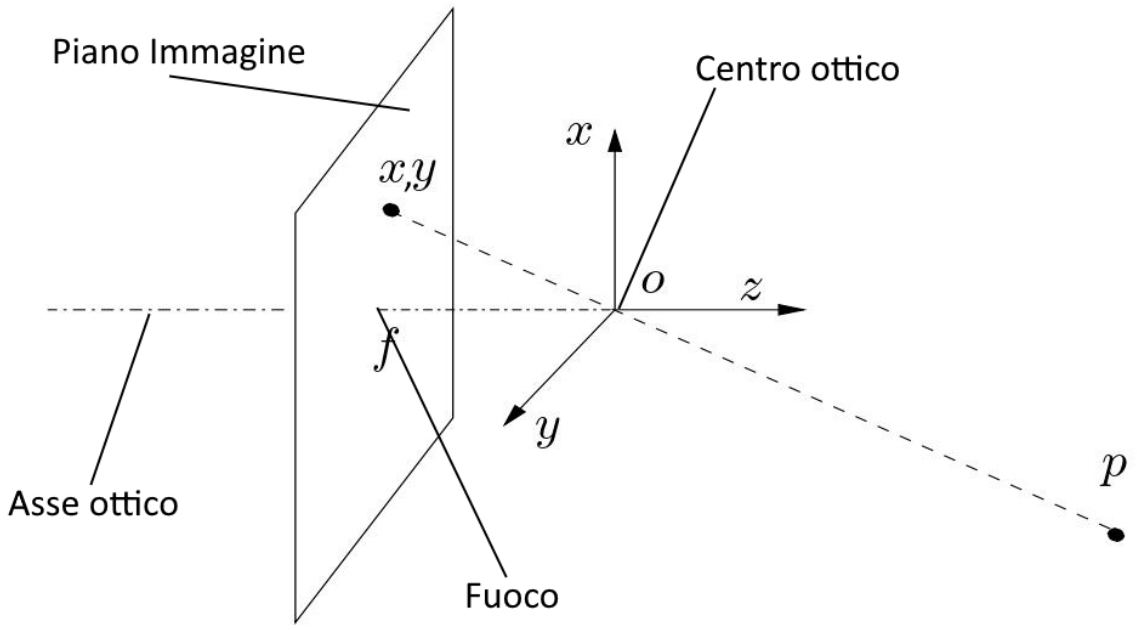


Figura 1.8: Proiezione del punto p sul piano focale o *piano immagine*

1.2.5 Modello camera ideale

Ricapitolando, si può ridurre il processo di formazione dell'immagine al tracciamento dei raggi luminosi provenienti dai punti dell'ambiente ai rispettivi pixel, tramite le leggi della proiezione prospettica descritte dal modello *pin-hole* ed un cambio di coordinate. Quello che si vuole fare è allora trovare una correlazione tra un generico sistema di riferimento ambiente inizialmente con un sistema di riferimento camera tridimensionale e quindi successivamente con il sistema di riferimento bidimensionale del piano immagine.
Per la prima di queste correlazioni, considerando allora una generica terna ortogonale $\{o, X, Y, Z\}$ chiamata riferimento *world* "w", si possono scrivere le coordinate del punto p come

$$\mathbf{X}_w = [X_w, Y_w, Z_w]^T \in R^3 \tag{1.34}$$

e volendo scrivere le coordinate dello stesso punto p in un altro sistema di riferimento, come un riferimento *camera* "c", è necessario, riprendendo quanto ricordato in *Sezione 1.1*, conoscere la trasformazione geometrica tra una terna e l'altra, perciò, una relazione tale che

$$X_w = g_{wc}(X_c) \tag{1.35}$$

e

$$X_c = g_{cw}(X_w) \tag{1.36}$$

con

$$g_{wc} = {g_{cw}}^{-1}. \tag{1.37}$$

Tale trasformazione è del tipo

$$g_{cw} = (R_{cw}, T_{cw}) \tag{1.38}$$

con $T_{cw} \in R^{[3\times1]}$ vettore di traslazione e $R_{cw} \in R^{[3\times3]}$ matrice di rotazione. Il cambio di coordinate allora è dato da:

$$X_c = g_{cw}(X_w) = R_{cw}X_w + T_{cw} \tag{1.39}$$

o passando a coordinate omogenee,

$$\bar{X}_c = [X_c, 1]^T = \bar{g}_{cw}\bar{X}_w \tag{1.40}$$

con

$$\bar{X}_w = [X_w, 1]^T \tag{1.41}$$

e

$$\bar{g}_{cw} = \begin{bmatrix} R_{cw} & T_{cw} \\ 1 & 1 \end{bmatrix} \in R^{[4\times4]} \tag{1.42}$$

Accorpiamo i modelli visti fin'ora e per semplicità si chiameranno d'ora in avanti solo $\mathbf{X} = [X, Y, Z]^T$ le coordinate del generico punto p nel sistema di riferimento *camera*, $\mathbf{X}_0 = [X_0, Y_0, Z_0]^T$ le coordinate dello stesso punto nel riferimento *world* ed R e T rispettivamente la matrice di rotazione e il vettore di traslazione per passare dalla terna *world* alla terna *camera*. Si ha perciò

$$\mathbf{X} = g(\mathbf{X}_0) = R\mathbf{X}_0 + T \tag{1.43}$$

oppure

$$\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_0 \\ 1 \end{bmatrix} \tag{1.44}$$

Si vuole ora trovare invece la correlazione tra il sistema di riferimento *camera* e il sistema di riferimento del *piano immagine*. Si sceglie di prendere il sistema di riferimento *camera* coincidente con il sistema di riferimento posto nel centro ottico

come in *Sezione 1.2.4*, ovvero con asse Z lungo l'asse ottico, e adottando il modello *pinhole*, le coordinate $\mathbf{X}$ del punto p in terna camera vengono proiettate sul piano immagine secondo

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = -\frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{1.45}$$

dove $[x, y]^T$ sono le coordinate del punto p proiettato sul piano immagine e riferite ad sistema di riferimento bidimensionale con origine nel punto centrale o principale (coincidente con intersezione tra l'asse ottico ed il piano focale) ed f la distanza focale. In coordinate omogenee l'equazione 1.45 si può riscrivere come

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.46}$$

Finchè la coordinata Z (la profondità) rimane incognita, viene indicata con uno scalare arbitrario $\lambda \in R_+$. Si fa inoltre notare che la matrice dell'equazione 1.46 può essere decomposta in

$$\begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.47}$$

Riprendendo l'equazione 1.44 con i vettori coordinate in forma estesa

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 1 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{1.48}$$

ed unendo all'equazione 1.46, si ottiene l'equazione fondamentale del *modello di camera ideale*,

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 1 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{1.49}$$

L'equazione 1.20 permette quindi di ricavare, a partire dalle coordinate tridimensionali di un generico punto in riferimento *world*, le coordinate della proiezzione dello stesso punto sul *piano immagine*, conoscendo la trasformazione geometrica che lega il riferimento *world* al riferimento *camera*, e la distanza focale propria della lente.

1.2.6 Parametri intrinseci

Nei modelli *pinhole* e di *camera ideale* (*Sezioni 1.2.4-1.2.5*), ci si riferisce al *piano immagine* come un piano ideale parallelo al piano focale e posto ad una distanza pari alla distanza focale f dal centro ottico. Questo, in determinate ipotesi semplificative, può essere fatto combaciare con il piano stesso del sensore della videocamera reale. Tali ipotesi semplificative sono che le dimensioni dei pixel combacino con la dimensione unitaria di lunghezza, ovvero il metro. Risulta evidente che per ragioni costruttive così non è, e per poter adattare tali modelli ideali all'utilizzo e la modellizzazione di una camera reale, è necessario introdurre alcune relazioni che ci permettano di riferirci ad una specifica geometria costruttiva. Per legare quindi il *piano immagine* al piano del sensore, si va a specificare la relazione tra le dimensioni dei pixel. Se le coordinate immagine su *piano immagine* del punto p $[x, y]^T$ sono pensate in unità di misura metriche, la trasformazione in coordinate pixel è data dalla matrice di scalari

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.50}$$

con $[x_s, y_s]^T$ le coordinate del punto sul piano del sensore e S_x ed S_y le densità di pixel per metro sul sensore, che possono essere in genere anche differenti tra loro nel caso i pixel non siano quadrati ma rettangolari. Inoltre, si usa solitamente indicare un dato punto sul piano del sensore riferendosi alle sue coordinate cartesiane rispetto ad un'origine posta nell'angolo in alto a sinistra, mentre le coordinate sul piano immagine $[x, y]^T$ sono riferite al centro ottico (o punto principale, *Sezione 1.2.4*). E' necessaria perciò una traslazione, del tipo

$$x_{im} = -(x_s - O_x) \tag{1.51}$$

$$y_{im} = -(y_s - O_y) \tag{1.52}$$

con (O_x, O_y) le coordinate (in pixel) del centro. Per cui, le reali coordinate immagine $[x_{im}, y_{im}]^T$ sono legate a quelle del piano immagine ideale $[x, y]^T$ da

$$\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} -S_x & 0 & O_x \\ 0 & -S_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1.53}$$

Compare un ulteriore parametro che solitamente viene ignorato ed utilizzato uguale a zero, ma di cui si fa accenno poichè in realtà anche se molto piccolo compare nei test eseguiti. Questo è S_θ, chiamato *skew*, legato alla cotangente dell'angolo

tra l'asse x e l'asse y del piano del sensore che possono non essere perfettamente ortogonali ad esempio per incertezze costruttive, anche se in minima misura. La matrice di trasformazione dell'equazione 1.53 diviene allora

$$A = \begin{bmatrix} -S_x & -S_\theta & O_x \\ 0 & -S_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.54}$$

In aggiunta a questo cambio di coordinate, si può avere, ad esempio in caso di campi di vista molto ampi, che l'immagine proiettata sul piano presenti un certo grado di distorsione, generalmente in direzione radiale ma talvolta anche in direzione tangenziale, un esempio in *figura 1.9*. Il modello che ci permette di "ricostruire"

Figura 1.9: Esempio di distorsione dovuta ad un ampio campo di vista

un'immagine assegnando gli effettivi punti ottenuti alla posizione che dovrebbero avere senza distorsione è tipicamente modellato, per la distorsione radiale da

$$x = x_d(1 + a_1 r^2 + a_2 r^4 + a_3 r^6 + ...) \tag{1.55}$$

$$y = y_d(1 + a_1 r^2 + a_2 r^4 + a_3 r^6 + ...) \tag{1.56}$$

con (x_d, y_d) le coordinate dei punti distorti, $r = \sqrt{{x_d}^2 + {y_d}^2}$ e la sommatoria che continua a seconda del numero di coefficienti di distorsione radiale $a_1, a_2, ...$ con cui si decide di modellizzare il fenomeno. Mentre il modello per la rettifica della distorsione tangenziale è dato da

$$x = 2p_1 x_d y_d + p_2(r^2 + 2{x_d}^2) \tag{1.57}$$

$$y = p_1(r^2 + 2{y_d}^2) + 2p_2 x_d y_d \tag{1.58}$$

con p_1, p_2 coefficienti di distorsione tangenziale.

Combinando quanto visto fin'ora con l'equazione 1.49 e nell'ipotesi di non avere distorsione o di averla già rettificata tramite metodi opportuni, la relazione finale che lega i punti nel sistema di riferimento *camera* a i punti realmente individuati sul piano del sensore, è

$$\lambda \begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} -S_x & -S_\theta & O_x \\ 0 & -S_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1.59}$$

Ricapitolando, il processo analitico di formazione dell'immagine di una camera reale è ottenuto in primo luogo dalle relazioni del modello prospettico *pinhole*, che proiettano la coordinate tridimensionali dei punti della scena inquadrata su un sistema di riferimento immagine ideale, e in secondo luogo da un'uteriore trasformazione dovuta a parametri che dipendono dalla specifica camera, come:

- lunghezza focale f;
- i fattori di scala S_x ed S_y;
- lo *skew*;
- le coordinate immagine del centro ottico (O_x, O_y).

La matrice che si ottiene moltiplicando la matrice dei fattori di scala e quella contenente le distanze focali,

$$K = \begin{bmatrix} -S_x & -S_\theta & O_x \\ 0 & -S_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} fS_x & fS_\theta & O_x \\ 0 & fS_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.60}$$

è una matrice di particolare importanza contenente quelli che vengono chiamati *parametri intrinseci* della specifica camera ed è appunto essa detta *matrice dei parametri intrinseci* o meglio *matrice di calibrazione*.

Si conclude il paragrafo sommando tutte le trasformazioni geometriche citate (equazioni $1.49 - 1.59$) e scrivendo in forma estesa l'equazione finale che lega le coordinate geometriche di un punto in terna *world* $\mathbf{X}_0 = [X_0, Y_0, Z_0]$ alle coordinate in pixel

$\mathbf{x}_{im} = [x_{im}, y_{im}]$ effettivamente rilevate sul piano immagine della particolare camera, dopo aver applicato un opportuno metodo di distorsione inversa, conoscendo i *parametri intrinseci* contenuti nella *matrice di calibrazione* e i cosidetti *parametri estrinseci* necessari alla trasformazione dal sistema di riferimento *world* al sistema di riferimento camera, ovvero la matrice di rotazione R e il vettore traslazione T

$$\lambda \begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} fS_x & fS_\theta & O_x \\ 0 & fS_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 1 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{1.61}$$

o in forma compatta, chiamando K la matrice dei parametri intrinseci, A la matrice rettangolare unitaria e $\mathbf{T}$ (da non confondere con T) la trasformazione da riferimento *world* a *camera*

$$\lambda[\mathbf{x}_{im}, 1]^T = [K][A][\mathbf{T}][\mathbf{X}_0, 1]^T \tag{1.62}$$

É bene notare che tramite l'equazione 1.62 è possibile in modo diretto ed univoco ottenere il vettore coordinate immagine $\mathbf{x}_{im}$ a partire da un vettore di coordinate *world* $\mathbf{X}_0$, ma non è possibile ottenere il processo inverso, non è cioè possibile da questa sola equazione ricavare le coordinate di un generico punto in un qualsivoglia sistema di riferimento tridimensionale a partire dalla sua proiezione sul piano del sensore. Questo è dovuto al fatto che l'Irradianza accumulata su di un generico pixel, è la somma dell'intensità luminosa che proviene da un qualunque punto giacente sulla retta che passa per il punto che viene proiettato sul pixel stesso ed il centro ottico (*Sezione 1.2.4*), e questo comporta la perdita di un'informazione nel processo proiettivo diretto, non immediatamente recuperabile nel processo proiettivo inverso. L'informazione che si perde è quella sulla profondità, quindi la coordinata z del punto p in riferimento *camera*, profondità che in eqauzione 1.46 viene infatti sostituita con un parametro scalare λ incognito.

1.2.7 Ricostruzione da visione multipla: la Stereocamera

Come descritto nei paragrafi 1.2.4, 1.2.5, 1.2.6, attraverso semplici trasformazioni geometriche derivanti dallo studio di modelli ottici semplificati e la possibilità di rendere questi modelli utilizzabili per una telecamera specifica, si può ricostruire il processo matematico analitico che lega un punto $\mathbf{X}_0 = [X_0, Y_0, Z_0]^T$ nello spazio tridimensionale in riferimento *world*, ad un punto $\mathbf{x}_{im} = [x_{im}, y_{im}]$ sul piano del sensore

in particolare tramite l'*equazione 1.61*. Il problema che ci si propone di risolvere è il processo inverso, risalire cioè dal punto sul piano immagine del sensore al punto nel sistema tridimensionale. Per fare questo l'equazione citata non è sufficiente, a causa della perdita di informazione dovuta al modello proiettivo, come descritto analizzando l'*equazione 1.62*. Ciò implica che attraverso l'uso di una singola videocamera, o monocamera, non sia possibile ricorstruire per via diretta la posizione nello spazio di un determinato punto, se non conoscendo altre informazioni a priori, come ad esempio grandezze caratteristiche note dell'oggetto a cui il punto d'interesse appartiene osservato a distanza nota, e in ogni caso attraverso l'uso di metodi iterativi di confronto di diversi punti, ad esempio il lavoro presentato in [5]. Seppur possano questi metodi essere in alcuni casi anche piuttosto precisi, possono in alcune condizioni richiedere uno sforzo computazionale decisamente elevato e quindi dei tempi di ricostruzione non accettabili. Con una stereocamera, o più in generale da visioni multiple, è invece possibile, triangolarizzando, risalire alla posizione tridimensionale del punto per via diretta. Si descrive in questo paragrafo come questo si rende possibile.

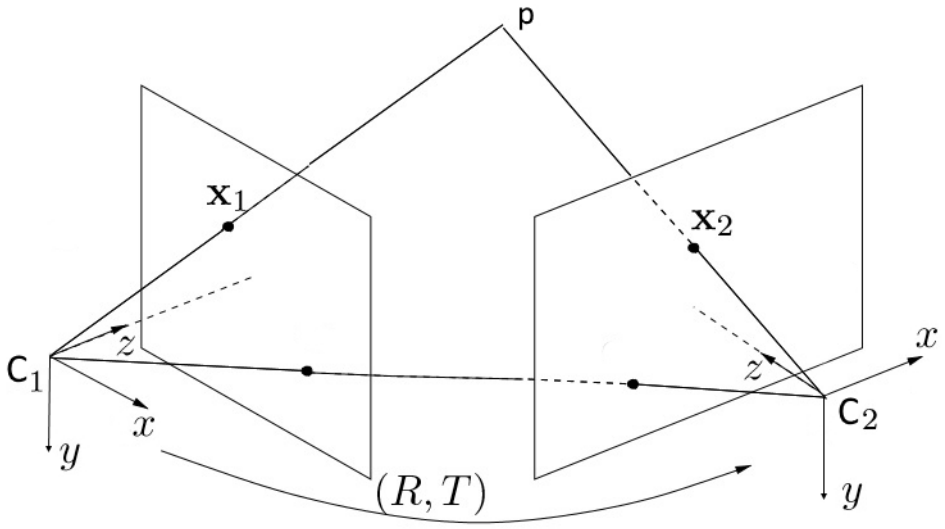


Figura 1.10: Modellizzazione stereocamera

Una stereocamera è un sistema di due videocamere, ognuna con il proprio gruppo ottico ed il proprio sensore, che inquadrano in modo sincrono la stessa scena da due punti di vista differenti, fornendo perciò differenti immagini dello stesso soggetto. Supponiamo allora di riconoscere uno stesso punto p nello spazio tridimensionale e di conoscere le coordinate delle sue proiezioni $\mathbf{x}_1$ e $\mathbf{x}_2$ sui piani sensori delle due camere

come in *Figura 1.10*, coordinate legate tra di loro da precise relazioni geometriche. Se si considera come sistema di riferimento principale il riferimento *camera* della camera di sinistra, si può considerare la terna solidale alla camera destra individuata tramite una matrice di trasformazione formata al solito da una matrice di rotazione R e un vettore di traslazione T. Allora chiamando $\mathbf{X}_1 \in R^3$ le coordinate spaziali del punto p rispetto alla terna della camera sinistra e $\mathbf{X}_2 \in R^3$ quelle riferite alla camera destra, per le relazioni descritte in *Sezione 1.1*, vale

$$\mathbf{X}_1 = R\mathbf{X}_2 + T \tag{1.63}$$

Le due camere possono essere generalmente disposte in diversi modi, riferendoci ai loro assi. Senza perdere di generalità si considera d'ora in poi una configurazione con camere ad assi paralleli. Si considerano inoltre paralleli tra loro anche gli assi dei sistemi di riferimento camera paralleli agli assi del piano immagine, quindi x_i e y_i, come in *Figura 1.11*. Ciò si traduce nell'avere la matrice di rotazinoe R tra le due camere pari alla matrice identità ed il vettore traslazione con tutte le componenti nulle a meno di una lungo la componente x, pari alla distanza tra i centri delle due terne ovvero pari alla distanza tra i punti focali delle due camere. Tale distanza è detta *baseline* b ed è uno dei parametri fondamentali della stereocamera.

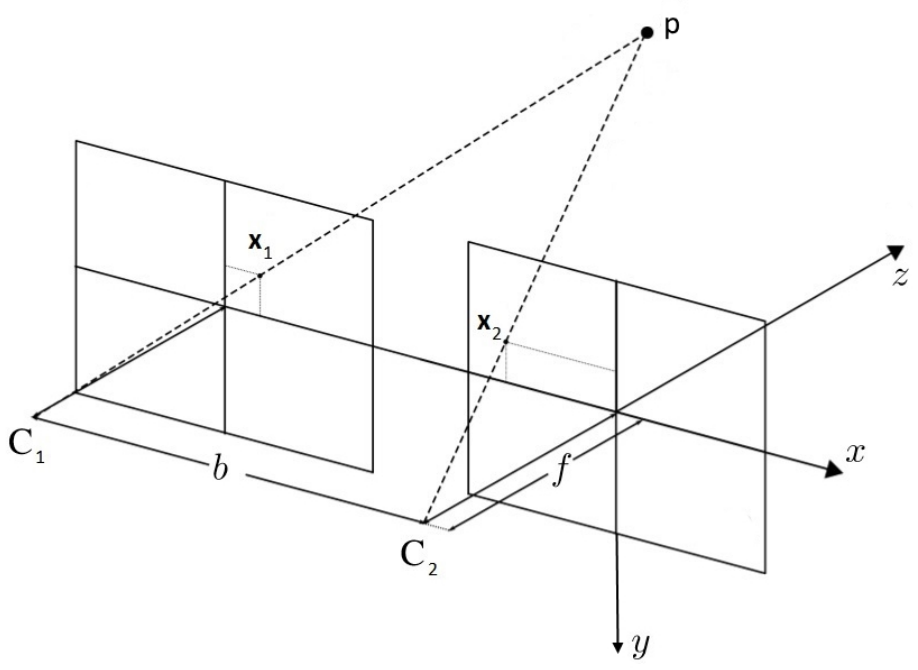


Figura 1.11: Modello di stereocamera ad assi paralleli

In questa configurazione, il piano formato dai centri delle terne delle due camere,

corrispondenti ai loro punti focali, ed un punto p, è quello che prende il nome di piano *epipolare* e che porta a definire un vincolo importante nel processo di proiezione del punto sui piani immagine delle due camere. Riconsiderando ora infatti le coordinate $\mathbf{x}_1 = [x_1, y_1]$ e $\mathbf{x}_2 = [x_2, y_2]$ del punto proiettato sui piani immagine delle due camere, questi hanno una diversa componente x_i, tale differenza prende il nome di *disparity* d ed è un parametro importante nel processo di ricostruzione della profondità, ma hanno per costruzione geometrica la stessa componente y_i e questo è quello che prende il nome di *vincolo epipolare*.

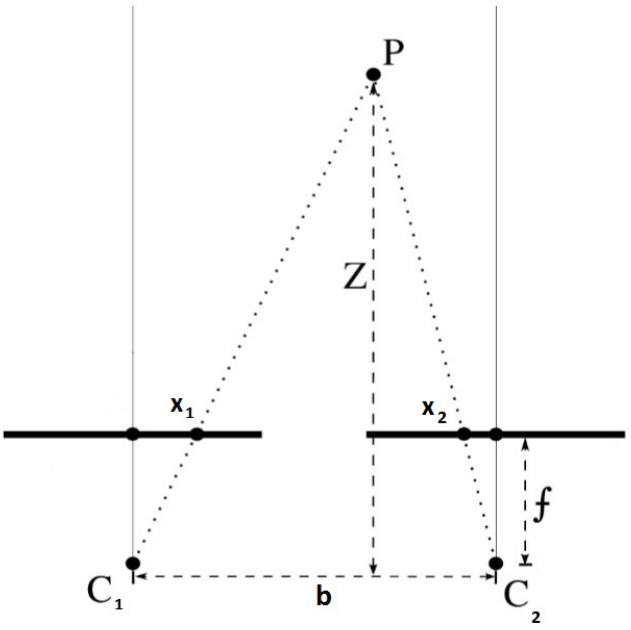


Figura 1.12: Relazione tra profondità Z e *disparity*

Considerare il sistema visto dall'alto come in *Figura 1.12* può aiutare a spiegare come utilizzare questa modellizzazione per risalire alla profondità del punto p. Per far questo si inizia trovando una relazione tra i triangoli di costruzione, infatti i triangoli C_1PC_2 e $\mathbf{x}_1P\mathbf{x}_2$ sono simili tra loro e questo permette di scrivere la proporzione

$$\frac{b}{Z} = \frac{b + x_2 - x_1}{Z - f} \tag{1.64}$$

dove b è la *baseline*, Z la profondità, f la distanza focale e la differenza $x_2 - x_1$ la *disparity* d. Esplicitando Z si ottiene

$$Z = \frac{bf}{d} \tag{1.65}$$

Si nota allora come, fissate la distanza focale f e la *baseline* b, la profondità Z dipenda solo dalla *disparity* d e sia ad essa inversamente proporzionale. Conoscendo

Z è poi possibile tramite l'*equazione 1.45* ricavare anche le coordinate X e Y rispetto alla terna desiderata.

1.3 I fiducial markers

Uno dei problemi che ci si propone di risolvere con il sistema di visione è quello di stimare la posizione di un punto preciso nella scena antistante al sistema stesso, rispetto ad un sistema di riferimento tridimensionale scelto. Per fare questo in modo diretto, si sceglie di utilizzare una stereocamera la cui modellizzazione è presentata in *Sezione 1.2.7*. Ricordando l'*equazione 1.65*, si è visto come conoscendo alcune caratteristiche della stereocamera come la *distanza focale* f e la *baseline* b, sia possibile, conoscendo la *disparity* tra le proiezioni del punto sui piani immagine delle due camere, risalire alla profondità e conseguentemente alle altre coordinate spaziali del punto stesso. Questo sottointende un ulteriore problema, la cui soluzione è spiegata nel presente paragrafo. Il problema è quello del *matching*, ovvero la necessità di sapere con certezza che le due proiezioni che si stanno considerando sui piani immagine delle camere per la ricostruzione della profondità, si riferiscano effettivamente allo stesso punto della scena inquadrata dalla stereocamera. Per porre rimedio a questo si possono usare degli oggetti la cui proiezione sul piano immagine sia facilmente ed univocamente riconoscibile analizzando l'array bidimensionale contenente l'immagine stessa. Per fare questo le opzioni per gli oggetti utilizzabili sono svariate, da elementi strutturali particolari, ad emettitori infrarossi o a semplici decalcomanie, da applicare sul punto di cui si vuole stimare la distanza. In questo lavoro si sceglie di usufruire dell'ultima opzione, ricorrere cioè all'utilizzo di *fiducial markers*. Questo per diverse ragioni:

- semplicità di carattere costruttivo come la facilità di fabbricazione ed applicazione;
- la possibilità di eventuali modifiche in corso d'opera in modo veloce e non dispendioso;
- legato al motivo precedente, la possibilità di esaminare quindi anche diverse opzioni per giungere all'ottimale senza elevato dispendio di energie;
- i bassissimi costi che essi comportano.

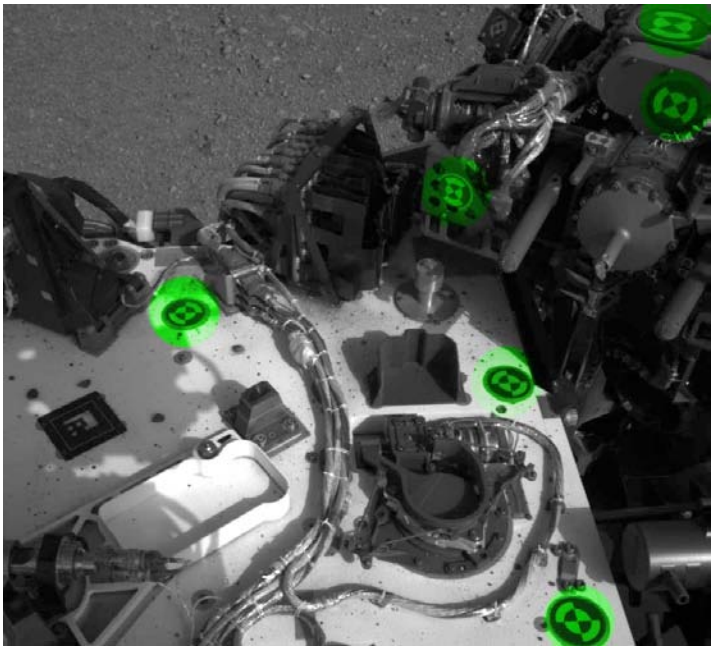


Figura 1.13: Esempio di fiducial markers in ambito spaziale

Il concetto è simile a quello dei codici a barre o degli odierni QRcode, una figura, generalmente disegnata in nero su fondo bianco (o viceversa) per facilitarne l'individuazione, contenente nelle sue stesse caratteristiche geometriche le informazioni necessarie al suo riconoscimento univoco. Sono attualmente utilizzati in svariati ambiti, dalla robotica, all'attività spaziale stessa (*Figura 1.13*) ma anche alla vità di tutti i giorni grazie alla presenza sempre più consistente di telecamere al altissima definizione sugli smartphone in commercio. Questo porta ad avere un'ampissima scelta di figure adatte al riconoscimento già esistenti ed utilizzabili. Nell'ottica di creare ed utilizzare qualcosa che sia il più semplice possibile e costruito su misura per questo lavoro, sono stati però pensati da zero dei nuovi markers.
Per la scelta delle *features* ossia le caratteristiche grafiche dei nostri markers, si parte da pochi semplici concetti che si traducono nei seguenti requisiti:

- il marker deve avere una geometria tale da facilitarne il riconoscimento, quindi possibilmente con una forma macroscopica differente da altre forme individuabili nella scena inquadrata;

- nel caso vi siano più punti d'interesse da riconoscere e quindi più markers, deve essere possibile riconoscere *univocamente* un marker rispetto ad un altro;

- il riconoscimento deve essere robusto e soprattutto veloce, questo implica che i markers debbano essere figure il più semplici possibile in modo da non generare

un sovraccarico computazionale.

Si devono decidere quindi la forma e la feature dei markers, aspetti tra loro strettamente legati poichè influenzano il metodo di riconoscimento delle stesse. Si sceglie innanzitutto di scartare le forme poligonali, in quanto in un lavoro precedente già citato [5] sono stati testati dei markers di forma quadrata, che una volta riconosciuti davano l'informazione attraverso il riconoscimento di altre features al loro interno; le tempistiche di rilevamento non risultavano però accettabili e soprattutto si vuole che l'informazione che permette il riconoscimento della feature del marker sia intrinseca nel marker stesso. La scelta è allora ricaduta su forme circolari, anche grazie alla ricerca e scoperta in letteratura recente di un algoritmo per il rilevamento delle stesse (più in generale per il rilevamento di forme ellittiche) particolarmente efficiente e soprattutto veloce [10] grazie ad un concetto semplice, ovvero che il metodo di selezione non è svolto attraverso la ricerca delle features su ogni singolo punto dell'immagine ma attraverso strutture più macroscopiche che portano a diminuire drasticamente il costo computazionale. L'algoritmo è stato ideato da due ricercatori, Michele Fornaciari dell'Università di Modena e Andrea Prati dello IUAV di Venezia, e viene illustrato nel dettalgio in *Sezione 1.4*.
Una volta decise le caratteristiche dei markers, i passi per il loro riconoscimento sono macroscopicamente:

1. ridurre l'immagine da tre matrici d'Irradianza RGB (*Sezione 1.2.2*) ad un'unica matrice d'Irradianza in scala di grigi, ovvero rendere l'immagine da colori a bianco e nero. Questo si fa per facilitare il riconoscimento dei contorni accentuando i contrasti luminosi;

2. ricercare all'interno dell'immagine le forme ellittiche tramite l'algoritmo Fornaciari-Prati;

3. selezionare tra le forme ellittiche riconosciute, quelle le cui caratteristiche combaciano con le features create.

Per riuscire in questo ultimo passo si utilizzano informazioni relative ad esempio a determinati valori d'Irradianza, combinazioni di diverse forme circolari o rapporti dimensionali di grandezze caratteristiche che dipendono da come effettivamente si decida di creare e riconoscere i markers. Nello specifico i markers creati per questo lavoro sono presentati nella descrizione del Set Up sperimentale in *Sezione 2.3*.

1.4 Algoritmo Fornaciari-Prati

In questa sezione viene descritto l'algoritmo presentato in [10], ovvero il metodo scelto per il riconoscimento delle forme ellittiche nell'immagine elaborata. Esistono allo stato dell'arte diversi metodi disponibili per il riconoscimento di svariate geometrie siano esse poligonali o circolari/ellittiche. Il più utilizzato fra questi è la trasformata di Hough applicata alle ellissi con 5 parametri (coordinate del centro, angolo con l'asse x e semiassi maggiore e minore): essa si occupa di utilizzare la relazione biunivoca che lega i parametri dell'equazione analitica di una figura geometrica con le coordinate cartesiane dei punti che appartengono alla figura stessa, iterando quindi alla ricerca di tutti i punti sul piano immagine che rientrano in tale figura. Questo comporta un peso computazionale elevato ed infatti la maggior parte delle ricerche fatte a suo seguito sono atte a trovare un modo per alleggerire tale peso, ad esempio applicando risolutori probabilistici [15].
L'algoritmo Fornaciari-Prati utilizza invece un altro metodo di ricerca della figura, basato sul calcolo delle ellissi che meglio approssimano la forma riscontrata dai bordi calcolati all'interno dell'immagine. Questo è infatti il primo di diversi vantaggi:

- non va a ricercare la corrispondenza sui singoli punti del piano immagine ma su interi bordi già riconosciuti a priori, riducendo quindi i dati da elaborare;
- la strategia di selezione si basa su una proprietà del punto medio di corde parallele che tagliano l'ellisse, non sono quindi necessarie particolari informazioni sulla direzione dei bordi ed è quindi adatta ad analizzare immagini reali;
- vengono usati solo i dati raccolti dalla strategia di selezione per stimare le coordinate del centro, non dovendo quindi ricorrere ad accumulatori;
- non è necessario siano riconosciuti i bordi dell'intera figura perchè venga riconosciuta un'ellisse e ciò aumenta la robustezza del riconoscimento in caso di occlusioni parziali;
- la selezione non si basa su vincoli dati su parametri come eccentricità, dimensioni dei semiassi o distanze massime tra i bordi, è perciò il più generale possibile.

tutto ciò ha come risultato uno dei più veloci algoritmi all'attuale stato dell'arte, ed è inoltre piuttosto accurato.
Esso prevede tre fasi:

1. un pre-processing atto a riconoscere i segmenti di bordo, ovvero punti tra essi collegati e che condividono una stessa direzione di convessità;
2. una strategia di selezione dei bordi per il riconoscimento delle ellissi e la stima dei relativi parametri;
3. infine un post-processing dove vengono analizzati,uniti e scremati i riscontri multipli di una stessa ellisse.

1.4.1 Pre-processing

La fase di pre-processing è quella che si occupa del riconoscimento dei bordi, l'immagine viene quindi trasformata in scala di grigi, smussata con un filtro Gaussiano e vengono estratti i contorni con un *Canny edge detector* che restituisce anche i gradienti di luminosità nelle due direzioni, dX e dY, dai quali è possibile estrapolare anche un'indicazione del probabile orientamento di quel contorno.
I bordi vengono così suddivisi in due classi principali, {1,3} o {2,4} (*Figura 1.14* a sinistra), a seconda del segno della loro derivata dY/dX che restituisce l'inclinazione della tangente al bordo:

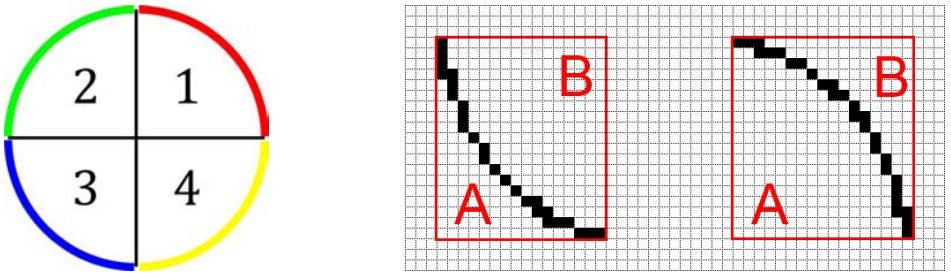


Figura 1.14: Classi di bordi rappresentanti un'ellisse e relativa convessità

Per ogni classe di bordi riconosciuti viene poi fatta una prima scrematura eliminando quelli troppo corti ottenuti a causa di rumore o quelli dritti, sicuramente non appartenenti ad una forma ellittica. Questi ultimi vengono individuati analizzando il rapporto tra le due aree in cui il segmento suddivide il rettangolo stesso che lo contiene, ugual metodo che si utilizza poi per rilevare il verso della convessità: se il rapporto non soddisfa determinati vincoli viene scartato come segmento dritto, altrimenti a seconda che l'area maggiore sia la A o la B (*Figura 1.14* a destra) si avrà convessità verso il basso o verso l'alto.

1.4.2 Selezione dei contorni e rilevamento ellissi

Una volta riconosciuti i bordi e la loro classe di appartenenza, bastano tre di essi appartenti alla stessa ellisse per poter individuare e calcolare i parametri dell'ellisse stessa. Per evitare di iterare su tutte le combinazioni possibili tra i contorni trovati, è applicata una prima strategia risolutiva; c'è infatti un'ovvia correlazione tra la classe i di un segmento e le classi j e k dei segmenti che rispettivamente lo seguono e lo precedono. Considerando allora di prenderli in senso antiorario, le sole possibili classi di terne (i, j, k) saranno $\{(1, 2, 4), (2, 3, 1), (3, 4, 2), (4, 1, 3)\}$.
Inoltre è evidente che tutti i bordi appartenenti alla stessa ellisse guardano allo stesso centro e si può quindi introdurre un secondo vincolo dato dall'obbligata mutua posizione tra i bordi delle quattro classi appartenenti ad una stessa ellisse, ovvero:

	1	2	3	4
1		right		above
2	left		above	
3		below		left
4	below		right	

Tabella 1.1: Mutua posizione tra le classi di bordi

Tra tutte le terne (e_i, e_j, e_k) che soddisfano queste due condizioni si procede alla stima delle coordinate del centro dell'ellissi per le due coppie (e_i, e_j) e (e_i, e_k). Se i centri così calcolati sono vicini abbastanza si è allora trovata un'ellisse e si può procedere al calcolo dei suoi parametri.
Il calcolo delle coordinate del centro a partire da una coppia di bordi è implementato utilizzando una proprietà delle ellissi: date due corde parallele tra loro, la retta che passa per i loro punti medi passa anche per il centro dell'ellisse. Presa allora una coppia del tipo (e_a, e_b) con (a, b) appartenenti a $\{(1, 2), (2, 3), (3, 4), (4, 1)\}$, le coordinate del centro si possono calcolare con

$$C_{ab}x = \frac{M_b y - t_b M_b x - Ma.y + t_a + M_a x}{t_b - t_a} \tag{1.66}$$

$$C_{ab}y = \frac{t_a M_b y - t_b M_a y + t_a t_b (M_a x - M_b x)}{t_b - t_a} \tag{1.67}$$

dove (P_a, P_b) e (H_a, H_b) sono rispettivamente due estremi e i punti medi dei segmenti considerati, ph_a e ph_b sono i segmenti che li congiungono ed M_a e M_a sono i punti medi di tali segmenti (*Figura 1.15*).

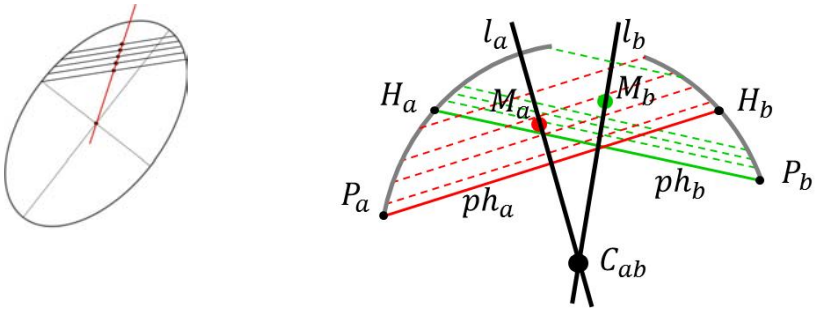


Figura 1.15: Metodo per il calcolo delle coordinate dei centri dell'ellisse data una coppia di bordi del tipo (e_a, e_b)

Una volta trovati i centri dalle due coppie (e_i, e_j) e (e_i, e_k) si calcola il centro definitivo facendone una semplice media

$$x_c = \frac{C_{ij}x + C_{ik}x}{2} \tag{1.68}$$

$$y_c = \frac{C_{ij}y + C_{ik}y}{2} \tag{1.69}$$

e si ottengono poi per decomposizione tutti gli altri parametri d'interesse come A e B (rispettivamente il semiasse maggiore e quello minore), l'orientazione rispetto all'asse delle ascisse ed anche un parametro di accuratezza sul rilievo, chiamato *Score*.

1.4.3 Post-processing

La fase finale di *post processing* si occupa della scrematura di eventuali ridondanze. Se infatti più di una tripletta (e_i, e_j, e_k) dovesse dare come risultato uno stesso centro (o centri vicini entro limiti imposti) sarebbero solo un'ulteriore istanza della stessa ellisse ma con differenze nei calcoli dei parametri. Si procederebbe perciò ad una "fusione" delle ellissi con alto *Score* che presentino parametri simili entro determinate soglie in modo da dare il responso finale aumentando l'accuratezza.
Gli stessi autori dell'algoritmo presentano dei test su due dataset svolti per rilevare la dipendenza dalla risoluzione delle immagini e la velocità di rilevamento, uno

dei quali contenente immagini con una risoluzione fino a 1600x1200 ed uno invece con immagini 620x460. La differenza nei risultati dei due dataset è rilevante ma in entrambe i casi si parla di tempi molto piccoli e di molto al di sotto dei tempi di elaborazione di altri metodi sopracitati: per il primo dataset si hanno differenze dell'ordine minimo di 700ms e a salire (avg 60ms contro un minimo di 770ms) mentre per il secondo dataset differenze nell'ordine minimo dei 10ms (avg 1.5ms contro un minimo di 8ms). In *Figura 1.16* alcuni generici esempi di rilevamento di forme ellittiche ad opera dell'algoritmo descritto.

Figura 1.16: Esempi rilevamento ellissi con algoritmo Fornaciari-Prati

Capitolo 2

Set Up Sperimentale

Questo capitolo entra nel merito del lavoro svolto, in particolare andando a descrivere lo scenario hardware utilizzato e le scelte progettuali implementate. Ricordando lo scopo della tesi, quello che ci si propone di fare è creare un software di gestione di un sistema di visione, da integrare su un simulatore satellitare di volo in formazione, e che ha il compito di stimare la posa e l'assetto di un altro satellite target. Questo lavoro in particolare è creato per la gestione di satelliti cooperanti ma è un primo passo verso un sistema di visione per operazioni di prossimità fra satelliti non cooperanti. Si definiscono allora, per la trattazione a seguire, due soggetti:

- un satellite chiamato *inseguitore*, ovvero l'ipotetico simulatore su cui sarà integrato il sistema di visione;

- un satellite chiamato *target*, ovvero il simulatore di cui si vorrà conoscere una stima della posa e dell'assetto e su cui saranno applicati quindi i fiducial markers.

Nella pratica sperimentale l'unico effettivo simulatore utilizzato è quello che prende il ruolto di *target*, mentre il satellite *inseguitore* è impersonato dal sistema di riferimento solidale al solo sistema di visione. Il capitolo è così suddiviso: in una prima sezione viene descritto lo schema generale e costruttivo del simulatore sviluppato nel progetto "SPARTANS", nel seguito si presentano le componenti che effettivamente compongono il sistema di visione implementato ed infine si illustrano i fiducial markers creati e come sono stati applicati al *target*.

2.1 Formation Flight Hardware Simulator "SPARTANS"

Il progetto "SPARTANS" è sviluppato al CISAS (Center of Studies and Activities for Spaces) ed al DII (Dipartimento di Ingegneria Industriale) dell'Università degli Studi di Padova ad opera di studenti,dottorandi e ricercatori sotto la guida del professor Enrico Lorenzini. Lo scopo ultimo del progetto in piena operatività è quello di poter esaminare e validare in ambiente controllato a terra, strategie e algoritmi di controllo per manovre di prossimità fra satelliti in formazione di piccole dimensioni, manovre come rande-vouz e docking autonomi. Si tratta in pratica di implementare dei simulatori di micro satelliti in grado di mimare il più verosimilmente possibile l'interazione tra loro in un ambiente di microgravità, andando a testare eventuali algoritmi per la navigazione ed il controllo [11]-[12].

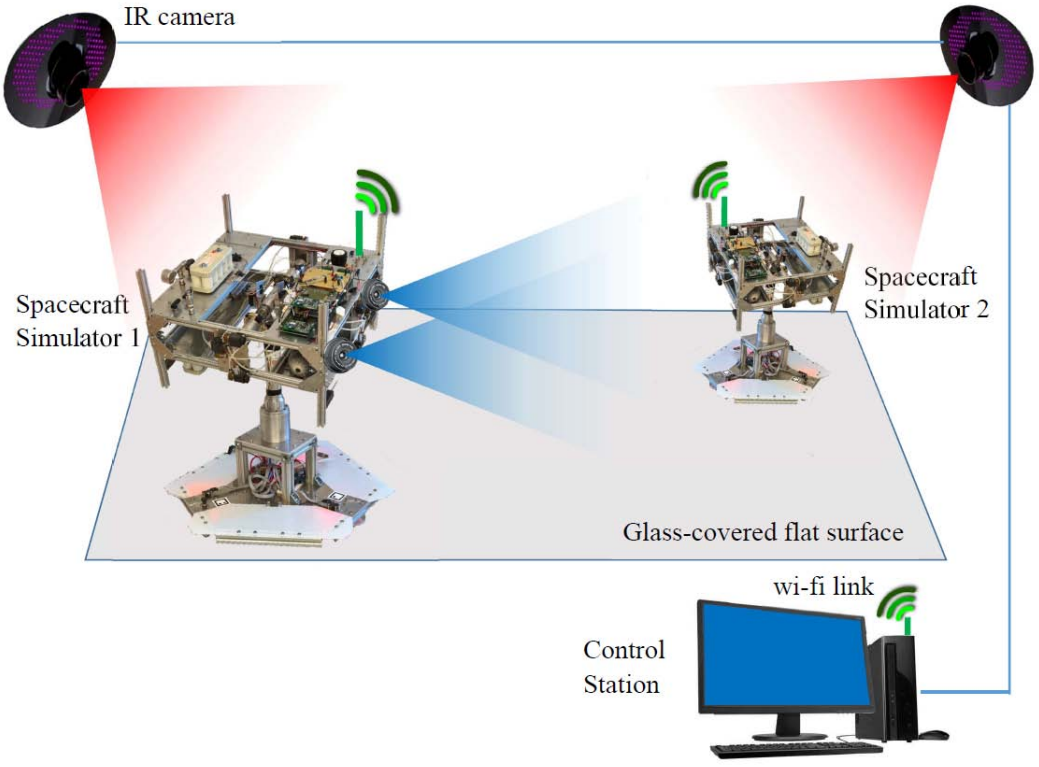


Figura 2.1: Schema generale progetto SPARTANS

In *Figura 2.1* è mostrato lo schema generale del simulatore hardware di terra, che consiste in:

- due o più simulatori di micro satelliti (allo stato attuale di sviluppo uno è completo e funzionante mentre uno è in fase di completamento) formati a loro volta da due componenti principali:
  - una parte superiore, il *Modulo d'Assetto*, con tre gradi di liberta rotazionali;
  - una parte inferiore, il *Modulo Traslazionale*, che fornisce due gradi di liberta traslazionali.
- un piano a basso attrito in vetro su cui i simulatori possono traslare;
- una *Control Station* per il controllo software dell'assieme;
- il *Global Navigation System*, ovvero un sistema di camere ad infrarossi per il tracciamento dei movimenti e la misurazione esterna dei micro satelliti.

In *Figura 2.2* come appare un micro satellite spoglio della protezione esterna, nei laboratori dell'Università di Padova e sopra al piano in vetro a basso attrito.

Figura 2.2: Il simulatore di un micro satellite "SPARTANS" nei laboratori dell'Università di Padova

2.1.1 Il Modulo Traslazionale

Il *modulo traslazionale* è la parte inferiore del simulatore del micro satellite ed ha due compiti importanti, quali

- sostenere il *modulo d'assetto*;
- permettere il moto traslazionale del micro satellite sul piano a basso attrito tramite dei cuscinetti d'aria.

Per svolgere i suoi compiti è composto a sua volta da cinque sottosistemi:

1. un sottosistema strutturale: consiste in una base formata da tre piatti di alluminio e rinforzata da elementi strutturali laterali anch'essi in alluminio. Su questa sono montati tre piatti in plexiglass che sostengono le bombole del sistema pneumatico e tre piccole basi in alluminio su cui sono montati i sensori. Si completa con una struttura centrale formata da altri elementi strutturali in alluminio a sostegno di un cilindro, su cui è posto il sostegno per il modulo d'assetto;
2. un sottosistema pneumatico: è costituito da 3 bombole d'aria compressa a $200bar$ di capacità $1l$ e il circuito pneumatico atto a convogliare l'aria per creare i cuscinetti necessari a far traslare l'intera struttura sul piano a basso attrito;
3. un sottosistema per la determinazione della posizione e dell'assetto: si occupa di determinare l'assetto della base in termini di orientazione rispetto alla verticale locale o la posizione in termini di di traslazione piana rispetto ad un sistema di riferimento scelto. É composta da un sensore ottico di flusso il cui output è rielaborato da un micro controllore a basso costo (Arduino UNO);
4. un sottosistema elettrico: fornisce energia ai componenti che ne necessitano ed è costituito da batterie ricaricabili;
5. un sottosistema per la gestione dei dati e delle comunicazioni: si occupa di trasportare i segnali di dati e controllo di navigazione tra il *modulo traslazionale* ed il *modulo d'assetto*.

2.1.2 Il Modulo d'Assetto

Il *modulo d'assetto* è la parte centrale del simulatore essendo in effetti questo a fare le veci di un reale micro satellite. É composto anch'esso da cinque sottosistemi:

1. un sottosistema strutturale;
2. un ADCS (Attitude Determination and Control Subsystem);
3. un sottosistema propulsivo;
4. un sottosistema elettrico;
5. un sottosistema per la gestione dei dati e delle comunicazioni.

Sottosistema strutturale

Il sottosistema strutturale del *modulo d'assetto* è di particolare importanza per il presente lavoro di tesi, in quanto le sue dimensioni e le sue caratteristiche determinano ed impongono vincoli e scelte progettuali.

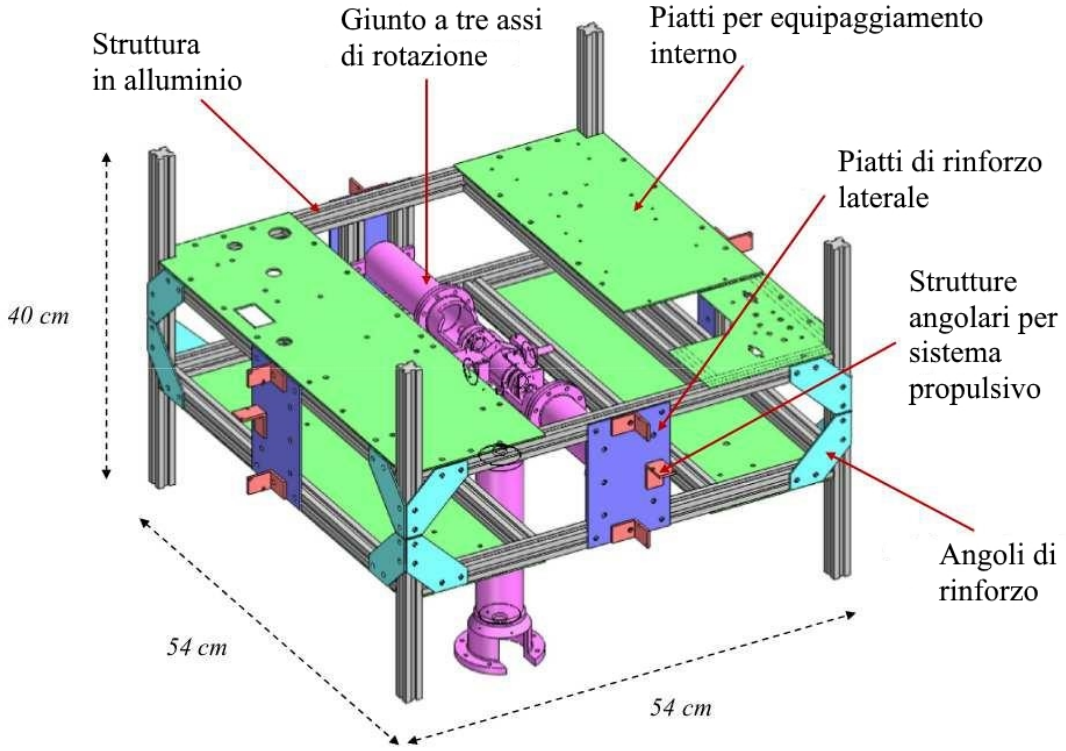


Figura 2.3: Sottosistema strutturale modulo d'assetto simulatore micro satellite

Come mostrato in *Figura 2.3*, il sottosistema è costituito da:

- un assieme di elementi strutturali di tipo trave, in alluminio, assemblati in modo da formare un parallelepipedo di misure $54cm \times 54cm \times 40cm$;

- quattro basi rettangolari in alluminio a sostegno dell'equipaggiamento di bordo, quindi il *payload*;
- vari angolari e piatti di rinforzo alle strutture;
- strutture angolari laterali su cui poter montare il sistema propulsivo;
- un sistema di giunti a tre assi e a basso attrito interno che permette la rotazione a tre gradi di libertà.

Riguardo quest'ultimo sottosistema, è necessario specificare i limiti angolari del giunto rotazionale a tre assi. Si considerino gli angoli di *Roll, Pitch e Yaw* come mostrati in *Figura 2.4*, ovvero con l'asse z verticale rispetto al piano a basso attrito e con semiasse positivo rivolto verso il basso. In tale convenzione, il *modulo d'assetto* è libero di ruotare senza vincoli attorno all'angolo di *Yaw*, mentre *Roll e Pitch* presentano una rotazione massima di $\pm 40 deg$ rispetto all'orizzontale parallela al piano a basso attrito.

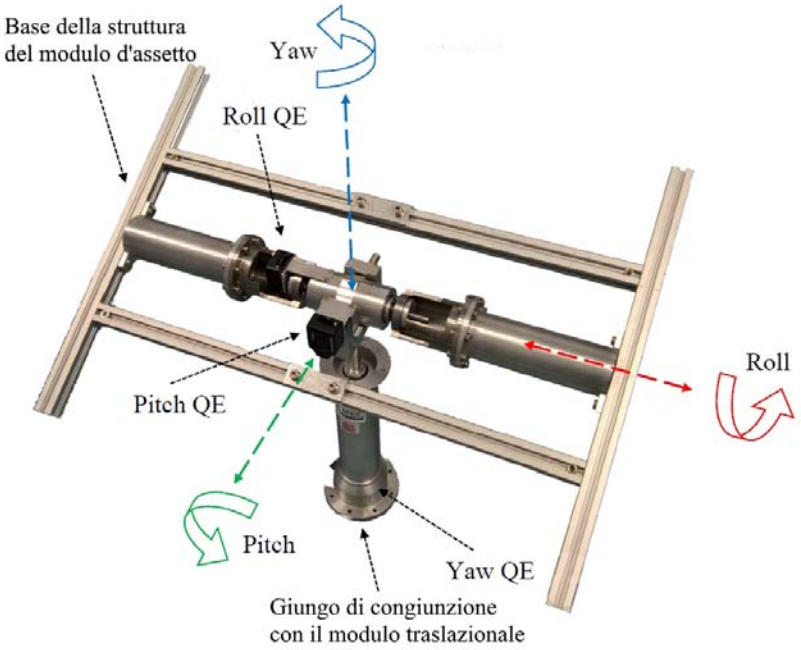


Figura 2.4: Giunto a tre assi sottosistema strutturale

Attitude Determination and Control Subsystem (ADCS)

L'ADCS ha il compito di determinare l'assetto e quindi l'orientazione del micro satellite ed eventualmente influire sul controllo per la sua stabilizzazione. Esso è

composto da due parti, i sensori di bordo e il sottosistema elettronico di bordo per l'elaborazione dei dati forniti dai sensori. Ogni micro satellite è fornito da due tipi di sensori:

- tre encoder incrementali montati sui giunti rotazionali del sottosistema strutturale che riescono con accuratezza a fornire lo spostamento angolare nelle tre direzioni di *Roll, Pitch e Yaw*. Li si possono individuare, nominati "QE", in *Figura 2.4*;
- una IMU (Inertial Measurement Unit) in grado di rilevare l'assetto rispetto ad una terna di riferimento esterna prescelta.

I dati forniti dai sensori vengono infine gestiti da un computer di bordo in grado di immagazzinarli ed elaborarli per fornire un controllo sulla navigazione attraverso un'interfaccia di collegamento con il sistema propulsivo.

Sottosistema propulsivo

Il sottosistema propulsivo è usato per fornire coppie di controllo sia per quanto riguarda l'assetto rotazionale che la traslazione sul piano a basso attrito. Per fornire le coppie si è implementato un sistema ad aria compressa suddiviso in due sezioni, una di alta pressione ed una di bassa pressione e controllato tramite tecnica PWM (Pulse Width Modulation).
Il primo è composto da una bombola di capacità $4l$ contenente aria pressurizzata alla pressione di $200bar$ fornita di due valvole per il sistema di ricarica e sfiato e attraverso le quali è in collegamento con il resto del circuito propulsivo. Fa parte della sezione ad alta pressione anche un regolatore di pressione che ha il compito di gestire l'aria compressa in arrivo a $200bar$ per restituirla alla più bassa pressione di utilizzo da parte degli ugelli ossia $10bar$.
La sezione a bassa pressione è formata dal circuito di collegamento che conduce l'aria e dai thrusters che forniscono la spinta. In particolare vi sono 12 thrusters per fornire coppie di controllo rotazionali e 8 thruster per fornire forze di controllo traslazionale. Ogni thrusters è formato da un'elettrovalvola che determina o meno lo scarico dell'aria e da un ugello.

Sottosistema elettrico

Come per il sottosistema elettrico del *modulo traslazionale*, il sottosistema elettrico del *modulo d'assetto* si occupa di fornire energia ai componenti che ne richiedono

ed è formato da due batterie a Litio-Polimero da 12 volt in serie ad un convertitore DC/DC atto a gestire le diverse richieste in termini di differenza di potenziale da parte dei diversi componenti dell'equipaggiamento, come sensori o valvole.

Sottosistema per la gestione dei dati e delle comunicazioni

Il sottosistema per la gestione delle comunicazioni del *modulo d'assetto*, oltre a permettere il passaggio di dati e segnali tra i vari componenti del micro satellite è configurato per altri due tipi di comunicazioni:

- una comunicazione *satellite-satellite* attraverso la quale le unità della formazione possono comunicare tra loro in modo da simulare un eventuale scenario di cooperazione in navigazione;

- una comunicazione *satellite-laptop* che permette di interfacciare un micro satellite a computer in modo da poter configurare l'hardware per i differenti test.

Allo stato attuale, di due unità solo una è completa e funzionante, perciò solo il secondo tipo di comunicazione è utilizzato.

2.1.3 Il Global Navigation System

Il *GNS* (*Global Navigation System*) è un sistema implementato per poter conoscere la posizione e l'assetto dei *moduli d'assetto* dei micro satelliti in navigazione sul piano a basso attrito. Esso è stato implementato in primo luogo per ottenere misurazioni precise nei test effettuati ma soprattutto per avere un confronto con i dati di stima della posizione e dell'assetto ottenuti dai sensori presenti sul carico utile dei micro satelliti, potendone così testare la bontà e la precisione. Il GNS è composto da tre macro componenti:

- 6 telecamere ad infrarossi poste ad altezza soffitto su tutto il perimetro del piano a basso attrito in modo da poter coprire l'intera regione possibile di navigazione simulata;

- dei marker strutturali riflettenti posti sul sottosistema strutturale del modulo d'assetto dei micro satelliti, necessari a far individuare e inseguire la terna ad essi solidale alle camere;

- un computer tramite il quale si elaborano i dati provenienti dalle camere per poter estrapolare i dati desiderati su posa e assetto della terna inseguita attraverso i markers riflettenti.

In particolare sono utilizzati tre markers riflettenti posti sulla parte superiore della parte strutturale dei micro satelliti, uno dei quali individua il centro di una terna solidale al modulo d'assetto e gli altri due ne individuano due degli assi, mentre il terzo asse è ottenuto per derivazione tramite prodotto vettoriale. Il GNS campiona ad una frequenza di $50Hz$ ed è usato per comparare i dati ottenuti da altri dispositivi poichè vanta un'ottima accuratezza di misura.

2.2 La stereocamera

Fa parte dei sensori che compongono l'ADCS dei *moduli d'assetto* dei micro satelliti anche la stereocamera, attraverso la quale infatti in questo lavoro ci si propone di far stimare al sistema di controllo di un satellite *inseguitore* la posizione e l'assetto relativi di un satellite *target* in formazione. Questo breve paragrafo descrive le caratteristiche essenziali della stereocamera integrata ed utilizzata nel set up sperimentale.

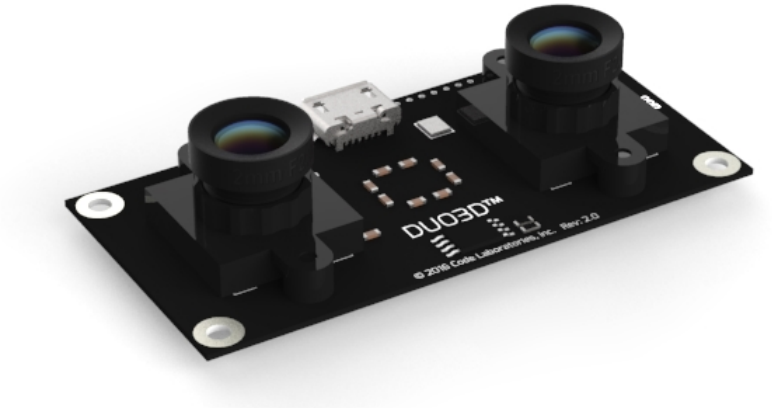


Figura 2.5: La stereocamera DUO M

La stereocamera è una DUO M (*Figura 2.5*) ossia un sensore ottico ultra compatto e configurabile interfacciandosi attraverso uno standard USB. I suoi punti di forza sono l'alta velocità e le dimensioni ridotte che la rendono ideale per l'implementazione sul simulatore di un satellite dalle dimensioni ridotte come le unità del progetto "SPARTANS". Le sue dimensioni sono infatti quelle in *Figura 2.6* ed ha un peso di

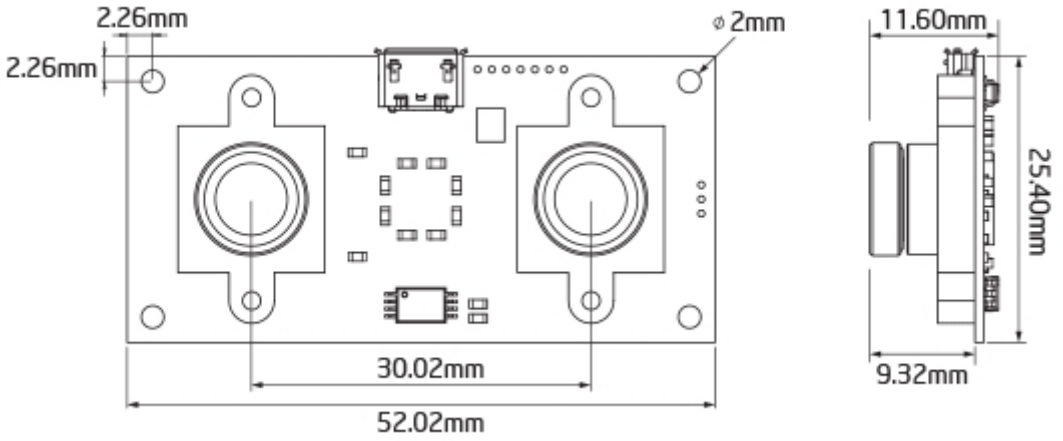


Figura 2.6: Dimensioni della stereocamera DUO M

appena $6.5g$.
Di particolare importanza si fa notare una *baseline* di $3cm$, dato rilevante per la ricostruzione dell'ambiente antistante la stereocamera (*Sezione 1.2.7*). Possiede un ampio campo di vista, di $170deg$, il che è si un vantaggio, in quanto permette di coprire visivamente una più vasta regione, ma anche uno svantaggio poichè un ampio angolo di vista fa percepire le distanze in modo diverso, in particolare fa sembrare gli oggetti più lontani di quello che sono. Questo non influisce sui processi di stima della profondità da parte dell'elaboratore, ma può influire su l'elaborazione d'immagine quando si è alla ricerca di determinate caratteristiche che possono rendersi meno riconoscibili. Per quanto riguarda la risoluzione la DUO M dispone di svariate configurazioni possibili fino ad un massimo di 752×480. Quella utilizzata nel presente set up è 640×480 e ad una frequenza di campionamento di $10Hz$ ovvero a $10fps$ (frames per secondo). Altro vantaggio del kit DUO è che permette una discreta gamma di combinazioni in termini di configurazioni, tra le quali la possibilità di estrapolare i video già rettificati dall'effetto di distorsione radiale che l'ampio campo di vista comporta, e di poter disporre inoltre di un dato di fabbrica noto quale la *matrice di calibrazione* caratteristica, ulteriore dato rilevante nel processo di ricostruzione.

2.3 Descrizione e design dei markers

Si descrive in questo paragrafo lo sviluppo dei markers grafici caratteristici per questa applicazione, il primo step che nasce effettivamente con questo lavoro.

In *Sezione 1.3* si è descritto cosa sono i fiducial markers, la loro utilità e il pattern macroscopico che si è deciso di utilizzare, ovvero le forme circolari. In *Sezione 1.4* si esamina poi l'algoritmo di riconoscimento scelto per il riscontro delle stesse.
Ricapitolando, l'idea è quella di utilizzare i markers per individuare dei punti strutturali precisi nel micro satellite *target*, attraverso i quali permettere ad un algoritmo di stimarne la posa e l'assetto relativi. Quello che a questo punto rimane da decidere è:

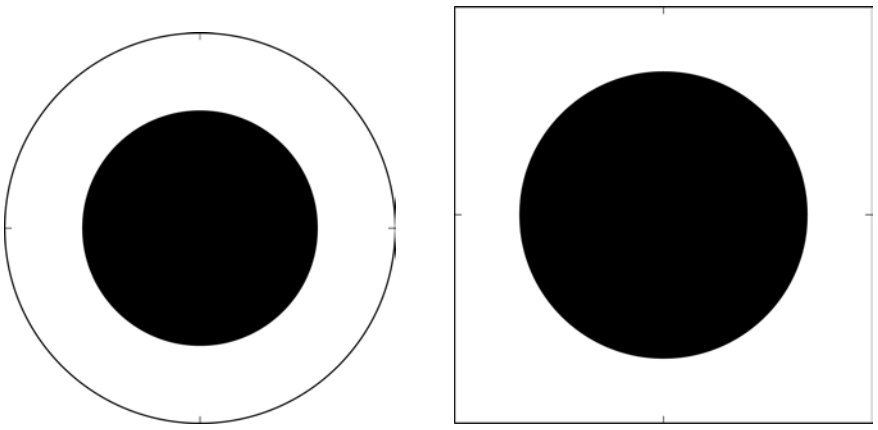
1. il numero di quanti diversi markers necessari allo scopo progettare;
2. le features dei markers che ne permettono l'individuazione nell'immagine e il riconoscimento univoco.

In *Sezione 1.3* sono esplicitati i requisiti che hanno portato alla scelta della forma di massima decisa per i markers. Per trovare soluzione al primo dei due problemi sopra elencati, si devono introdurre altri requisiti, specifici per l'utilizzo che si intende fare dei markers in questo lavoro di tesi, ossia l'applicazione su di un satellite *target*:

- nell'ottica di ottimizzazione della relazione costo-computazionale/velocità-di-riconoscimento, si vuole che vi siano da riconoscere il minor numero possibile di markers in ogni immagine;
- si desidera in ogni istante poter calcolare la posa e l'assetto del *target*, si deve perciò avere da ogni angolazione la possibilità di vedere un numero si minimo di markers, ma sufficiente a tale calcolo.

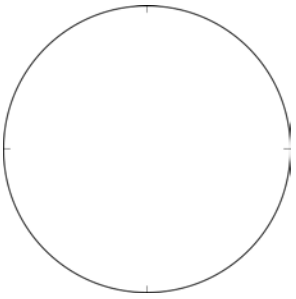
La forma del *modulo d'assetto* del micro satellite è un parallelepipedo a base quadrata (*Sezione 2.1.2*). Ciò implica che nel peggiore dei casi, ossia quello in cui nell'immagine della stereocamera che inquadra il *target* via sia il minor numero di informazioni, si ha in vista un'unica faccia del micro satellite e da questa si vuole in ogni caso poter stimare posa e assetto. Rispettando allora i due requisiti imposti si decide di utilizzare *tre* markers distinti. Tre perchè è il numero di punti necessario ad individuare un piano nello spazio, piano su cui giace la faccia in vista; meno punti danno una singolarità e più punti danno una ridondanza essendo necessariamente complanari agli altri tre.
Sono state poi esaminate svariate diverse combinazioni per le famiglie di markers e molteplici metodi per il riconoscimento univoco delle stesse, alla ricerca del matching che garantisse un buon compromesso tra velocità e robustezza di rilevamento. Si giunge così a quella che è la configurazione definitiva per questo primo lavoro.

Ai tre markers è stato affidato un *tipo*, ossia la caratteristica propria del marker che sta ad individuare quella che è la famiglia di appartenenza, in sostanza il suo identificativo. In particolare sono chiamati rispettivamente marker α, β, γ e sono presentati in *Figura 2.7*.



(a) Marker *tipo* α

(b) Marker *tipo* β



(c) Marker *tipo* γ

Figura 2.7: Features dei markers

Si nota la scelta di features volutamente semplici. Nel seguito si denotano quali sono allora le caratteristiche che permettono il riconoscimento univoco di ciascun *tipo* di marker:

- il marker di *tipo* α viene individuato ricercando le coppie di circonferenze/ellissi concentriche con un determinato rapporto tra i semiassi, rapporto che tra essi rimane fisso anche in caso di particolari angolazioni, verificando inoltre che la circonferenza/ellisse più interna sia totalmente nera;

- il *tipo* β è selezionato come circonferenza/ellisse unica che rispetta un determinato valore di soglia luminosa impostata, in particolare dovrà essere totalmente nera;

- il *tipo* γ viene riconosciuto in modo analogo al β, ma verificando che sia una forma ellittica unica totalmente bianca;

Una volta decisi e fabbricati i markers si procede all'applicazione sul modulo d'assetto del micro satellite. Per fare questo vengono aggiunti al micro satellite dei pannelli totalmente *neri* di plastica leggera e ultra fina, che oltre a fungere da sostegno per i markers svolgono un compito di protezione per i componenti interni del modulo d'assetto. Si sceglie di applicare i markers alle sole facce laterali del simulatore in quanto per i vincoli rotazionali d'angolazione del giunto a tre assi (*Sezione 2.1.2*) la superficie superiore ed inferiore non possono comunque essere in vista, avendo i micro satelliti il centro di massa obbligatoriamente alla stessa altezza, traslando nella sola direzione planare sul piano a basso attrito.
I markers applicati sono quindi 12, tre per ognuna delle quattro facce laterali del modulo d'assetto, ed essenso solo 3 le famiglie di markers, per riconoscere le facce in vista vengono utilizzate tutte le combinazioni di posizioni reciproche riscontrabili nell'immagine ed un sistema di tracciamento dei markers già riconosciuti. I dettagli del metodo di riconoscimento vengono descritti in *Sezione 3.3.2*.
In *Figura 2.9* è mostrato come appaiono allora le quattro facce laterali del modulo con la configurazione scelta ed in *Figura 2.8* come appare il modulo d'assetto di un'unità vista dalla stereocamera.

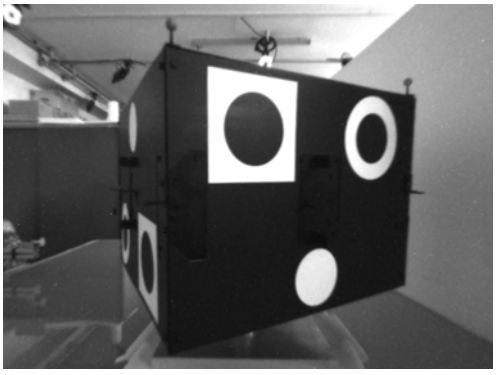
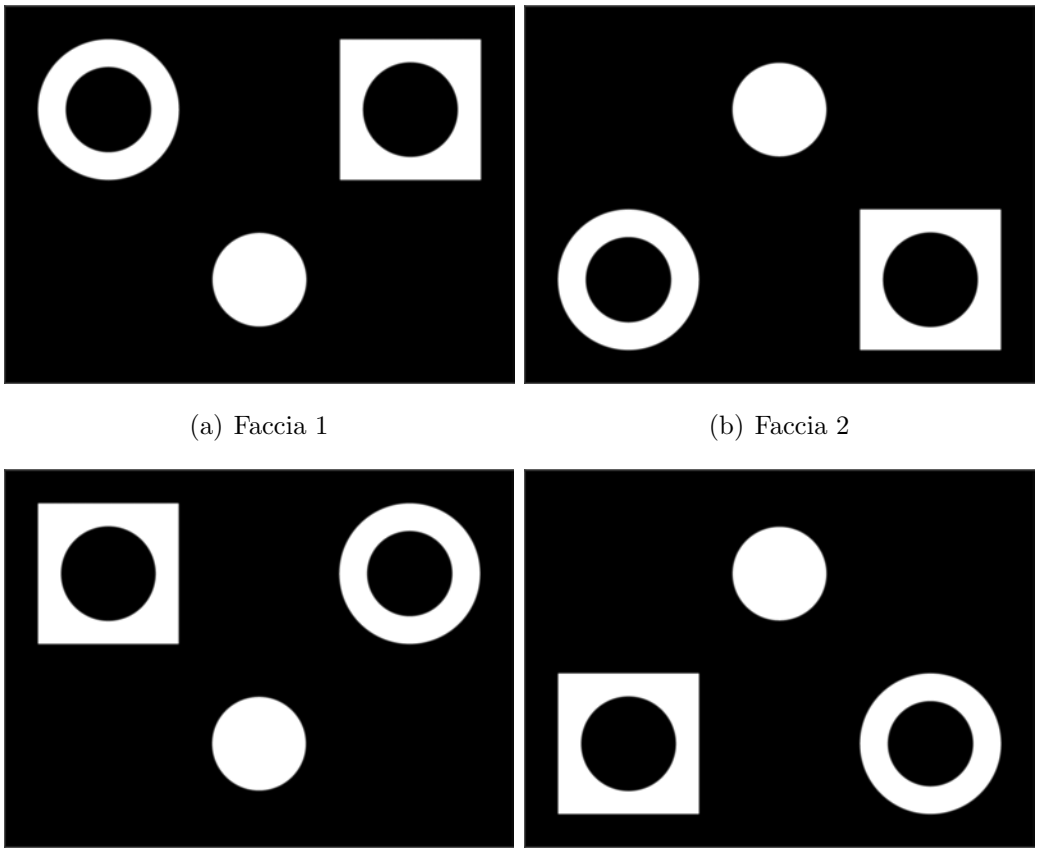


Figura 2.8: Il modulo d'assetto di *target* visto da uno dei canali della stereocamera

2.4 I sistemi di riferimento

Questo ultimo breve paragrafo del capitolo è necessario per descrivere i sistemi di riferimento scelti per il set up e che verranno dati per noti in tutto il resto della trattazione. In particolare avremo:



(a) Faccia 1 (b) Faccia 2

(c) Faccia 3 (d) Faccia 4

Figura 2.9: Configurazione dei markers sulle facce del modulo d'assetto

- il *Sistema di riferimento Camera*: è la terna ortonormale la cui origine combacia con il punto focale della camera sinistra della stereocamera e con gli assi così disposti:
  - X: parallelo al piano di traslazione ovvero al piano a basso attrito e con semiasse positivo crescente verso destra;
  - Y: asse verticale, ossia perpendicolare al piano di traslazione e con semiasse positivo rivolto verso il basso;
  - Z: asse parallelo al piano e con semiasse positivo nel senso della profondità.
- il *Sistema di riferimento Target*: è la terna ortonormale solidale al modulo d'assetto del micro satellite *target*, la cui origine coincide con il centro di massa e i cui assi sono così disposti, considerando di avere in vista frontale la *faccia 1*:

  - X: parallelo al piano di traslazione ovvero al piano a basso attrito e con semiasse positivo crescente verso destra;
  - Y: asse parallelo al piano e con semiasse positivo nel senso negativo della profondità, ovvero rivolto dal centro di massa verso la *faccia 1*;
  - Z: asse verticale, ossia perpendicolare al piano di traslazione e con semiasse positivo rivolto verso il basso.

- il *Sistema di riferimento Immagine*: è il riferimento cartesiano bidimensionale a cui ci si riferisce parlando delle immagini intese coma array bidimensionali, la cui terna è posta nell'angolo sinistro superiore e i cui assi sono così disposti:
  - X: orizzontale con semiasse positivo crescente verso destra;
  - Y: verticale con semiasse positivo verso il basso.

Capitolo 3

Struttura del Software

3.1 Ambiente di sviluppo

Questo capitolo entra nel vivo dello sviluppo di questo lavoro, andando a descrivere nel dettaglio com'è stato pensato e strutturato il software che si occupa di gestire il sistema di visione, ovvero di elaborare gli output della stereocamera per poter restituire la posa e l'assetto del satellite target nel contesto del volo in formazione. É necessario cominciare specificando qual è stato l'ambiente di sviluppo utilizzato. Per giustificare la scelta, bisogna precisare che il sistema che permette di interfacciarsi al simulatore attraverso il personal computer, è il ROS, *Robot Operating System*, ovvero un insieme di framework che fornisce le stesse funzioni di un sistema operativo contenente librerie, tools, driver di dispositivi, astrazioni hardware e molto altro, sviluppato apposta per interfacciarsi ad implementazioni robotiche. Tale sistema è per la gran parte sviluppato in C++ e questo spiega la scelta di sviluppare in tale linguaggio l'intero programma. C++ è un linguaggio di programmazione orientato ad oggetti [6] e sono descritte infatti in questo capitolo le classi sviluppate per l'obbiettivo prefissato. In aggiunta a tutte le librerie standard del C++ sono poi integrate un set particolare di librerie che per il nostro lavoro hanno particolare importanza, ovvero le librerie OpenCV. Queste sono un software libero di cui esistono svariati tutorial [7] e un'ampia documentazione a riguardo, contenente una vasta raccolta di librerie sviluppate principalmente in C++ (ma in realtà utilizzabili anche su diverse piattaforme come C, Python o Java) con metodi appositamente sviluppati per l'ambito dell'elaborazione d'immagine e la visione artificiale. Tutto questo è atto a semplificare al possibile il lavoro del calcolatore e dello sviluppatore, ma se pur potrebbe sembrare ovvio si precisa che la scelta del

linguaggio non fa perdere di generalità agli algoritmi implementati, poichè a meno di strutture e sintassi, possono essere sviluppati in qualunque altro linguaggio. A supporto di questo, si specifica che il software è stato pensato come *portable*, ovvero totalmente indipendente dal software di controllo generale del simulatore, il che lo rende ancora più generale. Ed è stato pensato, progettato ed implementato in un linguaggio orientato ad oggetti proprio perchè in questo modo, per poterlo poi integrare e gestire all'interno del paylaod, basta importare le classi implementate nelle librerie del sistema.

3.2 Struttura Generale

Si evince anche dallo schema a blocchi in *Figura 3.1* che la struttura generale del software è relativamente semplice e non eccessivamente ramificata. Come accennato parlando del sistema di visione nel *Capitolo 1*, senza pretendere di ricalcare un sistema tanto complesso quanto quello del sistema di visione umano, si è pensato il processo ispirandosi ad un modello simile. L'input della stereocamera, i nostri "occhi", viene perciò diviso in due canali clone, uno per la camera di sinistra ed uno per la camera di destra. I due canali si occupano di analizzare singolarmente le due immagini dello stesso istante provenienti da due punti di vista differenti, e una volta tratte le informazioni ricercate sulla presenza ed il tipo di eventuali markers, si fondono in un unico canale che rielabora le informazioni comuni per la stima di posa e assetto.
Il software è quindi strutturato da due macro classi principali:

1. classe *SingleCamera*

2. classe *Stereocamera*

completate da un corpo centrale, il *main*. A queste vanno poi integrate le classi dell'algoritmo Fornaciari-Prati per il riconoscimento dei bordi ellittici, ma che sono trattate marginalmente, avendo già spiegato nel dettaglio il processo di riconoscimento in *Sezione 1.4*, ed essendo in gran parte già implementate, ed usate come librerie preesistenti, esattamente come facessero parte dei metodi di riconoscimento bordi e forme poligonali già presenti nelle librerie OpenCV.
La classe *SingleCamera* è l'oggetto che si occupa di modellizzare una singola videocamera e si occupa di:

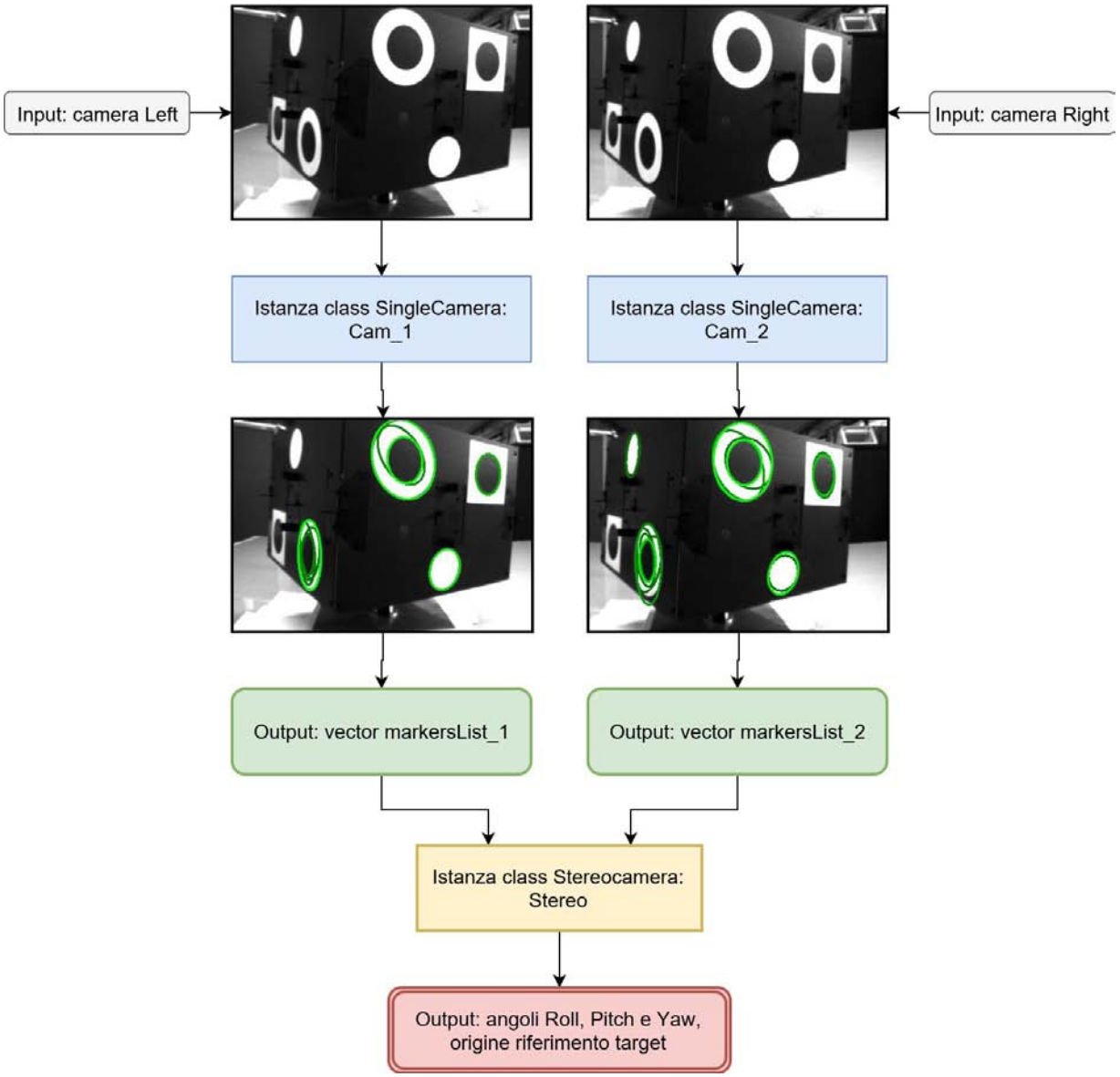


Figura 3.1: Struttura generale macroscopica del software

- elaborare il frame dato in ingresso, proveniente da uno dei due canali della stereocamera;
- ricercare le forme ellittiche al suo interno;
- analizzare le ellissi trovate alla ricerca delle features per il riconoscimento dei markers ed il loro *tipo*;
- rintracciare la faccia di appartenenza dei markers riconosciuti.

Sono instanziati perciò due oggetti *SingleCamera*, uno per ognuno dei due canali della stereocamera. Questi vengono poi dati in ingresso ad una singola istanza *Stereocamera*, che si occupa di:

- rielaborare i due oggetti *SingleCamera* alla ricerca di riscontri comuni nei markers riconosciuti da entrambe le camere della stereocamera;
- di questi ne calcola la posizione spaziale rispetto al riferimento *camera*;
- stima la posa relativa del centro di massa del *target*;
- stima l'assetto relativo del *target*;

Le classi sono state così strutturate con l'obiettivo di rendere l'intero processo di calcolo il più rapido possibile e quindi il più leggero possibile in termini di costi computazionali, ricercando l'equilibrio tra le mansioni da affidare ad una *SingleCamera* e quelle da affidare al canale unito della *Stereocamera*. Cercando di spiegare meglio questo concetto, i compiti affidati alla *SingleCamera* hanno un carico di dati inferiore, ma essendo due le istanze, ognuna per le due camere, tali compiti sono computazionalmente doppi. I compiti della stereocamera invece sono processati una singola volta ma dovendo gestire una mole di dati più ampia, rischiano di essere più onerosi. Le sottosezioni di questo paragrafo descrivono nel dettaglio ognuna delle mansioni delle due classi.

In un ottica di futura integrazione delle classi create nel sistema generale di gestione del simulatore, il corpo centrale del software creato, il *main*, è lasciato il più scarno possibile, in modo da non avere eccessive dipendenze o processi all'esterno delle classi stesse. Nella pratica, attualmente si occupa di:

- gestire l'iterazione frame per frame dei video processati;
- gestire le interazioni tra le classi;
- gestire la scala temporale dei processi.

3.3 Classe *SingleCamera*

Questa sezione spiega nel dettaglio cosa avviene all'interno di ognuna delle istanze della classe *SingleCamera*. Come già esposto essa si occupa di analizzare l'immagine in ingresso proveniente da una delle due camere della stereocamera, alla ricerca delle

forme ellittiche tra le quali ricercare le features che permettono il riconoscimento dei *fiducial markers*. A tale scopo è innanzitutto dichiarato un nuovo tipo di dato attraverso l'implementazione di una struttura definita come *Marker*.

Un oggetto di tipo *Marker* conterrà tra le sue caratteristiche membro:

- le coordinate immagine del suo centro (x_c, y_c) rispetto al riferimento *Immagine*;
- il valore in pixel dei suoi semiassi A e B, rispettivamente il maggiore ed il minore;
- l'angolo di rotazione ρ del semiasse maggiore rispetto al semiasse positivo delle ascisse;
- il *tipo* del marker (*Sezione 2.3*);
- un valore *idFace* contenente la faccia di appartenenza;
- il valore d'Irradianza medio dei pixel contenuti al suo interno.

Lo schema in *Figura 3.2* aiuta a capire il processo sequenziale che avviene per un oggetto *SingleCamera*. L'immagine data in ingresso come parametro, viene in ordine:

1. fatta elaborare dal metodo *ellipseDetect()* che si occupa tramite l'algoritmo Fornaciari-Prati di scovare tutte le forme ellittiche presenti nell'immagine;
2. queste vengono analizzate dal *markersDetect()*, che si preoccupa invece di scovare tra tutte le ellissi le features dei markers, scartando i falsi positivi e quindi dichiarando ed inizializzando gli oggetti di tipo *Marker*;
3. questi vengono infine elaborati da *structuresRecognizing()* che ha il compito di assegnare loro la faccia di appartenenza.

I markers così riconosciuti e completi di tutte le informazioni necessarie vengono stivati in un vettore membro che è in seguito utilizzato come parametro dall'oggetto *Stereocamera*.

3.3.1 ellipseDetect()

Il primo passaggio alla ricerca dei *fiducial markers* è come spiegato quello di andare alla ricerca delle forme ellittiche all'interno dell'immagine in ingresso. L'immagine presentata come esempio in *Figura 3.3* è già in bianco e nero ma va precisato

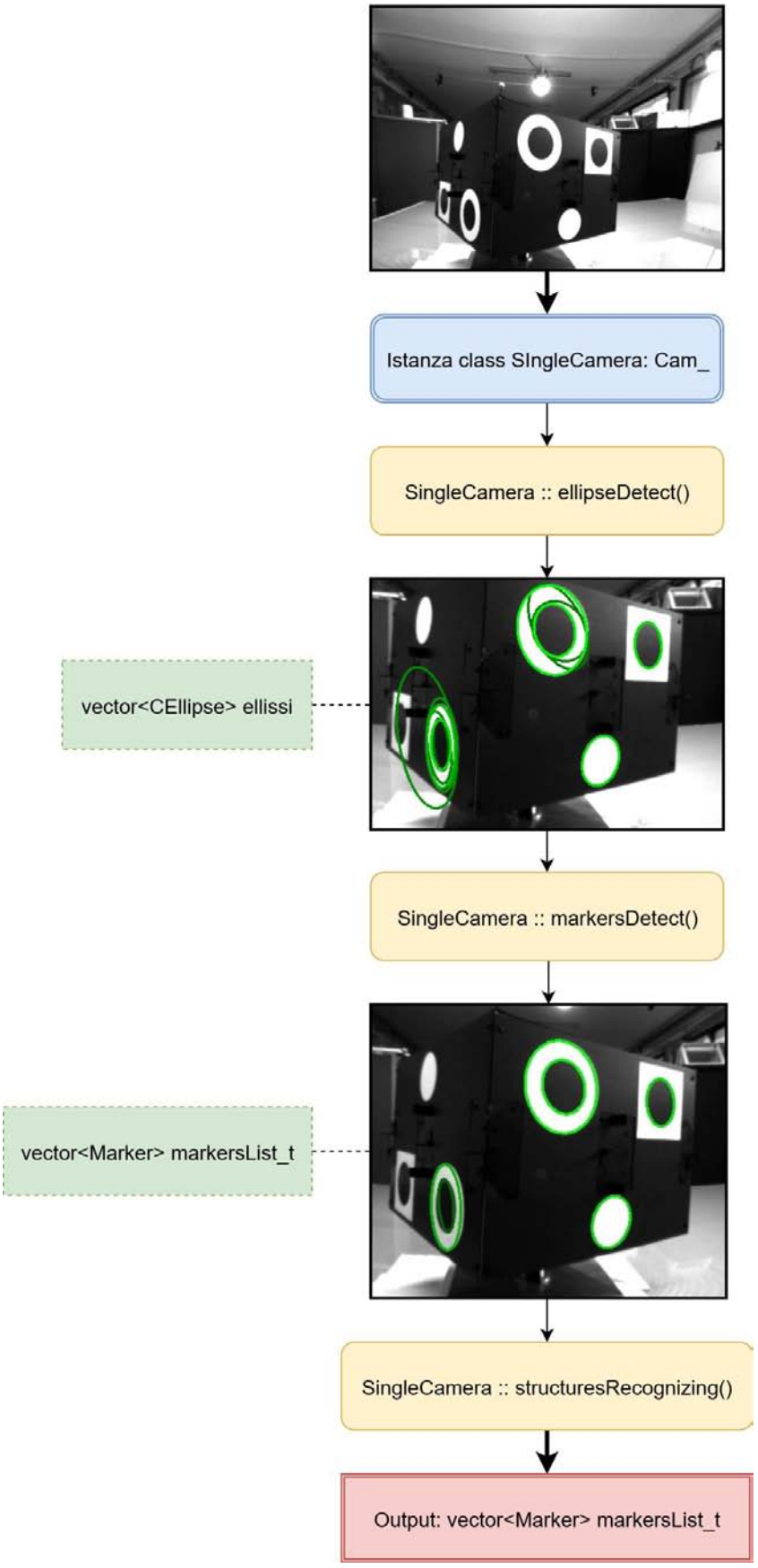


Figura 3.2: Struttura macroscopica della classe *SingleCamera*

che l'immagine in ingresso è data genericamente a colori o comunque come dato contenente le informazioni delle tre matrici d'Irradianza, una per ognuno dei tre colori primari RGB. É poi l'algoritmo a trasformarla in scala di grigi riducendo l'immagine ad una sola matrice d'Irradianza.

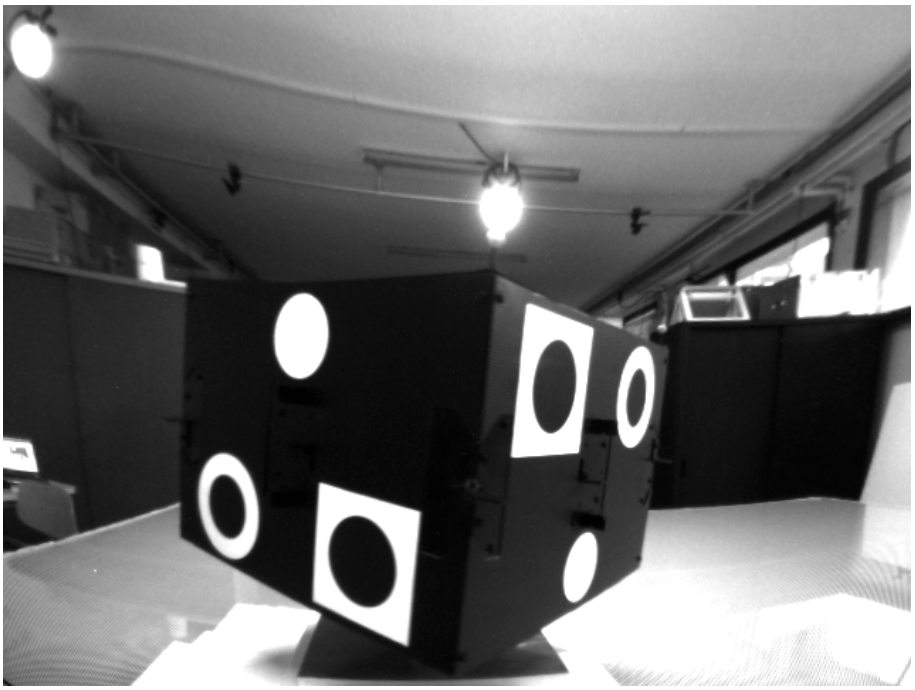


Figura 3.3: Esempio di frame in ingresso ad una *SingleCamera*

Il processo di ricerca e selezione delle forme ellittiche tramite l'algoritmo Fornaciari-Prati è già stato illustrato in *sezione 1.4* quindi non rientriamo nei dettagli del processo. É però importante parlare di un nuovo tipo di dato implementato, ovvero la struttura *CEllipse*, struttura generata per creare istanza fisica di ogni ellisse riscontrata nell'immagine, concettualmente simile alla struttura di tipo *Marker* ma utilizzata in ambito differente. Un oggetto di tipo *CEllipse* è costituito da

- x_c e y_c le coordinate cartesiane in riferimento *immagine*;
- A e B rispettivamente la lunghezza dei semiassi maggiore e minore dell'ellisse, in pixel;
- l'angolo ρ di rotazione tra il semiasse maggiore A ed il semiasse positivo delle ascisse;
- un valore chiamato *score* che indica la bontà dell'ellisse trovata ovvero l'accuratezza del rilievo;

- un'ulteriore valore chiamato *rel* atto ad indicare una sorta di probabilità che l'ellisse trovata esista effettivamente sull'immagine e calcolato in pratica sulla base di quante ridondanze sono state fuse per ottenere una singola ellisse.

I risultati ottenuti sono del genere di quelli presentati in *Figura 3.4*

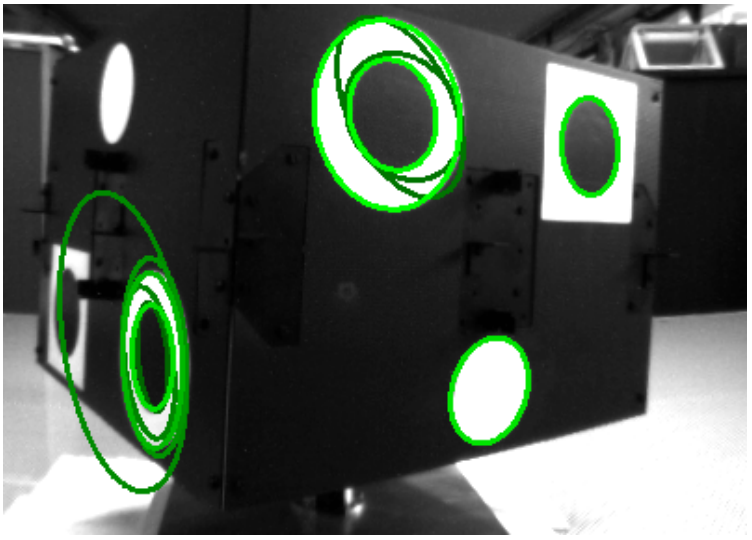


Figura 3.4: Esempio (1) di rilevamento ellissi tramite *ellipseDetect()*

Si può notare che il rilevamento può essere efficace, ma che questo non esclude la presenza di svariati falsi positivi. Questi possono essere ricondotti a diversi fattori come:

- risoluzione non ottimale;
- rumore d'immagine;
- soglie di precisione impostate nell'algoritmo di ricerca;
- ombre o illuminazioni sfavorevoli,

tutti fattori che nell'insieme possono "imbrogliare" l'algoritmo di selezione. Vi sono all'interno dell'algoritmo dei parametri per gestire questo marginalmente, ad esempio scegliendo il valore di *threshold* luminosa oppure impostando un valore minimo di *score* o di *rel* sotto il quale le ellissi verrebbero scartate. Ma restringere eccessivamente questi valori può portare a non riconoscere più in diversi casi nemmeno le ellissi effettive, per cui si cerca un compromesso funzionale lasciando ai metodi successivi il compito di setacciare ed individuare tra tutte le ellissi solo le features reali.

Per ogni ellisse riscontrata viene allora dichiarato ed inizializzato un oggetto *CEllipse* che viene stivato in un vettore di dati *CEllipse* il quale viene poi analizzato dal metodo *markersDetect()* per la ricerca delle features e l'inizializzazione dei markers.

3.3.2 markersDetect()

Il *markersDetect()* è uno dei metodi centrali del processo ed ha il compito di ricercare e selezionare i markers tra le ellissi trovate. Va precisato che il metodo si preoccupa di individuare nell'immagine la presenza dell'oggetto *Marker* individuale, senza utilizzare nessun tipo di correlazione tra i diversi markers o ricercando in determinate aree, cerca la semplice esistenza della singola feature e inizializza il marker. Sono metodi successivi a preoccuparsi di assegnare ad ogni singolo *Marker* trovato la sua faccia di appartenenza. L'utilizzo di informazioni note, essendo il lavoro pensato per satelliti in formazione e collaborativi, potrebbe essere implementato e potrebbe facilitare alcuni passaggi di ricerca e riconoscimento, ma si preferisce lasciare i passaggi separati e sequenziali in modo da aumentare la probabilità che le ellissi e successivamente i markers che superano tutte le selezioni ed i controlli, siano effettivamente i markers applicati sul target e non falsi positivi che possono far divergere i calcoli successivi.
La selezione inizia con la ricerca dei marker di *tipo* α (*Figura 3.5*), andando quindi alla ricerca di tutte le coppie di ellissi concentriche il cui rapporto tra i semiassi rispecchi la proporzione 5 : 3 e in cui l'ellisse interna rispecchi il fatto di essere totalmente nera (ovvero Irradianza media vicina a 0).

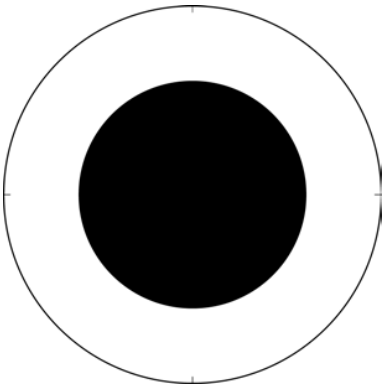


Figura 3.5: Marker di *tipo* α

La selezione delle caratteristiche appena citate è governata da dei parametri di precisione, ovvero delle soglie accettabili e settabili entro le quali ad esempio considerare la concentricità, attualmente entro un raggio di 4 pixel, la tolleranza al rapporto

5 : 3, settata su un ± 0.2, e non da meno entro che soglia ritenere l'intensità d'Irradianza media delle ellissi accettabilmente vicina al nero totale o al bianco assoluto: dopo diversi test queste sono settate su valori inferiori a 90 per il primo e superiori a 235 per il secondo. Il calcolo di quest'ultimo parametro, l'Irradianza media delle ellissi, è ottenuto tramite un sotto metodo sfruttato da diversi metodi principali, chiamato *getEllipseIntensity()* e che ottiene l'informazione iterarando sui pixel di un riquadro contenente l'ellisse alla ricerca di quali rientrino nella regione interna dell'ellisse stessa, soddisfando l'equazione analitica della conica rototraslata. Si fa quindi la media del valore d'Irradianza dei pixel, presi in coordinate (colonna, riga) corrispondenti alle coordinate (x, y) sul piano immagine, che soddisfano l'equazione

$$\frac{[(x - x_c)cos(\rho) + (y - y_c)sin(\rho)]^2}{A^2} + \frac{[(y - y_c)cos(\rho) - (x - x_c)sin(\rho)]^2}{B^2} < 1 \quad (3.1)$$

con x_c, y_c coordinate del centro dell'ellisse ed A e B rispettivamente i suoi semiassi maggiore e minore.
Come si può notare dall'esempio di rilevamento delle ellissi da parte di *ellipseDetect()* in *Figura 3.4*, la maggior parte dei falsi positivi risiedono proprio all'interno della feature del marker α, ed è per questo che il primo sbarramento alla ricerca dei marker degli altri due tipi, β e γ, è che non si trovino all'interno dell'area dei marker di *tipo* α già rilevati. Così facendo si andranno automaticamente a scartare tutte le corrispondenze date da rumore o errori di rilievo. A questo punto tra le restanti ellissi si vanno a ricercare quelle con soglia di Irradianza impostata per il nero totale, selezionando i marker di *tipo* β (*Figura 3.6*)

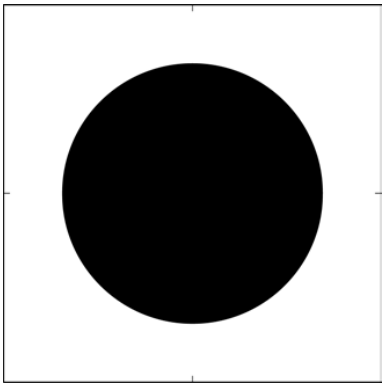


Figura 3.6: Marker di *tipo* β

e quelle con soglia d'Irradianza impostate per il bianco assoluto, selezionando i marker di *tipo* γ (*Figura 3.7*).
Si effettuano infine alcune operazioni di selezione e filtraggio atte in primis a scartare

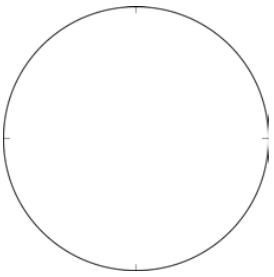
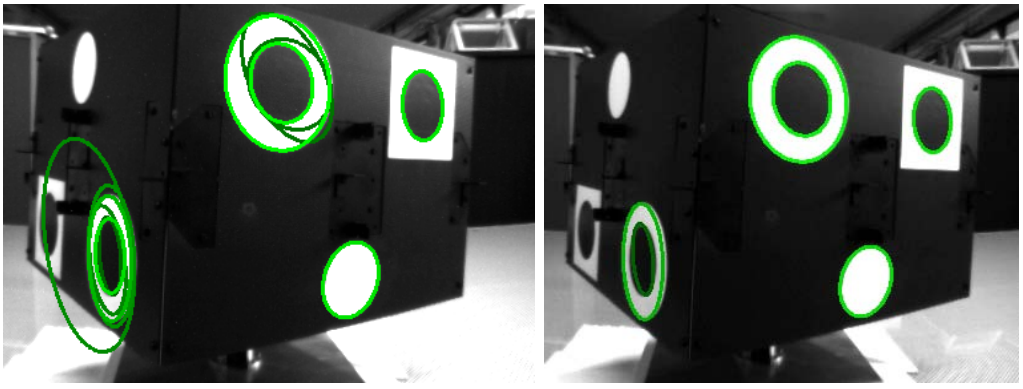


Figura 3.7: Marker di *tipo* γ

i doppioni possibili di uno stesso marker e soprattutto a scartare il secondo importante tipo di falso positivo, ovvero le ellissi rilevate che sono state in realtà create correlando bordi di diverse ellissi, ossia quelle coniche che appaiono sull'immagine come eccessivamente grandi. Queste vengono scartate andando ad eliminare quelle curve che contengono al loro interno i centri di altri markers già selezionati, caso geometricamente impossibile.
L'intero processo come per i markers di *tipo* α è gestito da un certo numero seppur non eccessivo di parametri configurabili, parametri che sono stati scelti dopo svariati test di selezione. Su questi si è deciso nel complesso di adottare una politica piuttosto restrittiva, in quanto è preferibile riconoscere una feature corretta in meno che selezionare un falso positivo in più. Dopo l'elaborazione da parte di tale metodo allora otteniamo risultati del tipo in *Figura 3.8*.



(a) Dopo rilievo *ellipseDetect()* (b) Dopo selezione *markersDetect()*

Figura 3.8: Esempio (1) di selezione dei markers tramite *markersDetect()*

Gli oggetti di tipo *Marker* così dichiarati e inizializzati, vengono stivati in un vettore membro della classe detto *markersList_t* che viene a sua volta rielaborato dal metodo a seguire in modo da andare ad assegnare l'unica caratteristica dei markers trovati lasciata insoluta, ovvero la loro faccia di appartenenza.

3.3.3 structuresRecognizing()

L'ultimo passaggio fatto all'interno della classe *SingleCamera* e quindi l'ultimo processo doppio poichè svolto per ognuna delle due camere, è quello attraverso il quale viene assegnata la faccia di appartenenza agli oggetti *Marker* trovati. É un passaggio fondamentale, perchè solo conoscendo la faccia e quindi la posizione della feature sul target è possibile ricollegarlo alle sue coordinate relative in riferimento *target* necessarie al calcolo d'assetto. Ciò significa che con il fallimento di questo processo i marker trovati sono inutili e si ha il fallimento dell'intero processo di analisi del frame. É per questo che per rendere tale metodo il più robusto possibile sono state implementate due strade parallele di riconoscimento che vengono poi confrontate e fuse:

1. la prima è implementata basandosi sul riconosimento della posizione relativa tra i markers;

2. la seconda, più efficiente ma non indipendente, è basata sulla memoria di percorso, ovvero sul tracciamento.

Il primo metodo pur essendo concettualmente meno efficiente, è strettamente necessario, ad esempio al primo frame cronologicamente analizzato in cui non si possiede memoria pregressa dello stato, o in caso di perdita saltuaria di un frame nell'analisi del video, che reinizializza la memoria. É strutturato come un complesso diagramma ad albero che tiene conto di ogni combinazione possibile di riconoscimento dei markers e tramite una serie di confronti scorre alla ricerca di due strutture di base in grado di permetterci di assegnare la faccia di appartenenza al marker, ovvero

- la *faccia*;

- lo *spigolo*.

Per come sono state pensate le posizioni dei markers (*Figura 2.9*), in base alla combinazione di markers rilevata nell'immagine, è possibile tramite la loro reciproca posizione capire che struttura stiamo vedendo, quindi un eventuale *faccia* o *spigolo*. Mentre la struttura *faccia* può essere intuitiva, la struttura *spigolo* è una combinazione del tipo in *Figura 3.9*.
Una volta riscontrata una di queste strutture è possibile assegnare ai markers che la compongono la faccia di appartenenza. Inoltre potendo vedere contemporanemante al più due facce, nel caso siano presenti ulteriori markers riconosciuti, si può assegnare anche a loro la faccia di appartenenza per completamento o esclusione.

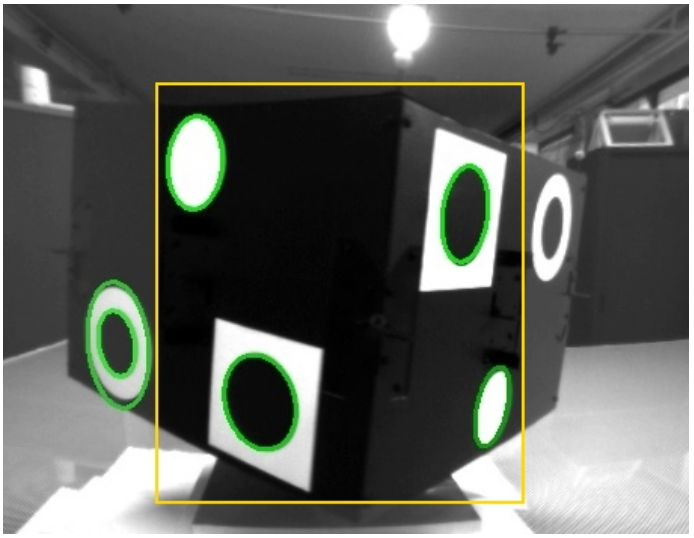


Figura 3.9: Esempio di *spigolo*, struttura geometrica ricercata per l'assegnazione della faccia di appartenenza al *Marker*

Il secondo metodo, che prende il nome di *tracking*, è invece concettualmente più efficiente ma non può funzionare da solo, ha bisogno che **almeno** al primo frame vengano riconosciute le strutture di appartenenza. Si basa infatti sul confronto dei marker riconosciuti dal *markersDetect()* su di un certo frame e contenuti nel vettore membro *markersList_t* (dove "*t*" sta per istante attuale *t*) con i markers riconosciuti sul frame precedente e contenuti in un altro vettore membro clone del primo chiamato *markersList_t_1* (dove "*t_1*" sta per istante *t-1*) che viene riempito all'inizio del processo di riconoscimento trasferendo i markers trovati al frame precedente e svuotando il vettore membro principale in modo da prepararlo al frame attuale. Il

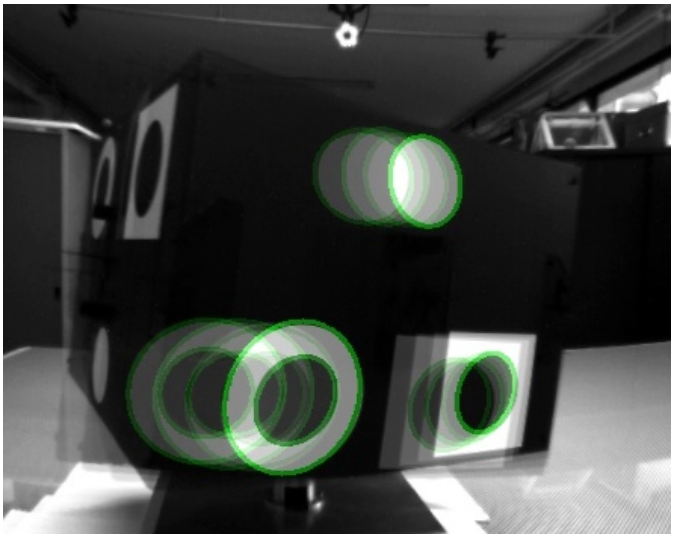


Figura 3.10: Concetto di tracking

processo di comparazione è concettualmente semplice, per ogni marker trovato al frame in corso, si itera la lista dei markers precedenti alla ricerca di un oggetto dello stesso *tipo* e il cui centro si trovi entro un certo raggio dal centro dell'oggetto precedente. Si suppone infatti che essendo i frame separati temporalmente da un tempo relativamente molto piccolo ed essendo le velocità relative in gioco non elevate, una stessa feature non possa aver cambiato drasticamente la sua posizione sul piano immagine e si trovi in una zona limitrofa alla precedente (*Figura 3.10*). Questo permette non solo di inseguire un marker (almeno fino a quando non verrà più riconosciuto) ma di assegnare ulteriormente per completamento o per eliminazione la faccia di appartenenza anche ad eventuali "nuovi" markers riconosciuti al frame del tempo t ma non nel frame del tempo *t-1*, generando in questo modo un autosostentamento del *tracking*. Tale metodo è gestito inoltre da soli due parametri, ovvero le soglie orizzontali e verticali entro le quali considerare i due markers, quello attuale e quello precedente, come limitrofi e quindi corrispondenti. Nel nostro caso, considerando velocità relative piuttosto basse, si imposta un valore funzionale di 30 pixel, ma è pensabile che tale valore possa essere reso variabile in corso di processo in base alla velocità rototraslazionale calcolata all'istante precedente.
Per motivi di efficienza, che si rispecchiano inevitabilmente sulla velocità di calcolo, la precedenza sul riscontro viene data al *tracking* e vengono invece usati i risultati del metodo alternativo solo in caso di errori di riconoscimento o vero e proprio fallimento del tracciamento, ad esempio nel caso in cui un frame intramezzo non venga riconosciuto e si perda memoria dello stato precedente. Gli oggetti di tipo *Marker* vengono quindi modificati nella loro caratteristica *idFace* e reinseriti nello stesso vettore membro *markersList_t* che è il frutto finale degli oggetti *SingleCamera* che vengono dati in ingresso all'oggetto *Stereocamera*.

3.4 Classe *Stereocamera*

Una volta processati i due frame, provenienti dalla camera di sinistra e da quella di destra, i due oggetti *SingleCamera* vengono dati in ingresso ad un oggetto *Stereocamera* che analizzando i vettori membro che trasportano con sé le due camere, cerca di calcolare la posa e l'assetto del target seguendo il processo schematizzato in *Figura 3.11*. Il primo passaggio importante è quindi quello di determinare la posizione dei marker trovati rispetto al sistema di riferimento *camera* e successivamente grazie a queste informazioni tentare di determinare la matrice di trasformazione tra

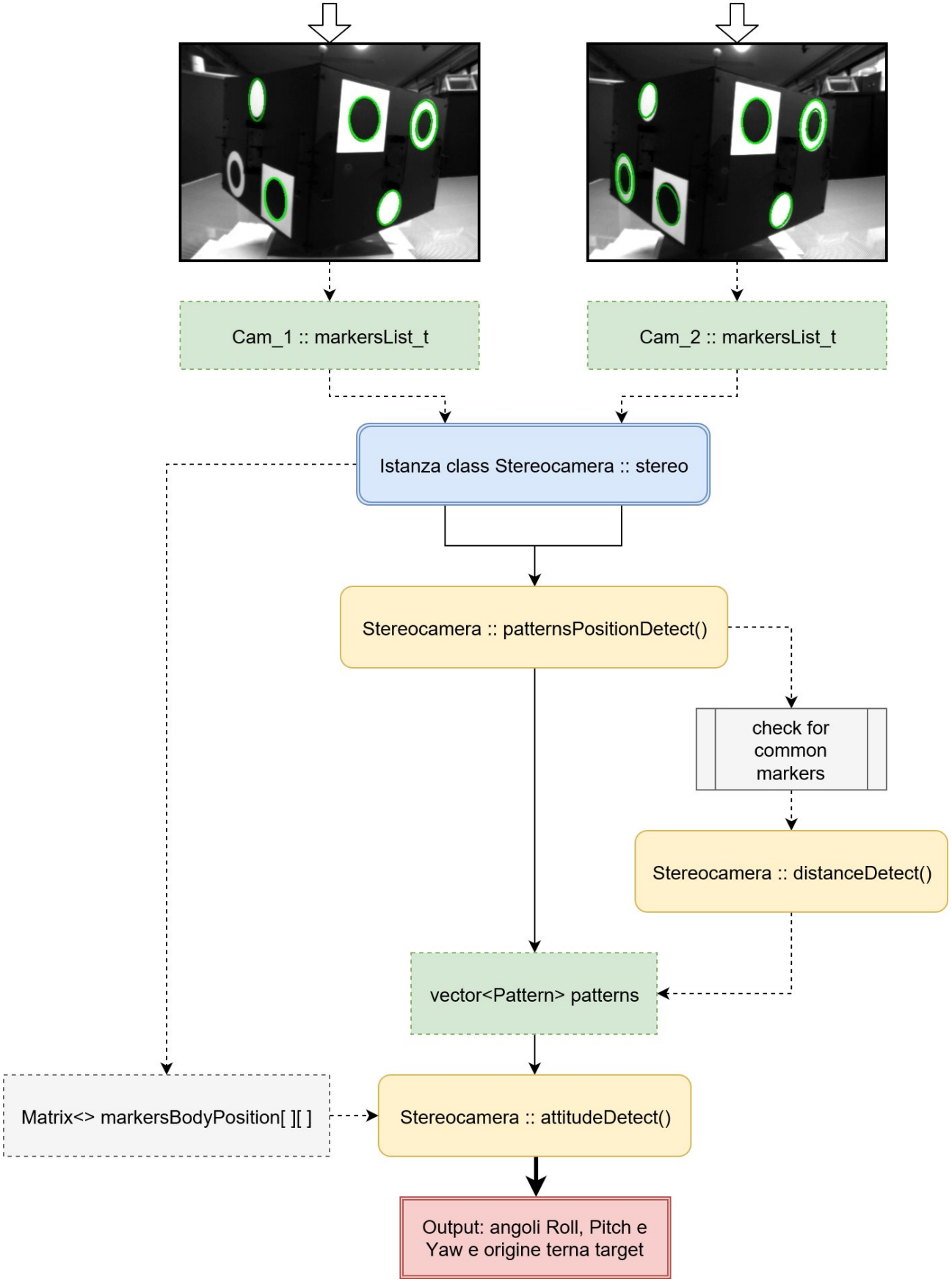


Figura 3.11: Struttura macroscopica della classe *Stereocamera*

il sistema *camera* e il sistema *target*.
Un primo elemento da precisare è l'implementazione all'interno della classe *Stereocamera* di un altro nuovo tipo di dato, tramite la struttura definita come *Pattern*. Questa è l'evoluzione della struttura creata precedentemente all'interno della classe *SingleCamera*, ovvero il *Marker*, ma nata dalla necessità di distinguere l'oggetto legato alla feature riscontrata nel sistema di riferimento *immagine* e l'oggetto legato alla stessa feature vista però in riferimento *camera*. Un oggetto di tipo *Pattern* ha come caratteristiche:

- il *tipo*, esattamente come l'oggetto *Marker*, ovvero α,β o γ;
- la *faccia* di appartenenza;
- i due oggetti *Marker* corrispondenti che hanno generato l'oggetto *Pattern* stesso;
- un contatore che tiene memoria del numero del *frame* in cui è stato riscontrato;
- un vettore contenente le coordinate cartesiane del *Pattern* in terna di riferimento *camera*.

Un altro elemento, contenuto anch'esso all'interno della classe *Stereocamera*, è la matrice membro definita *markersBodyPosition*, una matrice 12×5 contenente per ogni feature posta sul satellite target le coordinate relative rispetto alla terna di riferimento *target* e le informazioni per correlarli ai *Pattern* riscontrati, ovvero il *tipo* e la *faccia* di appartenenza.

3.4.1 patternsPositionDetect() e distanceDetect()

É questo il metodo che si preoccupa di gestire l'estrazione dei vettori proprietari *markersList_t* contenenti gli oggetti *Marker* trovati sulle due immagini dai due oggetti *SingleCamera* e di confrontare le due liste alla ricerca delle corrispondeze di *tipo* e *faccia*. Per ogni coppia di markers affini viene dichiarato ed inizializzato un oggetto di tipo *Pattern* assegnandogli le informazioni comuni e richiamando per ognuno il *distanceDetect()* che stima la posizione spaziale dell'oggetto rispetto alla terna di riferimento *camera*.

distanceDetect()

In questa sezione si analizza un altro dei metodi cardine dell'intero processo ovvero quello che ha il compito, dati in ingresso i due oggetti *Marker*, di calcolare

le coordinate cartesiane tridimensionali del punto che essi individuano, rispetto al riferimento *camera*.

In *sezione 1.2.7* dove si descrive il modello di una stereocamera, si analizza come scelti due sistemi di riferimento analoghi fissati sulle due camere, è sufficiente conoscere la *baseline* b, la distanza focale f caratteristica delle due camere e la *disparity* tra le proiezioni di uno stesso punto sui due piani immagine, per ottenere il valore della profondità e quindi la coordinata Z del punto stesso in riferimento *camera*. Questo attraverso l'*equazione 1.65* che riportiamo

$$Z = \frac{bf}{d}$$

La proporzionalità inversa tra la Z e d può però portare ad ottenere una ricostruzione errata, in quanto minimi errori nella stima della *disparity* dovuti ad esempio a rumore di misura, specialmente quando essa è molto piccola, possono generare grandi errori nella stima della profondità. Questo è il motivo per cui si sceglie di usare un altro metodo, seppur affine, per il calcolo di Z.

Concettualmente quello che si fa è trattare le due camere come fossero a se stanti ed applicare per ognuna il modello di camera ideale corretto dalle caratteristiche intrinseche visto in *sezione 1.2.6*. Riferendosi all'*equazione 1.62*, si descrive nello stesso paragrafo come in questo modello preso singolarmente vi sia una perdita di informazione legata alla profondità, data dal fatto che a dare contributo d'Irradianza ad un singolo pixel sono tutti e soli i punti che giacciono sulla retta che passa per un punto nello spazio tridimensionale e il fuoco della camera. Ed è proprio questa retta ad essere calcolata e sfruttata. Ricordando l'*equazione 1.61* in forma estesa

$$\lambda \begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} fS_x & fS_\theta & O_x \\ 0 & fS_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 1 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix}$$

una prima semplificazione è quella di non considerare, o meglio considerare come matrice identità I, la matrice di rototraslazione tra un eventuale sistema di riferimento *world* e il sistema di riferimento scelto, visto che in questo caso entrambi coincidono con il riferimento *camera*. Si perde allora anche la necessità della quarta coordinata nel vettore delle coordinate spaziali e l'equazione diventa

$$\lambda \begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} fS_x & fS_\theta & O_x \\ 0 & fS_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} \tag{3.2}$$

Ricordando che il parametro λ parametrizza proprio la profondità incognita nel modello di singola camera ideale, possiamo scrivere l'equazione inversa

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \lambda \begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} \begin{bmatrix} fS_x & fS_\theta & O_x \\ 0 & fS_y & O_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \tag{3.3}$$

dove quella che si ottiene è l'equazione parametrica di una retta nello spazio, passante per l'origine del sistema di riferimento e per un punto la cui proiezione sul piano immagine corrisponde a (x_{im}, y_{im}), retta del tipo

$$\begin{cases} X = \lambda c_x \\ Y = \lambda c_y \\ Z = \lambda \end{cases} \tag{3.4}$$

o in forma compatta

$$\mathbf{X} = \lambda \mathbf{c} \tag{3.5}$$

Considerando allora i sistemi di riferimento centrati sulle due camere e la relazione che li lega, che come descritto in *sezione 1.2.7* se si considerano camere con assi ottici paralleli, è una semplice traslazione, dato un punto p e le sue proiezioni sui due piani immagine, le due rette su cui questo dovrebbe giacere sono

$$\begin{cases} X_1 = \lambda_1 c_{x1} \\ Y_1 = \lambda_1 c_{y1} \\ Z_1 = \lambda_1 \end{cases} \tag{3.6}$$

la retta nello spazio passante per il punto e il fuoco della camera di sinistra, e

$$\begin{cases} X_2 = \lambda_2 c_{x2} + b \\ Y_2 = \lambda_2 c_{y2} \\ Z_2 = \lambda_2 \end{cases} \tag{3.7}$$

la retta nello spazio passante per il punto ed il fuoco della camera di destra, riferita però rispetto alla terna della camera di sinistra, traslata infatti della distanza b cioè la *baseline* tra le camere. In forma compatta le due rette hanno forma del tipo

$$\mathbf{X}_1 = \lambda_1 \mathbf{c}_1 \tag{3.8}$$

$$\mathbf{X}_2 = \lambda_2 \mathbf{c}_2 + \mathbf{b} \tag{3.9}$$

con $\mathbf{b}$ il vettore $[b, 0, 0]^T$.
Nel caso ideale, in cui il metodo di distorsione inversa ricostruisce perfettamente l'immagine reale, la rilevazione delle ellissi ha precisione assoluta, le camere sono calibrate in modo impeccabile e non ci sono incertezze costruttive, il calcolo delle coordinate del punto p risulta banale, in quanto non bisogna far altro che trovare il punto di intersezione delle due rette, intersezione la cui esistenza è garantita dal vincolo epipolare che forza la complanarità delle due linee. Questo può accadere nel migliore dei casi, ma a causa delle incertezze e delle loro propagazioni, vi è un'altissima probabilità che le rette trovate siano tra di loro sghembe nello spazio come in *Figura 3.12*.

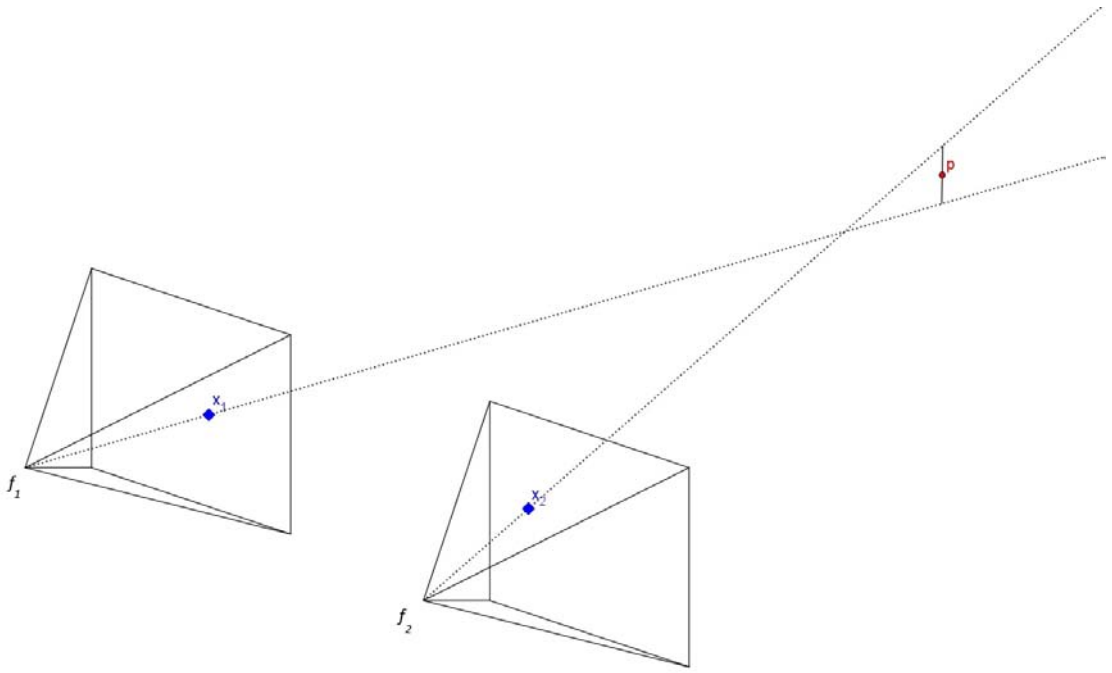


Figura 3.12: Problema di calcolo della posizione tridimensionale del punto p in caso di rette sghembe

Quello che si fa è allora cercare il valore dei parametri λ_1 e λ_2 delle *equazioni 3.8-3.9* tali da individuare i due punti che sulle due rette minimizzino la distanza tra le rette stesse, e quindi stimare come posizione del punto p il punto medio di detti punti. La distanza tra due generici punti delle due rette è definita da

$$\mathbf{d} = \mathbf{b} + \lambda_2 \mathbf{c}_2 - \lambda_1 \mathbf{c}_1 \tag{3.10}$$

e la quantità da minimizzare è perciò $\|\mathbf{d}\|^2$, il cui gradiente si annulla in

$$\mathbf{c}_1 \cdot \mathbf{c}_1\lambda_1 - \mathbf{c}_1 \cdot \mathbf{c}_2\lambda_2 = \mathbf{c}_1 \cdot \mathbf{b} \tag{3.11}$$

$$\mathbf{c}_2 \cdot \mathbf{c}_1\lambda_1 - \mathbf{c}_2 \cdot \mathbf{c}_2\lambda_2 = \mathbf{c}_2 \cdot \mathbf{b} \tag{3.12}$$

sistema di equazioni tramite le quali è possibile calcolare λ_1 e λ_2 che inserendo all'interno delle *equazioni 3.8-3.9* delle rette, permettono a loro volta di ricavare le coordinate $\mathbf{X}_1$ e $\mathbf{X}_2$ attraverso la cui media si trovano infine le coordinare $[X, Y, Z]$ del punto p nel sistema di riferimento *camera*.

Il *patternsPositionDetect()* quindi, dopo aver dichiarato e inizializzato gli oggetti *Pattern* ed aver assegnato loro le coordinate cartesiane calcolate tramite *distanceDetect()*, accumula tali oggetti nel vettore membro *patterns* dove saranno a disposizione del calcolatore d'assetto.

3.4.2 attitudeDetect()

L'ultimo metodo in ordine sequenziale del processo di elaborazione dei frame, è quello che si occupa effettivamente del calcolo dei parametri di posa e assetto. Questo si suddivide in base a due casistiche:

1. il caso in cui si stiano vedendo 4 o più markers;
2. il caso in cui si stiano vedendo 3 markers appartenenti alla stessa faccia.

Qualunque altro caso, quindi quelli in cui si riscontrino 2 soli markers o 3 markers non appartenenti alla stessa faccia, non permette la stima di posa e assetto e determina quindi il fallimento dell'intero processo per il frame in analisi.

Per il primo caso, dopo essere risaliti alle coordinate in terna *target* dei punti trovati, tramite la matrice proprietaria *markersBodyPosition*, ricordando ed adattando l'*equazione 1.11* è possibile scrivere

$$[\mathbf{P}^{cam}] = [\mathbf{T}] \left[\mathbf{P}^{trgt}\right] \tag{3.13}$$

dove

$$[\mathbf{P}^{cam}] = \begin{bmatrix} \mathbf{p}_i^{cam} \\ 1 \end{bmatrix} = \begin{bmatrix} x_1^{cam} & x_2^{cam} & x_3^{cam} & x_4^{cam} & \dots \\ y_1^{cam} & y_2^{cam} & y_3^{cam} & y_4^{cam} & \dots \\ z_1^{cam} & z_2^{cam} & z_3^{cam} & z_4^{cam} & \dots \\ 1 & 1 & 1 & 1 & \dots \end{bmatrix} \tag{3.14}$$

ovvero una matrice contenente le coordinate di tutti i *Pattern* riconosciuti rispetto al riferimento *camera*, e

$$\left[\mathbf{P}^{trgt}\right] = \begin{bmatrix} \mathbf{p}_i^{trgt} \\ 1 \end{bmatrix} = \begin{bmatrix} x_1^{trgt} & x_2^{trgt} & x_3^{trgt} & x_4^{trgt} & ... \\ y_1^{trgt} & y_2^{trgt} & y_3^{trgt} & y_4^{trgt} & ... \\ z_1^{trgt} & z_2^{trgt} & z_3^{trgt} & z_4^{trgt} & ... \\ 1 & 1 & 1 & 1 & ... \end{bmatrix} \tag{3.15}$$

ovvero una matrice contenente le coordinate degli stessi *Pattern* rispetto però al riferimento *target*. Dalla relazione inversa è possibile ottenere la matrice di rototraslazione $\mathbf{T}$, quindi

$$[\mathbf{T}] = [\mathbf{P}^{cam}] \left[\mathbf{P}^{trgt}\right]^{-1} \tag{3.16}$$

dove in particolare, nel caso in cui si stia utilizzando una matrice di quattro punti e quindi una matrice $\left[\mathbf{P}^{trgt}\right]$ quadrata 4×4, la matrice $\left[\mathbf{P}^{trgt}\right]^{-1}$ è l'*inversa* di $\left[\mathbf{P}^{trgt}\right]$, mentre nel caso si stia utilizzando una matrice con più di quattro punti (un massimo di *sei* potendo vedere al massimo due facce contemporaneamente con tre markers ognuna) e quindi una matrice $\left[\mathbf{P}^{trgt}\right]$ rettangolare 4×5 o 4×6, essendo la matrice inversa calcolabile solo per matrici quadrate, $\left[\mathbf{P}^{trgt}\right]^{-1}$ è invece la matrice *pseudo-inversa* di $\left[\mathbf{P}^{trgt}\right]$.
La matrice *pseudo-inversa* o matrice *pseudo-inversa di Moore-Penrose* è la generalizzazione della matrice inversa per matrici rettangolari e data una generica matrice $A \in n \times m$ viene indicata con A^+. Esistono due matrici *pseudo-inverse* $m \times n$, una destra ed una sinistra, entrambe a soddisfare

$$A^+ A = I \tag{3.17}$$

$$AA^+ = I \tag{3.18}$$

con I la matrice identità. Esse sono definite, la *pseudo-inversa* sinistra come

$$A^+ = \left(A^T A\right)^{-1} A^T \tag{3.19}$$

mentre la *pseudo-inversa* destra come

$$A^+ = A^T \left(AA^T\right)^{-1} \tag{3.20}$$

Quella che si calcola in questo ambito è la *pseudo-inversa* destra della matrice $\left[\mathbf{P}^{trgt}\right]$.

Per la seconda casistica questo metodo non è però utilizzabile, in quanto nel caso si stiano vedendo 3 soli markers appartenenti alla stessa faccia, tali punti sono

obbligatoriamente complanari e questo da una matrice $\left[\mathbf{P}^{trgt}\right]$ che una volta invertita e moltiplicata per $[\mathbf{P}^{cam}]$, genera una matrice $\mathbf{T}$ singolare.
Quello che si fa allora è utilizzare concettualmente la stessa equazione utilizzata sopra ma facendo un passo indietro dalle coordinate omogenee, ovvero tornando all'*equazione 1.8* nella forma

$$[\mathbf{p}_{cam}] = \left[R^{cam}_{trgt}\right]\left[\mathbf{p}_{trgt}\right] + \left[T^{cam}_{trgt}\right] \quad con \quad \mathbf{p}, \mathbf{R} \in R^{3\times 3} \quad e \quad T \in R^{3\times 1} \tag{3.21}$$

La singolarità del caso dei tre markers complanari, pur conoscendo le loro coordinate relative in terna *target*, è data dal fatto che le soluzioni alla posizione relativa della terna stessa rispetto al piano formato dai tre punti, sono effettivamente due, una frontale ed una posteriore al piano, ovvero una per ognuno dei due semispazi in cui il piano divide lo spazio tridimensionale. Ma se si conosce l'origine reale della terna *target*, quest'ambiguità può essere risolta sottraendo ai punti in terna *camera* calcolati, la posizione relativa tra le due terne, traslando cioè i punti in un riferimento con la stessa origine del riferimento *target*. A questo punto l'unica differenza è quella rotazionale e basta invertire la matrice dei punti 3×3 per ottenere la matrice di rotazione da cui estrarre gli angoli di Eulero. In equazioni

$$[\mathbf{p}_{cam}] - \left[T^{cam}_{trgt}\right] = \left[R^{cam}_{trgt}\right]\left[\mathbf{p}_{trgt}\right] \tag{3.22}$$

dove T^{cam}_{trgt} è il vettore traslazione contenente le coordinate dell'origine della terna *target* in riferimento *camera*, e quindi

$$\left[R^{cam}_{trgt}\right] = \left[\mathbf{p}_{cam} - T^{cam}_{trgt}\right]\left[\mathbf{p}_{trgt}\right]^{-1} \tag{3.23}$$

Il problema che rimane da risolvere è allora quello di riuscire a determinare l'origine della terna *target*, che coincide in pratica col determinarne la posa. Per fare questo (*Figura 3.13*), quello che si decide di fare è:

1. generare il versore perpendicolare al piano della faccia, ovvero il coseno direttore del piano su cui giacciono i tre punti;
2. calcolare il punto centrale della stessa faccia;
3. generare la retta nello spazio passante per detto punto centrale e della stessa inclinazione del versore trovato;
4. conoscendo la distanza d della faccia dall'origine della terna, trovare il punto appartenente alla retta trovata e a distanza d dal punto centrale della faccia.

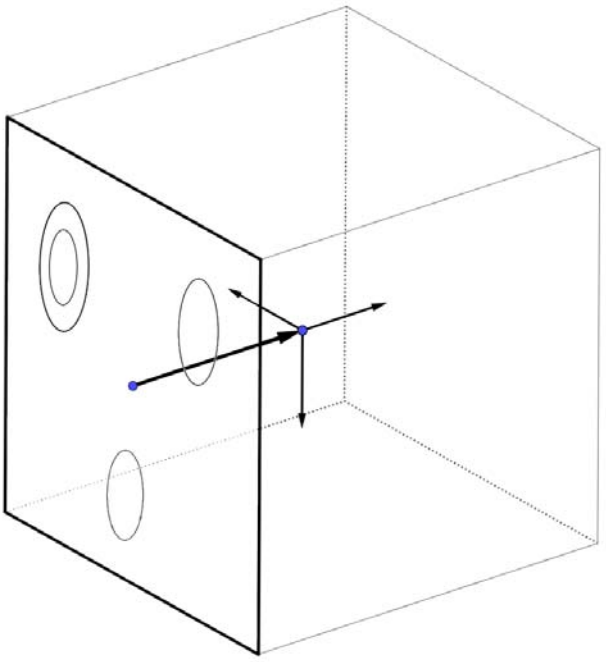


Figura 3.13: Problema di calcolo d'assetto nel caso di 3 markers complanari

Per ricavare il versore del piano è sufficiente calcolare i due vettori che da un marker puntano ai due restanti e di questi farne il prodotto vettoriale, accertandosi che il versore risultante abbia prodotto scalare negativo rispetto alla terna di riferimento *camera*, ovvero che punti sempre in direzione opposta. Dette allora a, b, c le componenti del versore già normalizzate e (c_x, c_y, c_z) le coordinate del centro della faccia nel sistema di riferimento camera, la retta nello spazio passante per il centro della faccia è

$$r : \begin{cases} x &= at + c_x \\ y &= bt + c_y \\ z &= ct + c_z \end{cases} \tag{3.24}$$

ed impostando la distanza d di un generico punto appartenente alla retta dal centro della faccia

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = d^2 \tag{3.25}$$

$$(at + c_x - c_x)^2 + (bt + c_y - c_y)^2 + (ct + c_z - c_z)^2 = d^2 \tag{3.26}$$

$$a^2t^2 + b^2t^2 + c^2t^2 = d^2 \tag{3.27}$$

si ottiene facilmente t come

$$t = \frac{d}{\sqrt{a^2 + b^2 + c^2}} \tag{3.28}$$

che sostituendo all'interno dell'*equazione 3.24* parametrica della rettà, restituisce le coordinate del centro del sistema di riferimento target.

Ottenuta la matrice di rotazione $[R_{trgt}^{cam}]$ si può passare ai quaternioni ed usare le relazioni che li legano agli angoli di Eulero viste in *Sezione 1.1.2* per ricavare gli angoli di *Roll*,*Pitch* e *Yaw*.

Capitolo 4

Presentazione Risultati dei Test

4.1 Test in Ambiente Virtuale

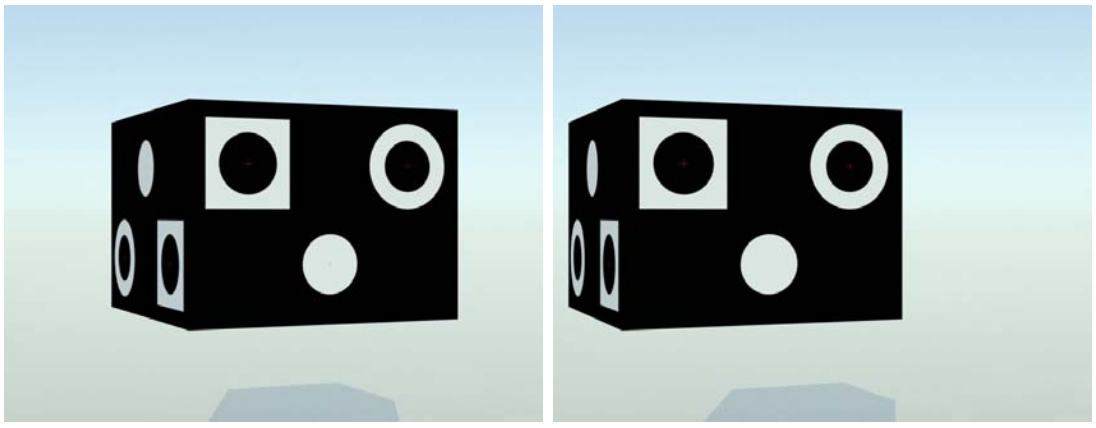
Si entra ora nella fase di test dell'intero sistema. Viene illustrata in questo paragrafo la prima parte dei test, effettuata in un ambiente virtuale per poterla rendere più veloce e flessibile in caso di modifiche necessarie e soprattutto per non andare a usurare inutilmente il sistema di simulazione fisico prima di avere a disposizione un software almeno in parte funzionante. Si sviluppa allora un semplice modello in 3D ricalcante le caratteristiche geometriche del *modulo d'assetto* del simulatore satellitare e nello stesso è stata implementata una stereocamera virtuale. Il tutto è stato sviluppato utilizzando il software di disegno e progettazione tridimensinoale Solidworks che tra le molte funzionalità ha anche quella di poter generare movimento degli *assiemi* creati ed estrapolare tramite degli oggetti di tipo *videocamera*, dei video che tramite rendering possano approssimare ottimamente l'ambiente reale. Questo grazie al fatto che le misure delle costruzioni geometriche costruite in ambiente di disegno, vengono mantenute nei video ed è quindi possibile utilizzare tali video come buone alternative ai reali input di una stereocamera fisica.
Prima di procedere a mostrare il modello riprodotto è bene specificare le ipotesi di lavoro nella quali ci si è posti per testare il modello ed il software sviluppato:

- Esula da questo lavoro la trattazione della manovra di avvicinamento orbitale tra i due satelliti in formazione, ci si pone in uno scenario di simulazione in cui il satellite *inseguitore* si trova già in un orbita relativa attorno al satellite target e a distanze relative piccole rispetto al raggio orbitale;

- In accordo all'ipotesi sopra, ma senza perdere di generalità, consideriamo le velocità traslazionali nulle o al limite molto basse;

- Il moto di maggior interesse è perciò quello rotazionale e con rotazione approssimativamente monoassiale, questo anche considerando le limitazioni di movimento angolari del giunto a tre assi del modulo d'assetto del micro satellite reale;
- Le dimensioni del modello virtuale sono identiche alle dimensioni del sottosistema strutturale del modulo d'assetto del simulatore.

Ecco in *Figura 4.1* come appare il modello virtuale visto dai due canali della stereocamera virtuale

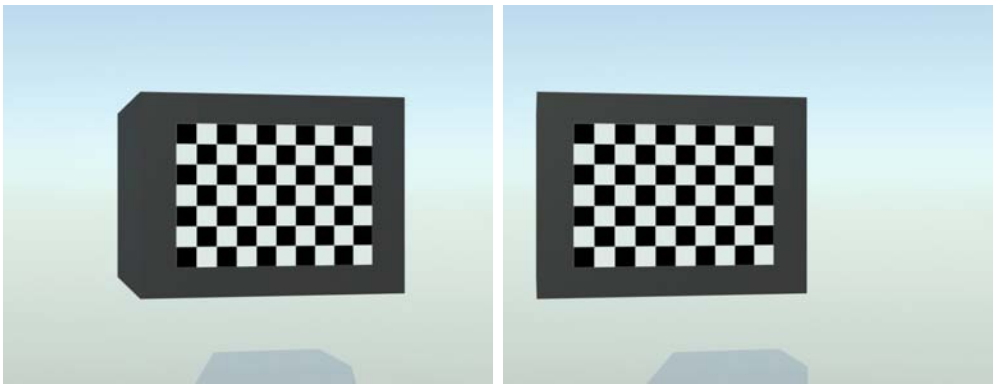


(a) Camera *left* (b) Camera *right*

Figura 4.1: Il modello virtuale *Solidworks* del simulatore di satellite, visto dalla stereocamera virtuale

Unico intoppo nell'utilizzo dell'ambiente virtuale Solidworks è che la costruzione delle videocamere non permette l'impostazione dei parametri intrinseci necessari alla ricostruzione tramite l'algoritmo, poichè non pensata per l'uso che se ne vuole fare in questo progetto di tesi, ma bensì alcuni parametri ottico-geometrici come il campo di vista e la grandezza del sensore o del piano immagine. Le uniche cose utili da noi impostabili sono la sua posizione rispetto al sistema di riferimento globale dell'ambiente virtuale, la *baseline* tra le due camere e la risoluzione dei video, scelta 800×618 per tenerla vicina al valore della stereocamera reale DUO M.

Il problema di dover ricavare i parametri instrinseci si risolve scegliendo un'impostazione Solidworks di base per ogni videocamera e la si usa per tutti i test in modo da avere come riferimento sempre la stessa stereocamera. Con queste impostazioni si crea poi un video del modello su cui è stata applicata una *dashboard* (*Figura 4.2*)



(a) Camera *left* (b) Camera *right*

Figura 4.2: Immagini *dashboard* per calibrazione stereocamera in MATLAB

e una volta estratti un sufficiente numero di frame da tale video, questi vengono processati in MATLAB il quale restituisce la matrice di calibrazione contenente i parametri intrinseci della stereocamera. In particolare i valori dei parametri intrinseci per la stereocamera virtuale risultano, in pixel

Param. Intrinsec. Stereo. Virtuale [pixel]	Camera *left*	Camera *right*
focale f_x $(f \cdot s_x)$	1649.2	1650.2
focale f_y $(f \cdot s_y)$	1648.6	1649.6
centro piano immagine C_x	399.2	400.52
centro piano immagine C_y	309.5	310.4
skew	0	0

Tabella 4.1: Parametri intrinseci stereocamera virtuale

Nel seguito si analizzano i risultati ottenuti ed i relativi errori per ognuna delle fasi del processo di stima, quindi:

- il riconoscimento delle forme ellittiche sulle immagini in ingresso;
- il calcolo delle posizioni dei markers nello spazio in riferimento *camera*;
- la posa e gli angoli d'assetto della terna target rispetto alla terna di riferimento *camera*.

Per fare questo, grazie all'utilizzo di un modello cinematico virtuale, dato in input il movimento programmato, sono state ricostruite le traiettorie ideali teoriche per

ognuna delle caratteristiche in esame in modo da poter confrontare i risultati ottenuti. Saranno analizzati inoltre i tempi di elaborazione e la velocità angolare stimata in fase di postprocessing.

Nelle ipotesi in cui ci si è posti, sono stati creati e processati due video test:

1. Nel primo video il modello compie una sola rotazione monoassiale lungo l'asse Z del riferimento target, con centro di massa posto frontalmente al riferimento *camera* ad una distanza di 2.27 *metri* e in rotazione ad una velocità angolare relativa di $\frac{2\pi}{5}$ rad/sec, completando 2 giri completi per una durata del video di 10 secondi;

2. Il secondo video introduce una lenta velocità traslazionale relativa. Rispetto al *riferimento camera* il modello partirà spostato lateralmente in posizione $(-0.25, 0, 3.27)$ *metri* percorrendo un primo tratto a velocità relativa di $(0.14, 0, 0)$ $metri/sec$ latidudinalmente (per cui restando a 3.27 *metri* di profondità) ed un secondo tratto in obliquo ad una velocità di $(-0.09, 0, -0.2)$ $metri/sec$ avvicinandosi così fino ad una distanza di 2.27 *metri*. Il tutto comprensivo di un egual moto rotazionale del video (1) per un totale anch'esso di 10 secondi di video.

4.2 Risultati test *rotazionale*

Si presentano in questa sezione i risultati relativi alla prima delle due manovre test, suddividendo in sottosezioni riguardanti fasi o aspetti diversi dell'intera catena di processo. Per ognuna sono presentati i grafici di comparazione tra i casi ideali ed i casi stimati ed i relativi errori, in particolare scarto massimo ε_{max} (in valore assoluto), scarto medio ε_{mean} e scarto quadratico medio σ.

4.2.1 Risultati *markersDetect()*

Sono mostrati nelle *Figure 4.3 - 4.4 - 4.5* i rilievi delle coordinate su piano immagine della camera di sinistra (i risultati per la camera di destra sono analoghi) dei centroidi delle ellissi per ognuna delle dodici features applicate sulla struttura esterna del modello e suddivise per *tipo* di marker. I simboli circolari indicano le coordinate dei markers sul piano immagine rilevate dal *markersDetect()* attraverso l'algoritmo Fornaciari-Prati. Le linee continue indicano invece le traiettorie ideali

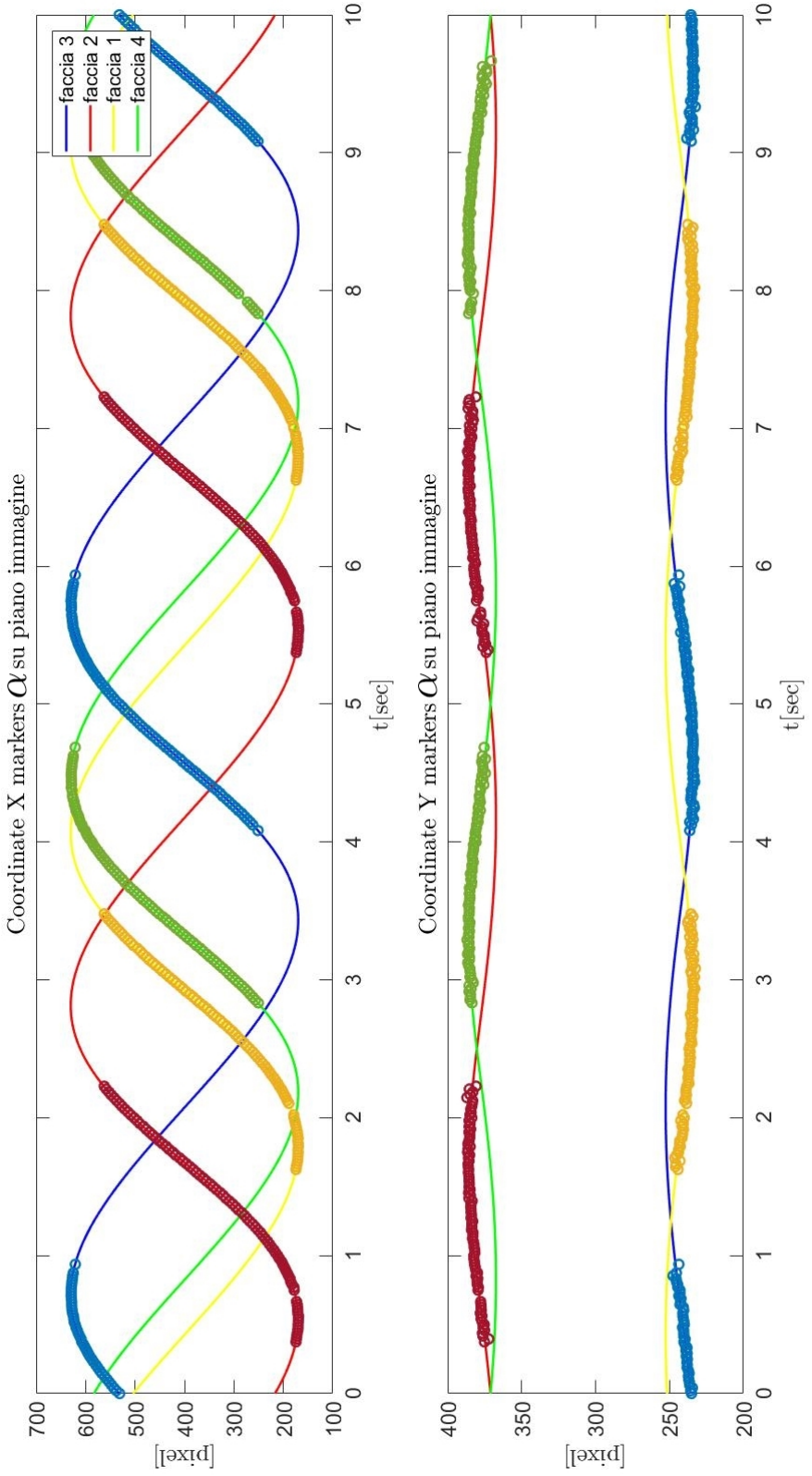


Figura 4.3: Test *rotazionale* risultati *markersDetect()* rilievo *markers* α

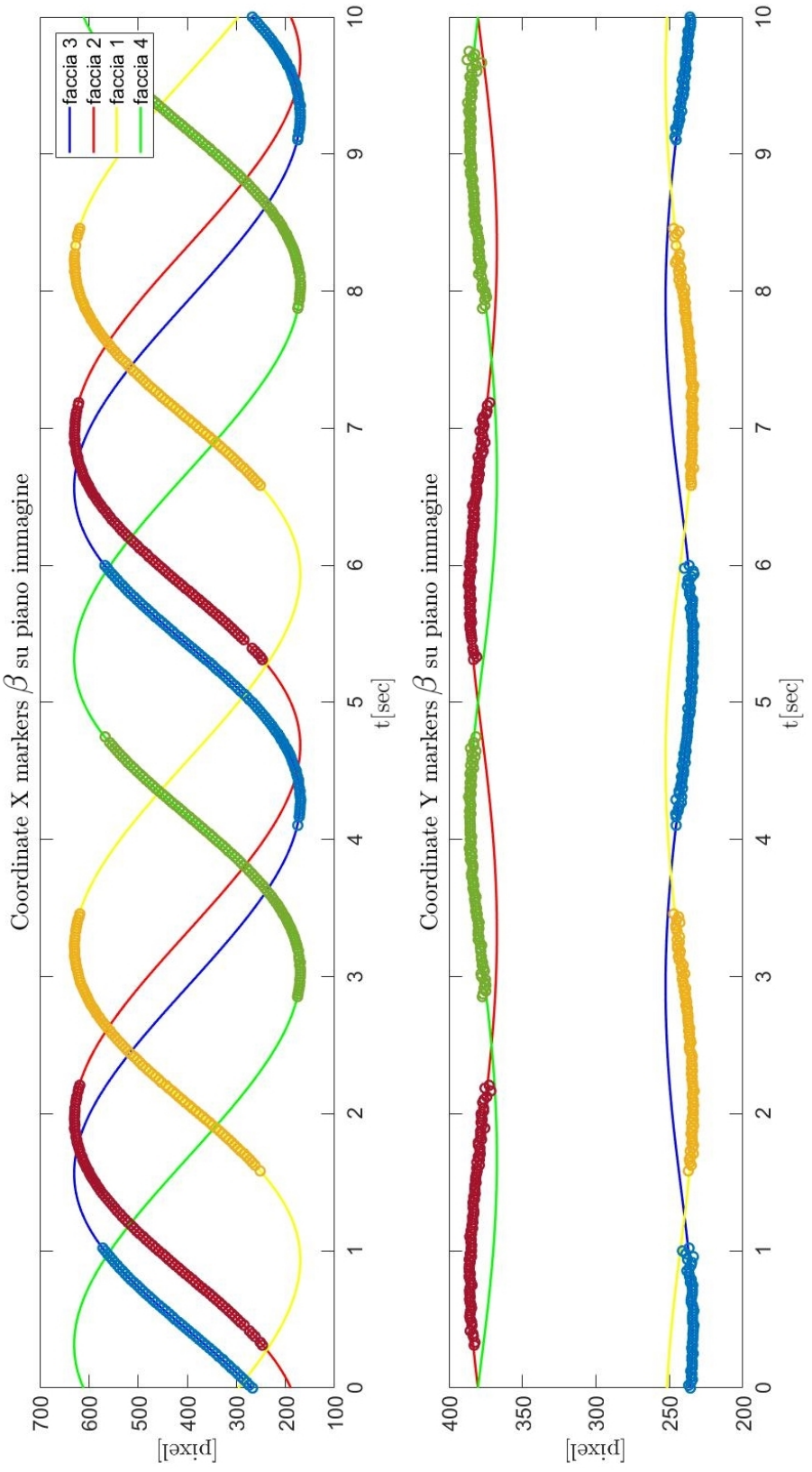


Figura 4.4: Test *rotazionale* risultati *markersDetect()* rilievo *markers* β

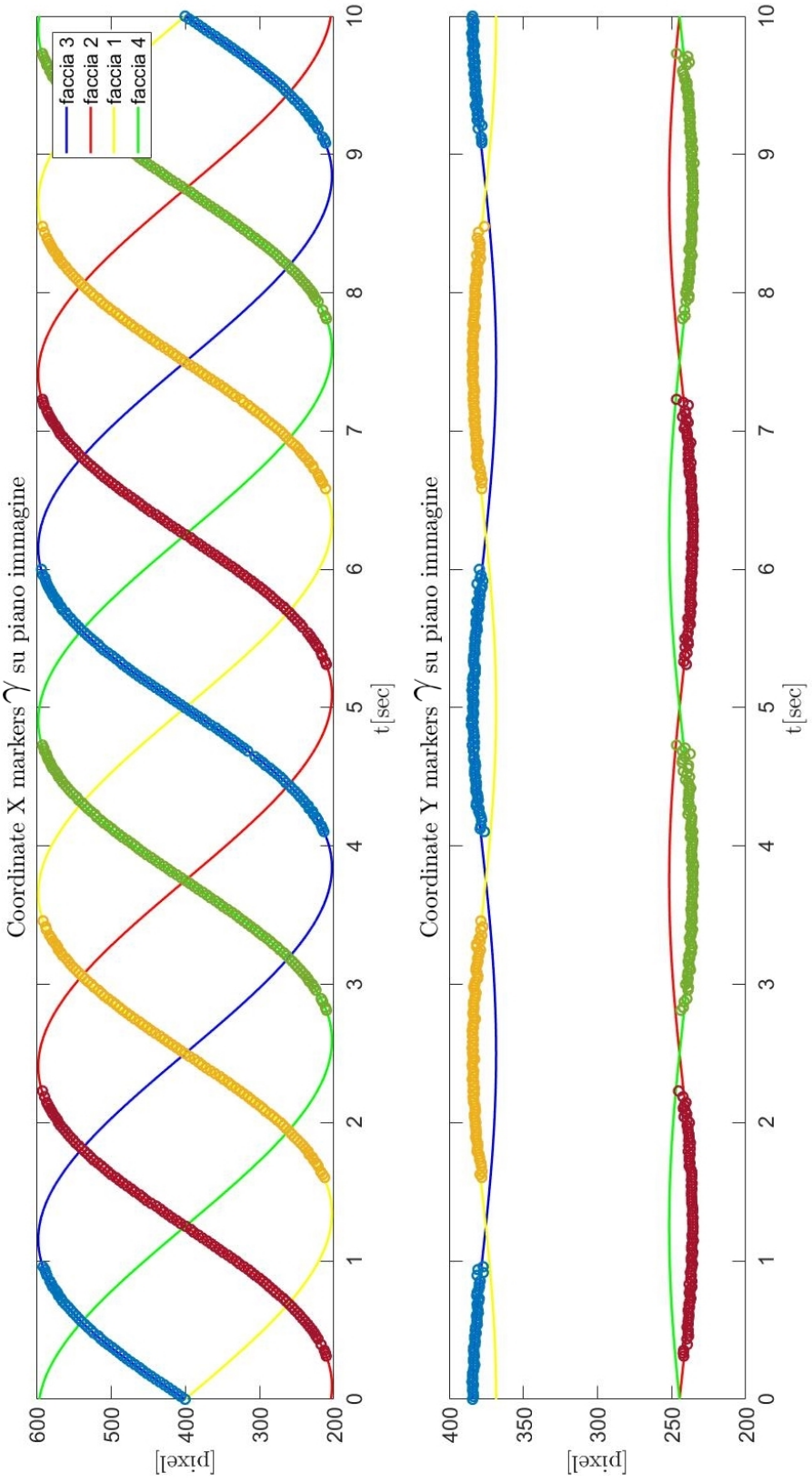


Figura 4.5: Test *rotazionale* risultati *markersDetect()* rilievo *markers* γ

delle proiezioni dei markers sul piano immagine che avrebbero considerata la loro posizione per l'intera manovra e che avrebbero se fossero perennemente in vista. Ovviamente questo non accade poichè un marker è in vista fintanto che lo è la sua faccia di appartenenza, o meglio, viene rilevato fino la normale alla faccia forma un determinato angolo con l'asse Z uscente del riferimento *camera*. In questo senso, si propone in *Figura 4.6* la corrispondenza tra il rilievo di un marker e la relativa angolazione della sua faccia di appartenenza in quel dato istante, in particolare è presentata la faccia (1) ma è da ritenersi generale per ognuna delle quattro facce. Si nota un'ampia angolazione di successo dell'*ellipseDetect()*, considerabile in un intervallo di $(-65, 65)deg$ circa. Il vuoto presente nella prima riga del grafico sta ad indicare istanti di tempo in cui nello specifico il marker α della faccia 1 non è stato individuato per possibili problemi legati al rilevamento o alla selezione, ma questo non pregiudica la buona riuscita del processo di stima se sono stati rilevati nel complesso un numero sufficiente di markers.

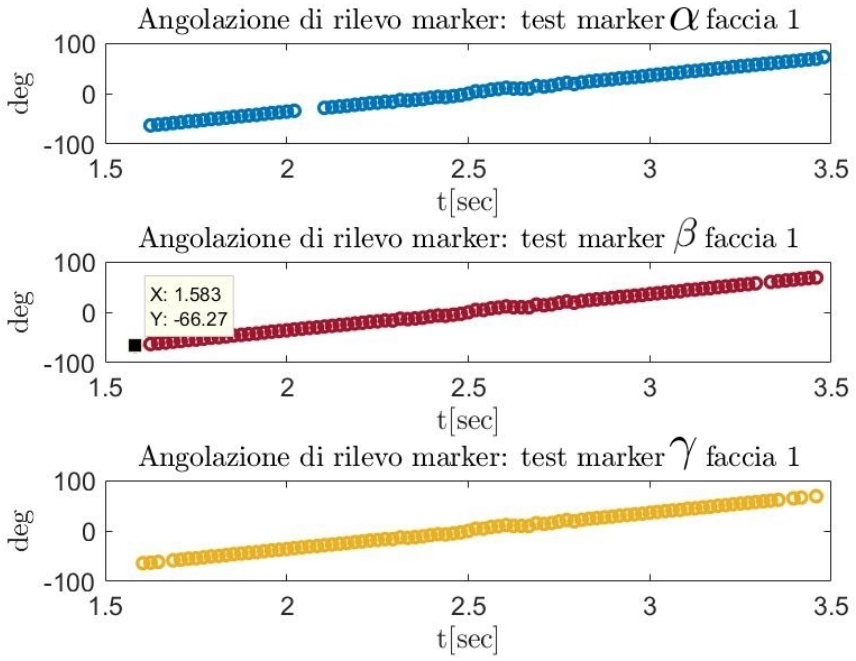


Figura 4.6: Range angolare in cui un marker e' in vista della camera, in funzione dell'angolo tra normale al piano dei markers e l'asse ottico

Riferendoci nuovamente al rilievo delle coordinate dei centroidi, si può dire che le traiettorie ideali sono ben seguite e si nota, anche grazie all'ampia angolazione di rilievo appena citata, come esistano degli intervalli temporali in cui sono visti contempo-

raneamente i markers dello stesso *tipo* ma di facce differenti, cosa che garantisce di non perdere la visione sul *target* in nessuna angolazione.
Presentiamo in *Figura 4.7* l'andamento degli errori di rilevamento dei centroidi dei markers, limitandoci a quelli dei markers di *tipo* α in quanto del tutto simili sono quelli dei markers di *tipo* β e γ.

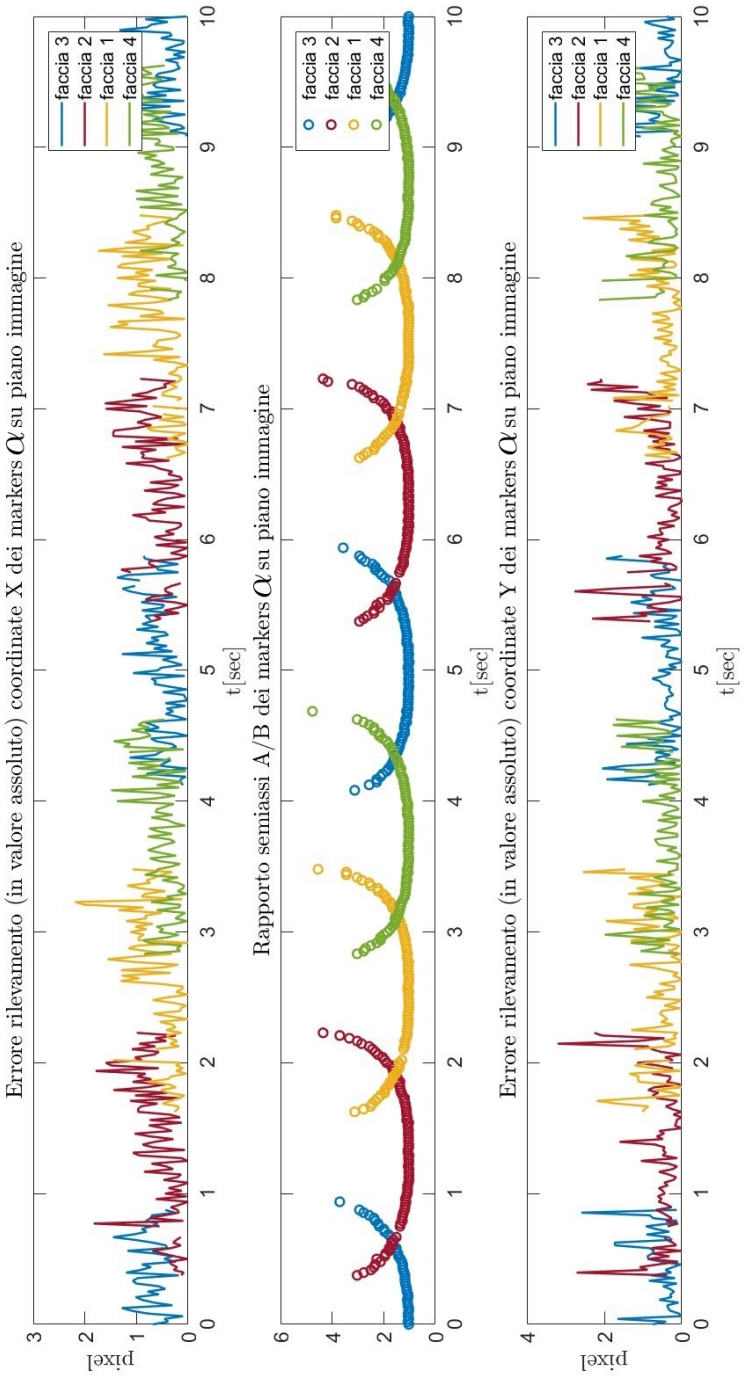


Figura 4.7: Test *rotazionale*, errori di rilevamento coordinate centroidi markers α

Insieme agli errori è graficato nel mezzo anche il rapporto tra i semiassi degli ellissoidi rilevati, questo per mostrare come i picchi d'errore si trovino soprattutto in quegli istanti di tempo in cui il rapporto fra semiasse maggiore e semiasse minore si accentua, ovvero quando l'angolo del coseno direttore della faccia rispetto all'asse di profondità Z della camera si accentua.
Si può notare inoltre come gli errori sulla coordinata y siano più elevati di quelli sulla coordinata x. A questo non è stata trovata una vera e propria spiegazione, l'ipotesi che si può fare è che nel video scelto, di rotazione monoassiale verticale, la maggior distorsione della feature circolare si ha nel senso delle ordinate e questo può riflettersi in un maggior errore di misura sulla coordinata verticale più che su quella orizzontale, ossia y.
Si riportano in *tabella 4.2* i valori degli errori calcolati per ognuno dei markers.
Si conferma numericamente il trend per il quale in generale l'errore è maggiore sulla coordinata y. Ad una prima analisi, un errore medio tra 0.5/0.6 *pixel* può ritenersi basso, ma in realtà questo dato va poi comparato con gli errori sulla stima della posizione e dell'assetto che esso comporta.

Faccia 1	ε_{max} $[pixel]$	ε_{mean} $[pixel]$	σ $[pixel]$
Marker α - coordinata X	2.1914	0.5938	0.7431
Marker α - coordinata Y	2.5528	0.5997	0.7862
Marker β - coordinata X	2.0109	0.5488	0.6767
Marker β - coordinata Y	2.5744	0.5312	0.7234
Marker γ - coordinata X	1.8481	0.6094	0.7456
Marker γ - coordinata Y	2.2423	0.4759	0.6310
Faccia 2	ε_{max} $[pixel]$	ε_{mean} $[pixel]$	σ $[pixel]$
Marker α - coordinata X	1.8287	0.5806	0.7092
Marker α - coordinata Y	3.2074	0.6071	0.8380
Marker β - coordinata X	1.9995	0.5764	0.7064
Marker β - coordinata Y	2.8109	0.5788	0.7755
Marker γ - coordinata X	1.7259	0.5732	0.7230
Marker γ - coordinata Y	4.7020	0.5158	0.7732
Faccia 3	ε_{max} $[pixel]$	ε_{mean} $[pixel]$	σ $[pixel]$
Marker α - coordinata X	1.4549	0.5497	0.6490
Marker α - coordinata Y	2.5922	0.5756	0.7794
Marker β - coordinata X	1.7294	0.6341	0.7626
Marker β - coordinata Y	4.0090	0.5958	0.8649
Marker γ - coordinata X	1.7693	0.5886	0.7242
Marker γ - coordinata Y	2.5547	0.5088	0.6888
Faccia 4	ε_{max} $[pixel]$	ε_{mean} $[pixel]$	σ $[pixel]$
Marker α - coordinata X	1.4898	0.5220	0.6325
Marker α - coordinata Y	3.7589	0.5433	0.7670
Marker β - coordinata X	1.6619	0.6168	0.7422
Marker β - coordinata Y	5.8566	0.6500	0.9933
Marker γ - coordinata X	2.0220	0.5838	0.7331
Marker γ - coordinata Y	4.7020	0.6560	0.9804

Tabella 4.2: Errori rilevamento coordinate su piano immagine centroidi markers

4.2.2 Risultati *patternsPositionDetect()*

Nelle *Figure 4.8,4.9,4.10* i simboli circolari riportano le coordinate spaziali stimate rispetto al riferimento *camera* per ognuno dei markers applicati al modello, sovrapposti anche qui alle linee continue che individiano le traiettorie teoriche nello spazio tridimensionale. Anche in questo caso la stima riesce a seguire ottimamente le linee ideali, ma per capire quanto, è necessario analizzare gli errori calcolati. In *Figura 4.11* gli errori stimati nelle tre coordinate X, Y, Z riguardanti, come per i risultati del *markersDetect()*, i soli markers di *tipo* α essendo del tutto simili a quelli degli altri due *tipi*.

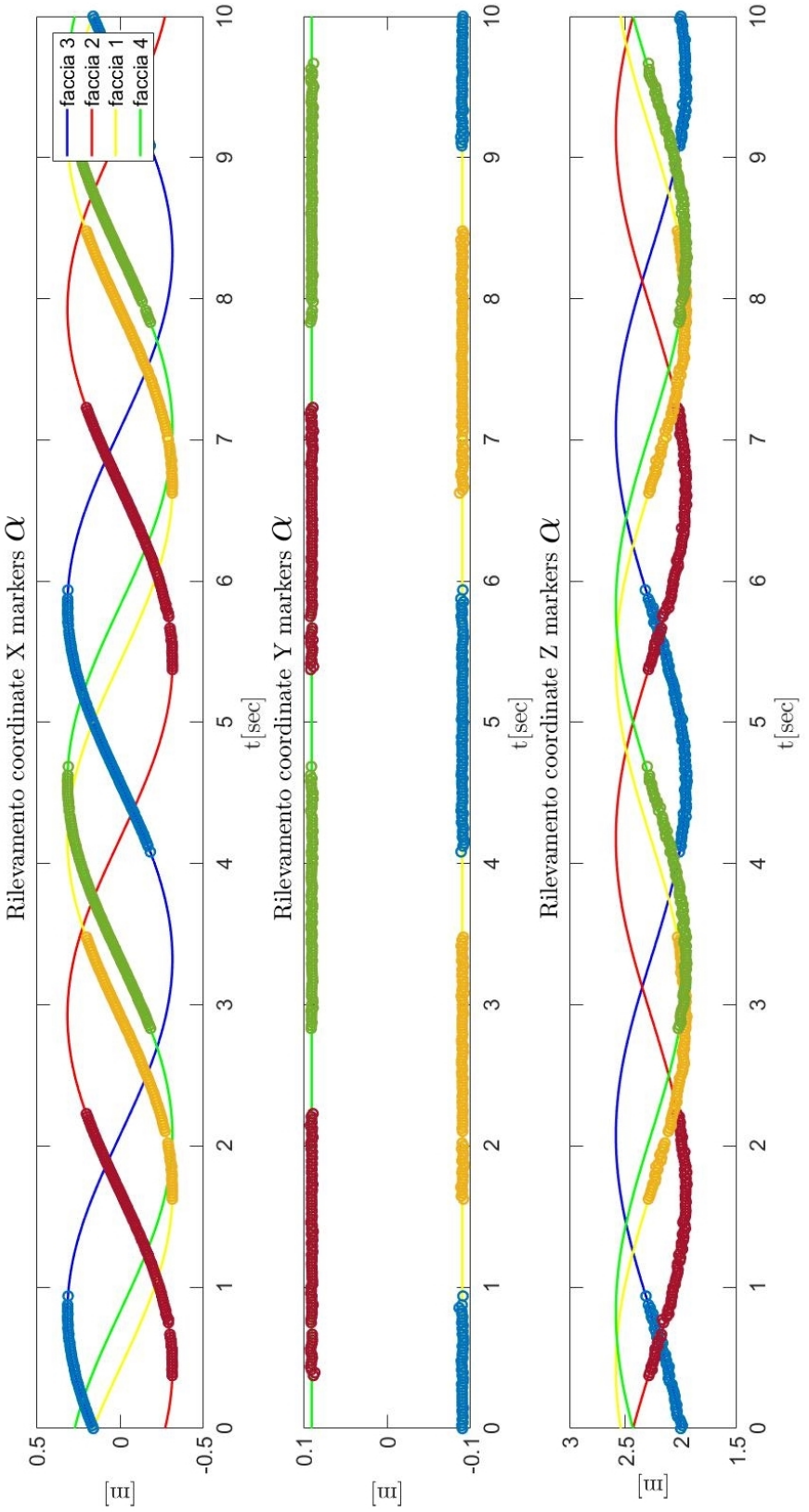


Figura 4.8: Test *rotazionale* risultati *distanceDetect()* rilievo *markers* α

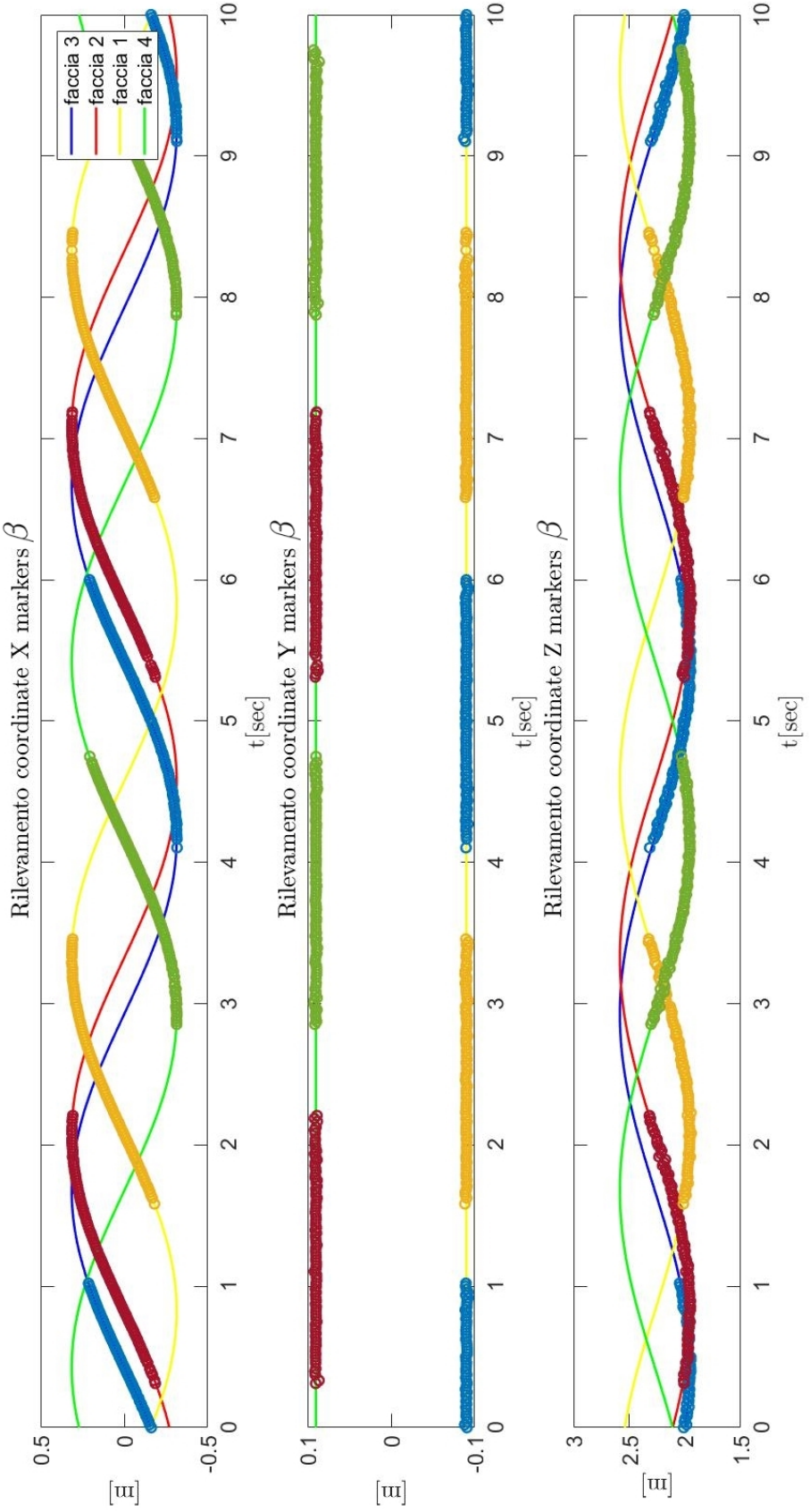


Figura 4.9: Test *rotazionale* risultati *distanceDetect()* rilievo *markers* β

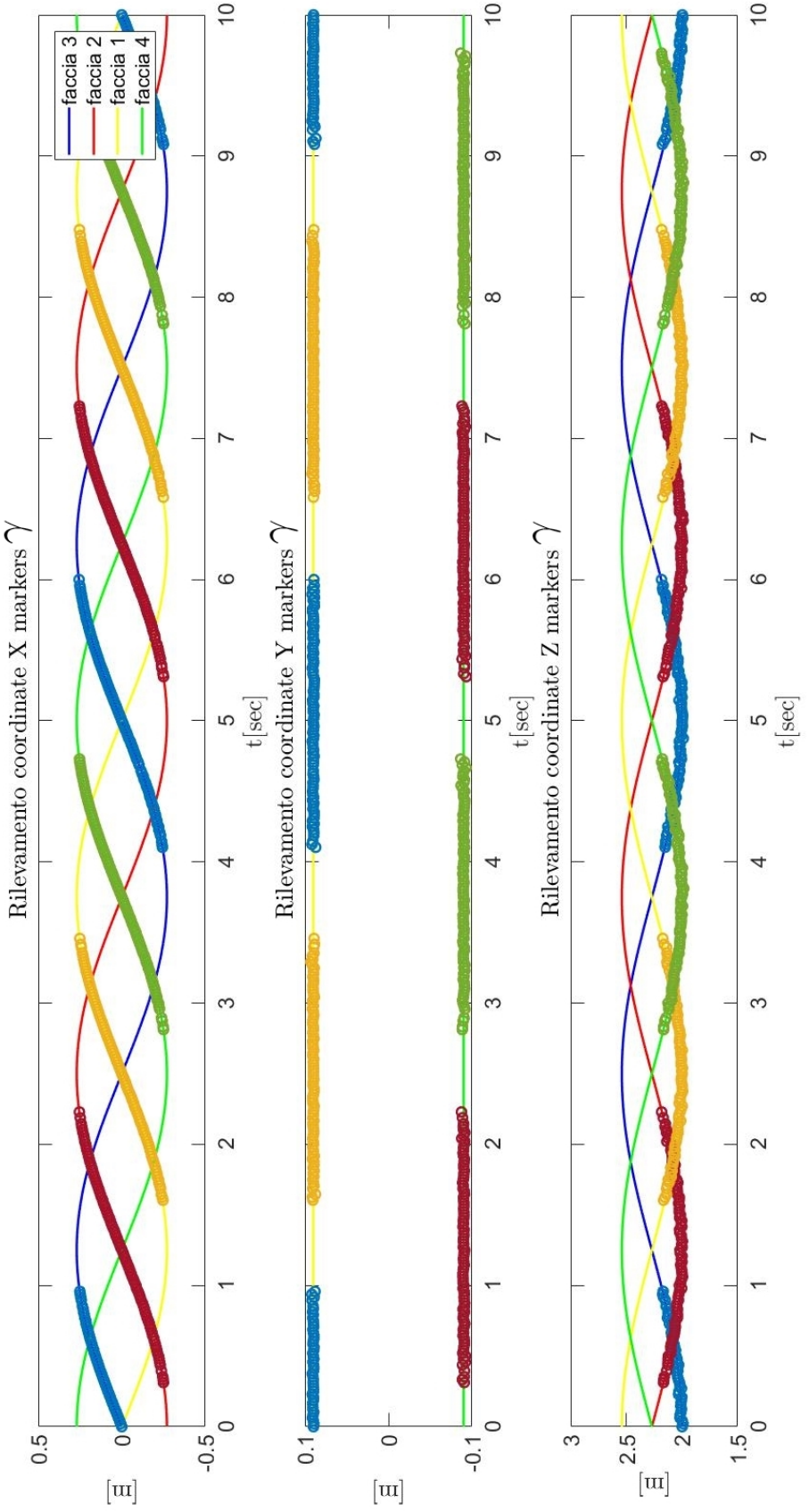


Figura 4.10: Test *rotazionale* risultati *distanceDetect()* rilievo *markers* γ

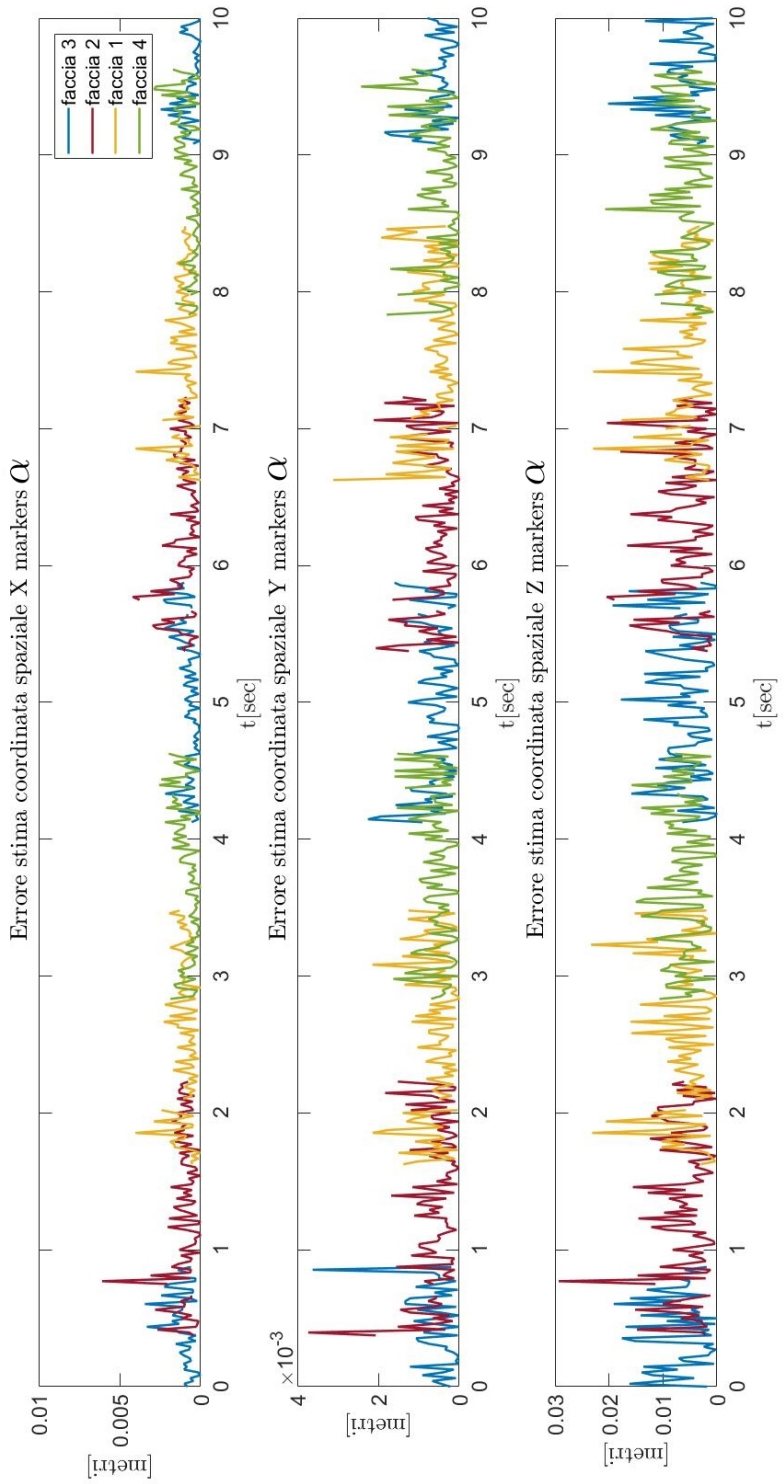


Figura 4.11: Test *rotazionale* errori stima posizione dei markers in riferimento *camera*

Si nota in figura come gli errori sulle coordinate X e Y sono molto bassi, nell'ordine dei millimetri, mentre gli errori sulla stima della profondità sono nell'ordine del centimetro. Su una distanza di circa 2 metri questo corrisponde sommariamente

ad un errore relativo molto piccolo $\epsilon_{rel} = 0.005$. Questo è un buon indicatore dell'accuratezza del processo di calcolo della posizione spaziale, ma non un indicatore significativo per la bontà dell'intero processo di stima d'assetto. Proprio come per gli errori sulla stima dei centroidi, è necessario vedere come questi errori sul calcolo della posizione in riferimento *camera* si ripercuotono sul calcolo dei parametri d'assetto. Si riportano in *tabella 4.3* anche in questo caso gli errori stimati.

4.2.3 Risultati *attitudeDetect()*

Sono riportati in *Figura 4.12* gli angoli di Eulero stimati dall'*attitudeDetect()*. Come per gli altri risultati i simboli circolari si riferiscono ai valori calcolati dal software mentre le linee continue indicano le traiettorie ideali che compiono gli angoli del sistema di *riferimento target*, linee che si intravedono appena ad esempio nel salto di dominio nello *Yaw*, questo a testimoniare la buona riuscita della stima. I valori calcolati riescono a seguire le traittorie, ma resta da stimare quanto bene effettivamente si sovrappongano andando ad analizzare gli errori. É visivamente evidente infatti come ad esempio l'angolo ψ di *Roll* risulta molto più rumoroso dell'angolo di *Pitch* e di *Yaw*, probabilmente perchè più sensibile agli errori, seppur contenuti, nel calcolo delle coordinate dei centroidi su piano immagine o nella stima della posizione tridimensionale nel *riferimento camera*. Si riportano allora in *Figura 4.13* l'andamento degli errori relativi agli angoli di Eulero.
Si conferma la rumorosità della stima dell'angolo di *Roll*, mentre si ha una buona stima dell'angolo di *Pitch* con un errore che non supera gli $0.5deg$. Nell'analisi dell'errore sull'angolo di *Yaw* si può constatare un fenomeno ricorrente, ossia degli intervalli di tempo in cui l'errore è minimo se non nullo, mentre degli istanti in cui si hanno dei picchi nell'errore e questo si vede comparire ad intervalli regolari. La spiegazione la si può trovare andando ad analizzare allo stesso modo la stima della posizione dell'origine della terna target.

Come si può notare in *Figura 4.14*, anche nella stima della posizione infatti compare un fenomeno analogo. In figura sono riportate le coordinate del centro della terna stimate in blu, sovrapposte alle linee rosse che individuano la posizione reale. La stima segue le coordinate ideali ma oltre al rumore sul calcolo della profondità già riscontrato nell'analisi del calcolo della profondità per i markers, si notano anche in questo caso i picchi ad intervalli regolari sul calcolo di X e Y, e sono proprio

Faccia 1	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
Marker α - coordinata X	0.0040	0.0010	0.0012
Marker α - coordinata Y	0.0031	0.0006	0.0008
Marker α - coordinata Z	0.0233	0.0062	0.0078
Marker β - coordinata X	0.0038	0.0007	0.0010
Marker β - coordinata Y	0.0024	0.0005	0.0007
Marker β - coordinata Z	0.0267	0.0058	0.0077
Marker γ - coordinata X	0.0030	0.0009	0.0011
Marker γ - coordinata Y	0.0030	0.0005	0.0007
Marker γ - coordinata Z	0.0184	0.0063	0.0077
Faccia 2	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
Marker α - coordinata X	0.0061	0.0010	0.0013
Marker α - coordinata Y	0.0037	0.0006	0.0008
Marker α - coordinata Z	0.0293	0.0059	0.0077
Marker β - coordinata X	0.0031	0.0008	0.0011
Marker β - coordinata Y	0.0037	0.0005	0.0008
Marker β - coordinata Z	0.0247	0.0063	0.0082
Marker γ - coordinata X	0.0025	0.0009	0.0010
Marker γ - coordinata Y	0.0032	0.0006	0.0008
Marker γ - coordinata Z	0.0191	0.0060	0.0075
Faccia 3	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
Marker α - coordinata X	0.0034	0.0008	0.0010
Marker α - coordinata Y	0.0036	0.0005	0.0007
Marker α - coordinata Z	0.0202	0.0060	0.0078
Marker β - coordinata X	0.0047	0.0011	0.0013
Marker β - coordinata Y	0.0032	0.0005	0.0007
Marker β - coordinata Z	0.0289	0.0064	0.0078
Marker γ - coordinata X	0.0028	0.0010	0.0011
Marker γ - coordinata Y	0.0031	0.0006	0.0008
Marker γ - coordinata Z	0.0216	0.0068	0.0083
Faccia 4	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
Marker α - coordinata X	0.0029	0.0007	0.0010
Marker α - coordinata Y	0.0024	0.0005	0.0007
Marker α - coordinata Z	0.0207	0.0061	0.0073
Marker β - coordinata X	0.0042	0.0012	0.0014
Marker β - coordinata Y	0.0039	0.0006	0.0009
Marker β - coordinata Z	0.0304	0.0061	0.0081
Marker γ - coordinata X	0.0027	0.0008	0.0010
Marker γ - coordinata Y	0.0035	0.0006	0.0009
Marker γ - coordinata Z	0.0198	0.0061	0.0076

Tabella 4.3: Errori rilevamento coordinate in *riferimento camera* dei patterns

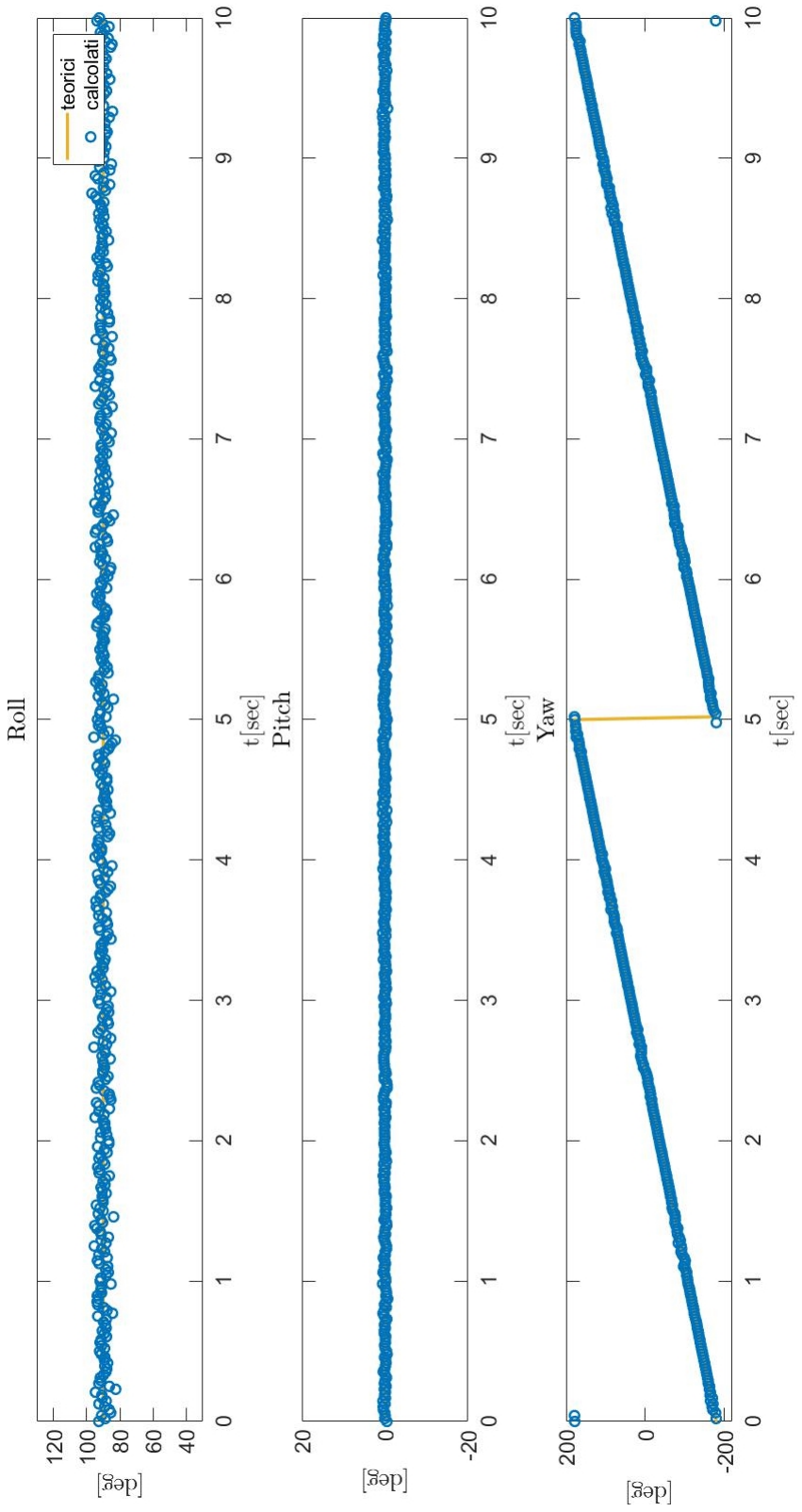


Figura 4.12: Test *rotazionale* angoli di Eulero risultati *distanceDetect()*

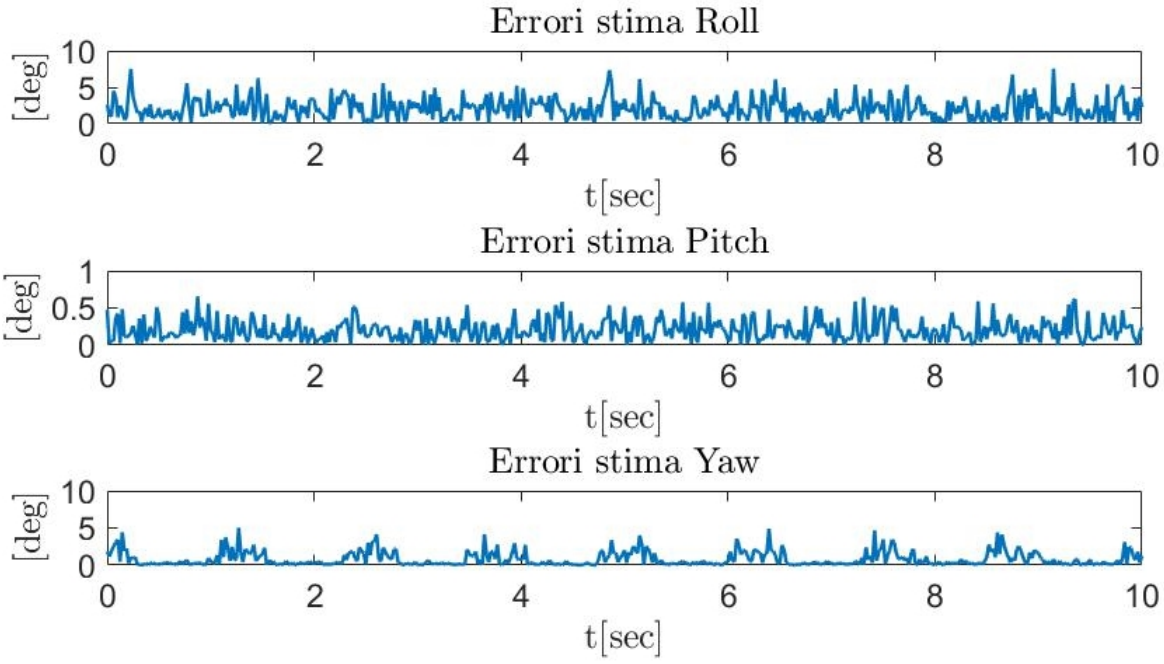


Figura 4.13: Test *rotazionale* errori stima angoli *Roll,Pitch,Yaw*

tali picchi ad influire sull'errore nella stima degli angoli, essendo questi strettamente correlati alla posizione dell'origine. Ma se gli errori sulle coordinate del centro influiscono sull'errore nella stima degli angoli di Eulero, bisogna capire cosa genera tali errori.

La spiegazione è da ricercare nel particolare metodo di calcolo dell'assetto nel caso in cui si stiano vedendo soli 3 markers, ovvero quei casi in cui la stereocamera è frontale o quasi ad una singola faccia. In *Figura 4.14* è infatti riportato in ultima riga un grafico che indica istante per istante quanti markers sono stati individuati correttamente e comparando questo con i grafici soprastanti risulta evidente che i picchi di errore si hanno in corrispondenza degli intervalli in cui sono elaborati tre soli markers. Il motivo di tale fenomeno è legato al metodo spiegato in *Sezione 3.4.2* che risulta essere efficace ma molto sensibile al rumore di misura.

Aiutati dalla *Figura 4.15* si può intuire come piccoli errori nella stima della distanza possano portare ad un errore di un angolo di alcuni gradi nella stima del coseno direttore della faccia, versore che viene utilizzato per stimare il centro della terna target. Ma un errore di pochi gradi se propagato per una distanza di 0.27 metri, quale la distanza tra la faccia e il centro di massa coincidente con il centro del riferimento *target*, può portare ad un errore di diversi centimetri nella stima. Si riportano in *Tabella 4.4* gli errori nel calcolo di tali coordinate, errori in media accettabili ma con picchi di errore rilevanti, soprattutto per quanto concerne la profondità.

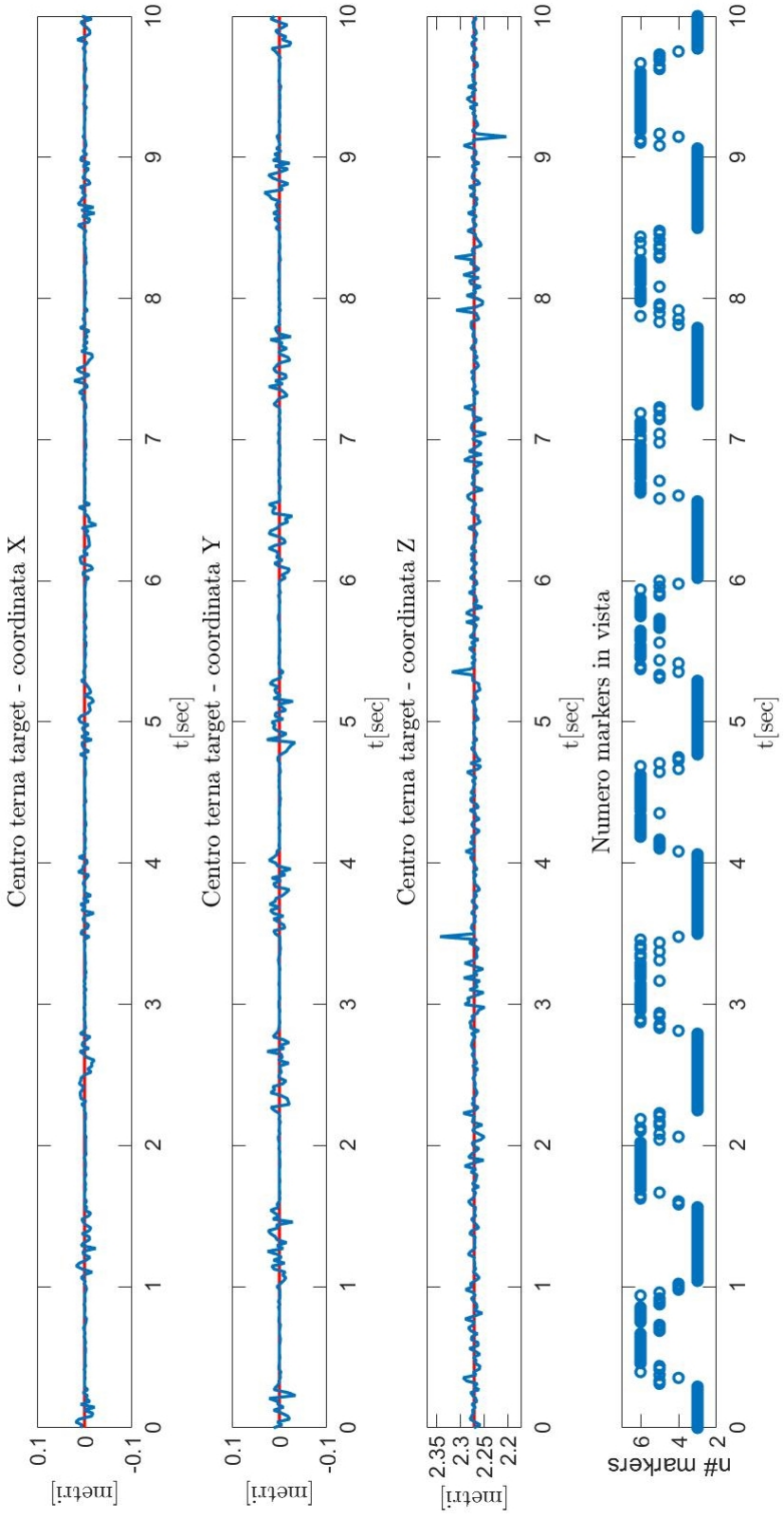


Figura 4.14: Test *rotazionale* stima della posizione dell'origine del *riferimento target*

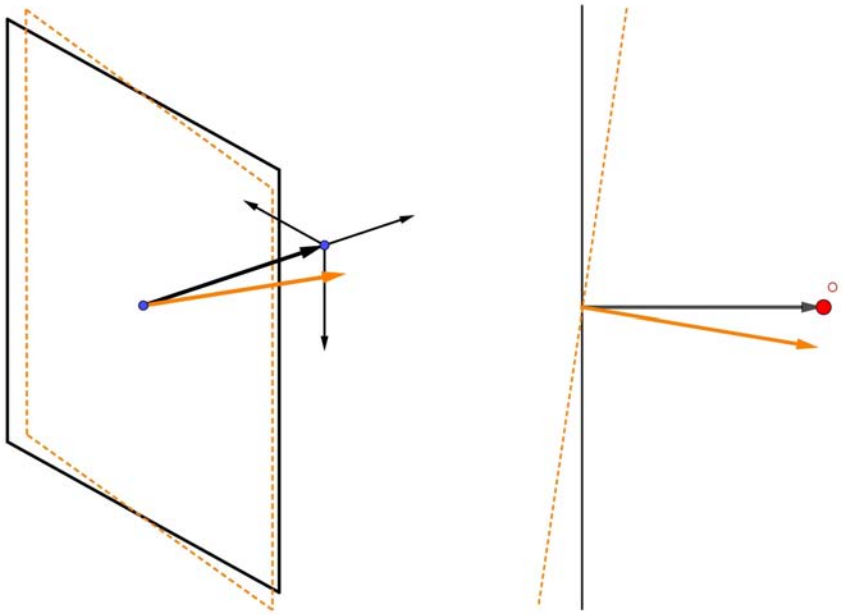


Figura 4.15: Errore legato al metodo usato per la stima dell'assetto nel caso di 3 markers in vista

Target Origin	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
X_c	0.0246	0.0040	0.0062
Y_c	0.0338	0.0054	0.0089
Z_c	0.0720	0.0061	0.0094

Tabella 4.4: Errori rilevamento coordinate centro target

Presentiamo infine in *Tabella 4.5* gli errori ottenuti nella stima dei parametri d'assetto come conseguenza della catena di errori ottenuti nei diversi passaggi, dal calcolo delle coordinate dei centroidi su piano immagine, alla stima delle coordinate in riferimento *camera* delle features individuate e non ultimo sulle discusse coordinate di posizione del *target*.

Angoli di Eulero	ε_{max} $[deg]$	ε_{mean} $[deg]$	σ $[deg]$
$Roll$	7.5665	2.0441	2.5330
$Pitch$	0.6570	0.2016	0.2495
Yaw	5.0391	0.8090	1.2684

Tabella 4.5: Errori nella stima dei parametri d'assetto *Roll,Pitch,Yaw*

A conferma numerica di quanto si denota in *Figura 4.13* l'errore sulla stima di *Roll*

è il più rilevante, oltre che come picchi anche come valori medi. La stima di *Pitch* si conferma la più accurata con un errore medio inferiore al mezzo grado sessagesimale, ed anche la stima di *Yaw* pur presentando dei picchi d'errore non bassi risulta buona con un errore medio sotto il grado.

Risultati stima velocità angolare

La stima delle componenti della velocità angolare non è integrata nel software sviluppato e per questo non è stata nominata nel capitolo 3. É un dato però ottenuto in postprocessing utilizzando la semplice derivata temporale degli angoli ψ, θ, ϕ stimati dall'*attitudeDetect()* ed analizzati nella corrente sezione. Il problema riscontrato nel fare questo è dato dal fatto che i video test sviluppati in Solidworks, sono stati creati a 48 fps, con l'obiettivo di analizzare quanti più frame possibile in modo da avere una maggiore quantità di dati da poter analizzare. Questo implica una costante di variazione temporale tra un frame e l'altro, e quindi anche tra una stima d'assetto e l'altra, di $\frac{1}{48}$ di secondo. E questo porta ad una notevole sensibilità al rumore presente nella stima degli angoli. Essendo infatti

$$\dot{\psi} = \frac{\Delta\psi}{1/48} \tag{4.1}$$

$$\dot{\theta} = \frac{\Delta\theta}{1/48} \tag{4.2}$$

$$\dot{\phi} = \frac{\Delta\phi}{1/48} \tag{4.3}$$

un piccolo errore nel calcolo di una variazione angolare viene moltiplicato per 48 e quindi ampiamente incrementato. L'idea per risolvere tale problema nasce dal fatto che una frequenza di 48 fps in un futuro uso online del software non è plausibile, essendo incompatibile con i tempi di processo di un singolo frame. Si sceglie allora di calcolare la velocità angolare con lo stesso metodo, ma abbassando la frequenza di campionamento nella variazione d'angolo. Si riportano ad esempio in *Figura 4.16* i risultati che si ottengono campionando ogni mezzo secondo, ricordando che le componenti della velocità angolare teorica per il video sono $\mathbf{W} = [W_x, W_y, W_z] = [\dot{\psi}, \dot{\theta}, \dot{\phi}] = [0, 0, 1.2566]$.
Si nota che la più rumorosa risulta la componente W_x come diretta conseguenza degli errori sugli angoli di *Roll*. Numericamente il calcolo delle componenti della velocità angolare presenta gli errori in *Tabella 4.6*.

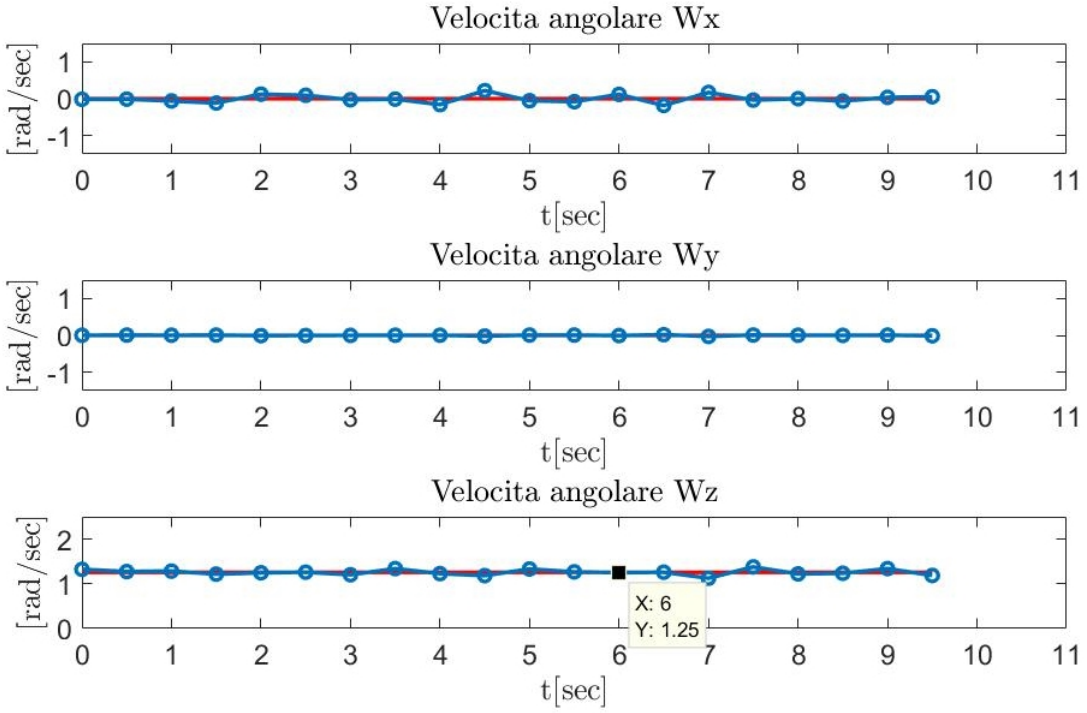


Figura 4.16: Stima *velocità angolare* con campionamento a 0.5 sec

Velocità angolare	ε_{max} $[rad/sec]$	ε_{mean} $[rad/sec]$	σ $[rad/sec]$
W_x	0.2202	0.0034	0.0210
W_y	0.0282	0.0003	0.0021
W_z	0.1316	0.0020	0.0124

Tabella 4.6: Errori nella stima della *Velocità angolare*

4.3 Risultati test *rototraslazionale*

Si presentano in questa sezione i risultati relativi al video test comprensivo di un moto rotazionale ed un moto traslazionale. Le prime due fasi di processo, quindi il *markersDetect()* ed il *patternsPositionDetect()*, hanno avuto anche in questo caso buon esito e graficamente i dati stimati ricalcano bene le linee ideali tracciate. Per questo motivo nelle due sezioni seguenti ci si limita a descrivere come variano gli errori in funzione del movimento aggiuntivo o del variare della distanza, senza riproporre tutti i risultati grafici, che potendo però essere d'interesse, possono essere visionati in appendice alla tesi.

4.3.1 Risultati *markersDetect()* e *patternsPositionDetect()*

Si presentano in questa sezione gli errori ottenuti nei primi due step del test rototraslazionale, ovvero il riconoscimento dei markers ed il calcolo della loro posizione spaziale in terna *camera*. Nell'elaborazione d'immagine di un singolo frame, per il processo di stima dell'assetto, poco influisce se in quel dato istante il *target* sia o meno in movimento, per cui quello che ci si può aspettare di diverso nei risultati del test rototraslazionale rispetto a quelli del primo test rotazionale, è che si ripercuota sugli errori l'aumento della distanza del *target* dalla *camera*, si ricorda infatti che mentre nel primo video test il modello virtuale del modulo d'assetto era ad una posizione di $2.27 metri$ dalla camera, in questo secondo video test parte ad una distanza di $3.27 metri$. In linea di concetto, se i centroidi dei markers venissero rilevati in modo preciso, questo non dovrebbe influire sull'algoritmo di stima della distanza. Ma all'aumentare della distanza diminuisce la definizione grafica dei markers e questo può influire sulla precisione con cui i centroidi vengono rilevati.
Per non presentare al lettore un eccessivo quantitativo di ulteriori numeri, si presentano gli errori medi per generico *tipo* di marker, senza specificarne la faccia di appartenenza, in modo da poter notare meglio una possibile variazione o costanza nel trend rispetto al caso rotazionale.

Marker α	ε_{max} $[pixel]$	ε_{mean} $[pixel]$	σ $[pixel]$
X	1.7919	0.5393	0.6665
Y	2.5551	0.7290	0.8855
Marker β	ε_{max} $[pixel]$	ε_{mean} $[pixel]$	σ $[pixel]$
X	1.9965	0.6034	0.7251
Y	3.2904	0.6941	0.8742
Marker γ	ε_{max} $[pixel]$	ε_{mean} $[pixel]$	σ $[pixel]$
X	2.0142	0.5982	0.7288
Y	2.4474	0.6863	0.8288

Tabella 4.7: Errori rilevamento centroidi nel caso rototraslazionale

Confrontando questi errori con quelli in *Tabella 4.2* non sembrerebbe esserci una vistosa variazione nel trend, riconfermando come l'errore sia generalmente più ampio sulla coordinata y rispetto a quello sulla coordinata x.
Presentiamo allora in *tabella 4.7* gli errori ottenuti per il calcolo della posizione dei markers in terna *camera*, anche in questo caso ridotti alla media dei valori per il

generico *tipo*.

Marker α	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
X	0.0069	0.0014	0.0019
Y	0.0035	0.0010	0.0013
Z	0.0395	0.0132	0.0158
Marker β	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
X	0.0075	0.0014	0.0018
Y	0.0039	0.0011	0.0013
Z	0.0532	0.0129	0.0180
Marker γ	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
X	0.0063	0.0014	0.0017
Y	0.0041	0.0011	0.0013
Z	0.0587	0.0142	0.0184

Tabella 4.8: Errori nella stima della posizione dei markers in terna *camera* in test rototraslazionale

Confrontando questi risultati con quelli del primo test in *Tabella 4.3* si nota come i valori delle coordinate X e Y siano rimasti della stessa ampiezza, mentre si riscontra un aumento rilevante per gli errori della coordinata di profondità sia per quanto riguarda il picco massimo che per quanto riguarda la media generale. Non essendo variati così vistosamente i valori degli errori sui centroidi, questo non è imputabile ad un peggioramento della definizione grafica dei markers. Un'ipotesi plausibile invece è da ricercare in un fenomeno simile a quello descritto in *Sezione 4.2.3* ed in particolare in *Figura 4.15* discutendo la propagazione dell'errore nel calcolo della posa. La coordinata Z è infatti ottenuta intersecando due rette nello spazio (*Sezione 3.4.1*), ed un errore anche minimo del calcolo dei centroidi, che può portare ad un piccolo errore nell'inclinazione delle rette, viene tanto propagato quanto più aumenta la distanza dell'intersezione.

4.3.2 Risultati *attitudeDetect()*

Pur riuscendo anche in questo caso a seguire l'andamento ideale di *Roll, Pitch e Yaw*, l'aumento dell'errore che si riscontra nella stima della profondità influisce sull'aumento dell'errore nella misura degli angoli. Si propone infatti il trend grafico in *Figura 4.17* dove si riscontra immediatamente l'aumento d'errore soprattutto per

il più rumoroso dei tre angoli ossia quello di *Roll*.

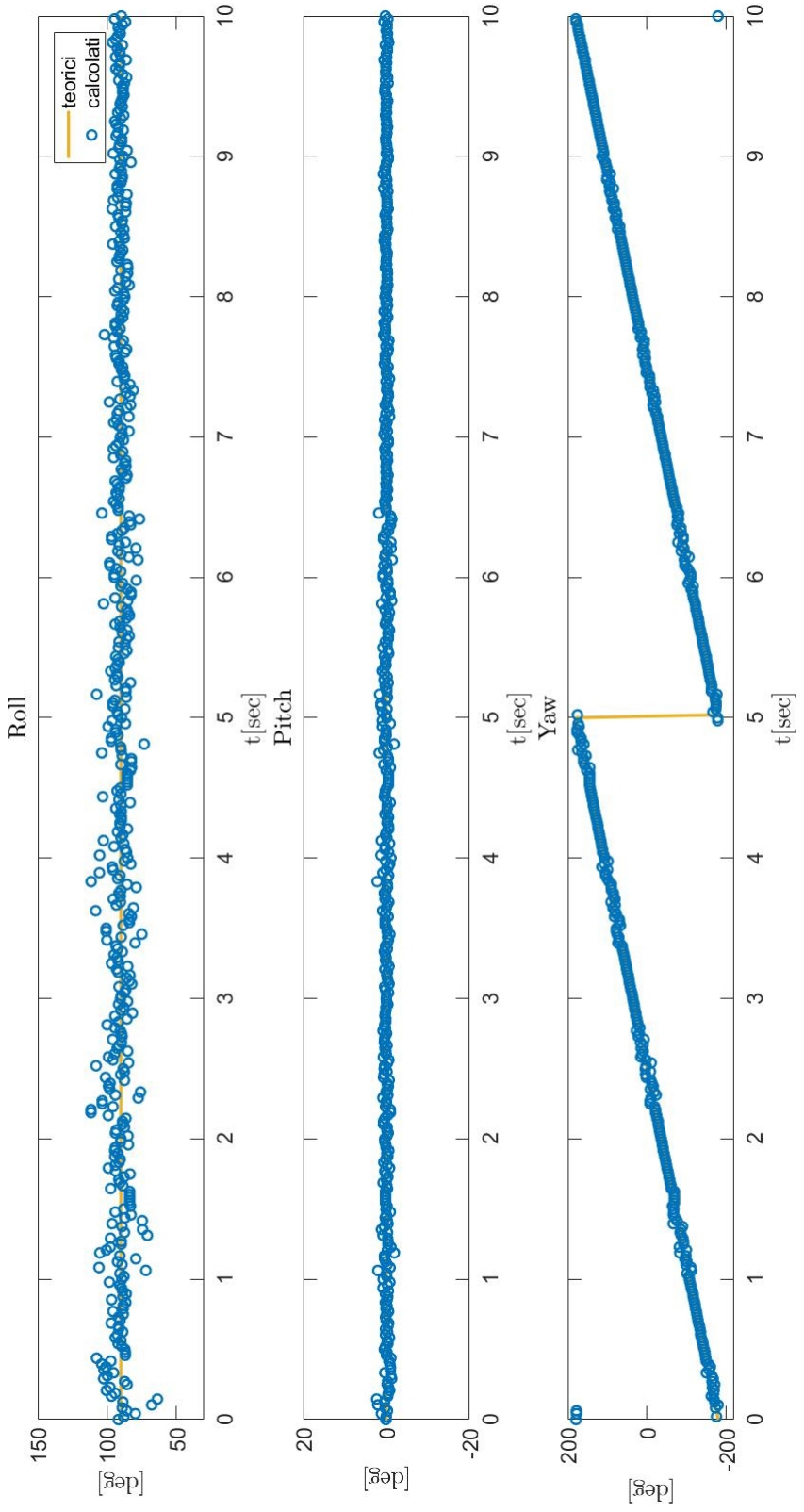


Figura 4.17: Test *rotraslazionale* angoli di Eulero

Si può notare però come scorrendo lungo le ascisse, ossia al passare del tempo, l'errore vada diminuendo, coincide infatti con l'avvicinarsi del *target* alla camera e la diminuizione quindi della propagazione d'errore sul calcolo della posizione. Questo fenomeno è meglio visibile andando a graficare in *Figura 4.18* l'andamento degli errori sul calcolo degli angoli.

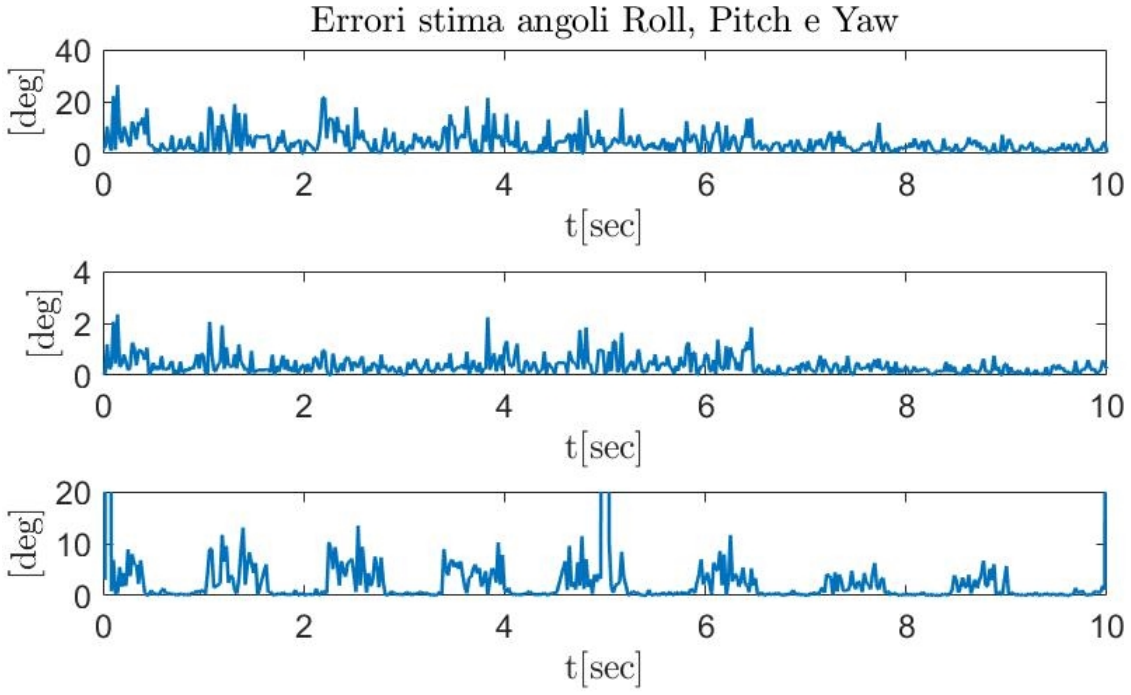


Figura 4.18: Errori su *Roll, Pitch e Yaw* nel caso *Rototraslazionale*

Ben si distingue la diminuizione progressiva soprattutto sui picchi d'errore dell'angolo di Yaw, dove nei primi 5 secondi toccano punte di $10deg$ ed arrivando via via a picchi più simili a quelli del primo video test intorno ai $5deg$.

Questo porta a fare una prima considerazione finale sulla bontà del processo di stima, ossia che risulta in ogni caso efficace ma come ci si poteva aspettare l'efficienza e la precisione aumentano al diminuire della distanza del *target* dalla stereocamera.

4.4 Test in Laboratorio

Vengono presentati in questo paragrafo i risultati dei test eseguiti in laboratorio con il set up descritto in *Capitolo 2*. In particolare i test sono stati condotti nei laboratori del Dipartimento di Ingegneria Industriale dell'Università degli Studi di Padova.
Va precisato che le ipotesi poste per i test in ambiente virtuale descritte in *Sezione 4.1* sono da ritenersi valide anche in questo ambito. Ci si limita perciò ad un moto puramentre rotazionale azionato manualmente, senza per questo perdere di generalità. Per questo motivo il modulo d'assetto del micro satellite *target* è posto tramite il giunto rotazionale a tre assi si di una base rigida invece che al proprio modulo traslazionale (*Figura 4.19*). La stereocamera DUO M è a sua volta montata su di una struttura in alluminio a simulare il micro satellite *inseguitore*.

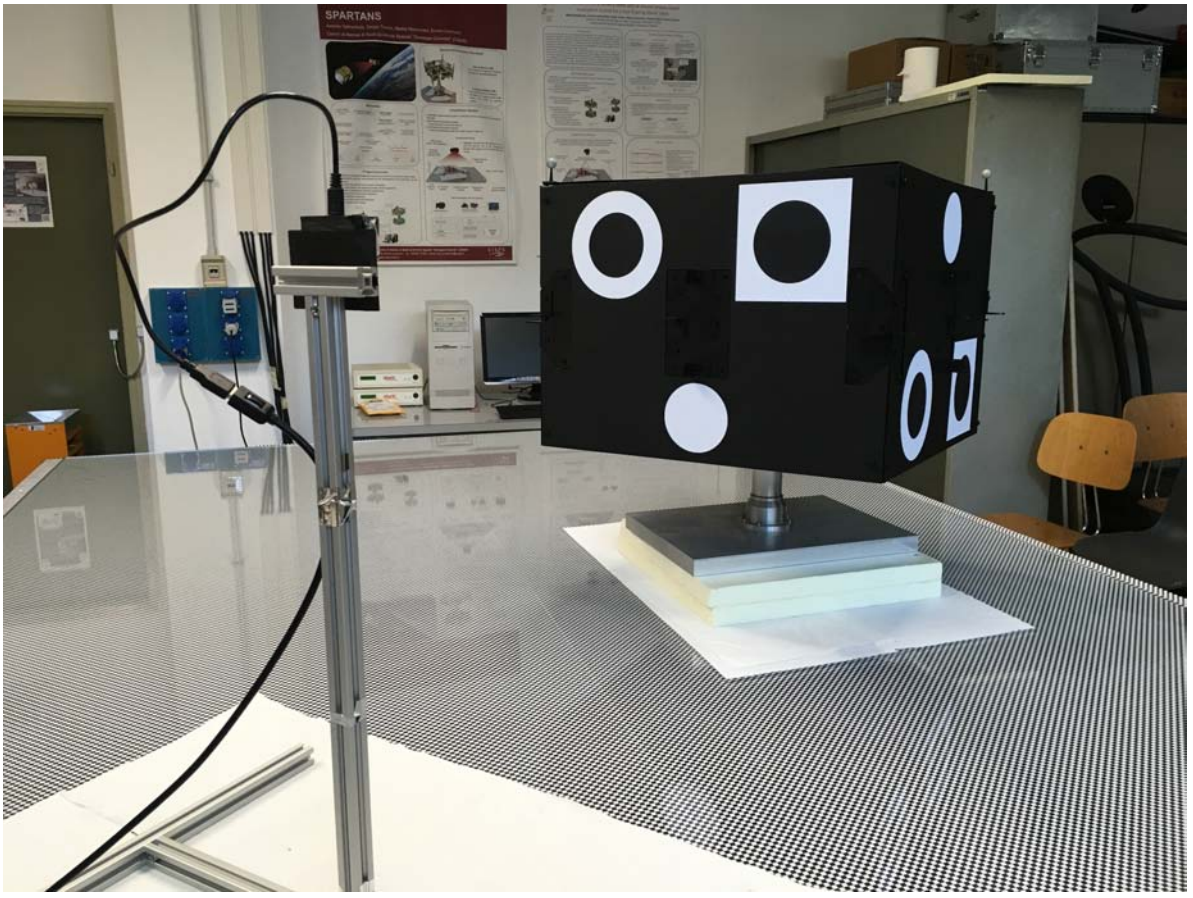


Figura 4.19: Acquisizione video test in laboratorio

Descrivendo la catena di misura:

- la stereocamera è collegata tramite USB ad una scheda di acquisizione a sua

volta collegato ad un monitor attraverso il quale è possibile configurare le impostazioni della DUO M e lanciare un'acquisizione;

- il GNS (*Sezione 2.1.3*) è collegato ad una postazione PC attraverso la quale è possibile configurarlo, lanciare un'acquisizione e in un secondo momento ottenere i dati di misura;

- si fanno partire entrambe le acquisizioni e si impone un moto al modulo d'assetto del *target*;

- si fermano le acquisizioni e una volta ottenuti i dati da entrambe le postazioni li si sincronizza per il confronto di misura.

L'analisi dei dati è fatta quindi in fase di postprocessing.
Si procede allora nelle sezioni a seguire a presentare i risultati ottenuti in termini di calcolo della posizione dei markers in riferimento *camera* ed il calcolo di posa ed assetto. Si tralasciano volutamente i risultati sul rilvamento dei centroidi trovandoli poco significativi, in quanto andrebbero confrontati con diverse ricostruzioni proiettive derivanti dai dati ottenuti dal GNS e affette quindi a loro volta dalla propagazione d'errore, pur essendo le misure del GNS sicuramente più accurate di quelle ottenute dalla stereocamera.

4.4.1 Risultati patternsPositionDetect()

Vengono presentati in questa sezione i risultati sulla stima delle posizioni in riferimento *camera* dei markers applicati al modulo d'assetto del micro satellite. Nelle *Figure 4.20 - 4.21 - 4.22* sono graficati i risultati suddivisi per *tipo* di marker. I simboli circolari indicano i valori stimati dal software di ricostruzione implementato, mentre le linee continue indicano le traiettorie ricostruite dai rilevamenti fatti attraverso il GNS e che usiamo come riferimento ideale.
Analizzando le immagini, pur essendo visivamente più rumorose, le stime riescono a seguire bene le traiettorie del GNS anche nel caso reale. Per capire quanto bene si presentano in *Tabella 4.9* gli errori rilevati per *tipo* di marker e per ogni coordinata spaziale.
Risulta evidente un rilevante aumento degli errori calcolati a confronto con i risultati del test in ambiente virtuale (*Tabella 4.3*), sia in termini di picchi massimi che per quanto riguarda l'errore medio, in un intervallo compreso tra gli 1 e i 2 cm.
A questo è possibile dare una spiegazione frutto di diversi motivi:

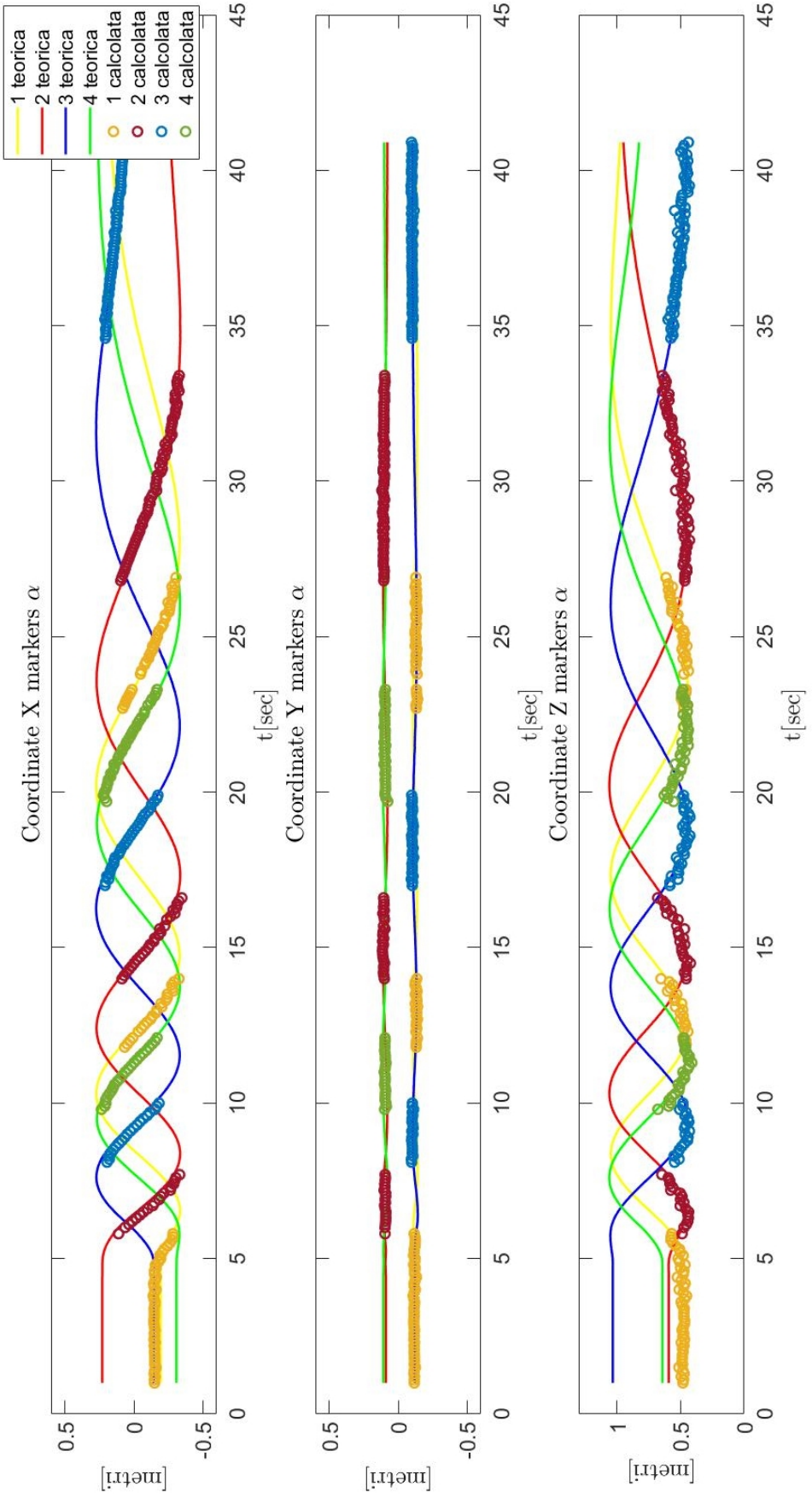


Figura 4.20: Test laboratorio posizioni in terna *camera* markers *tipo* α

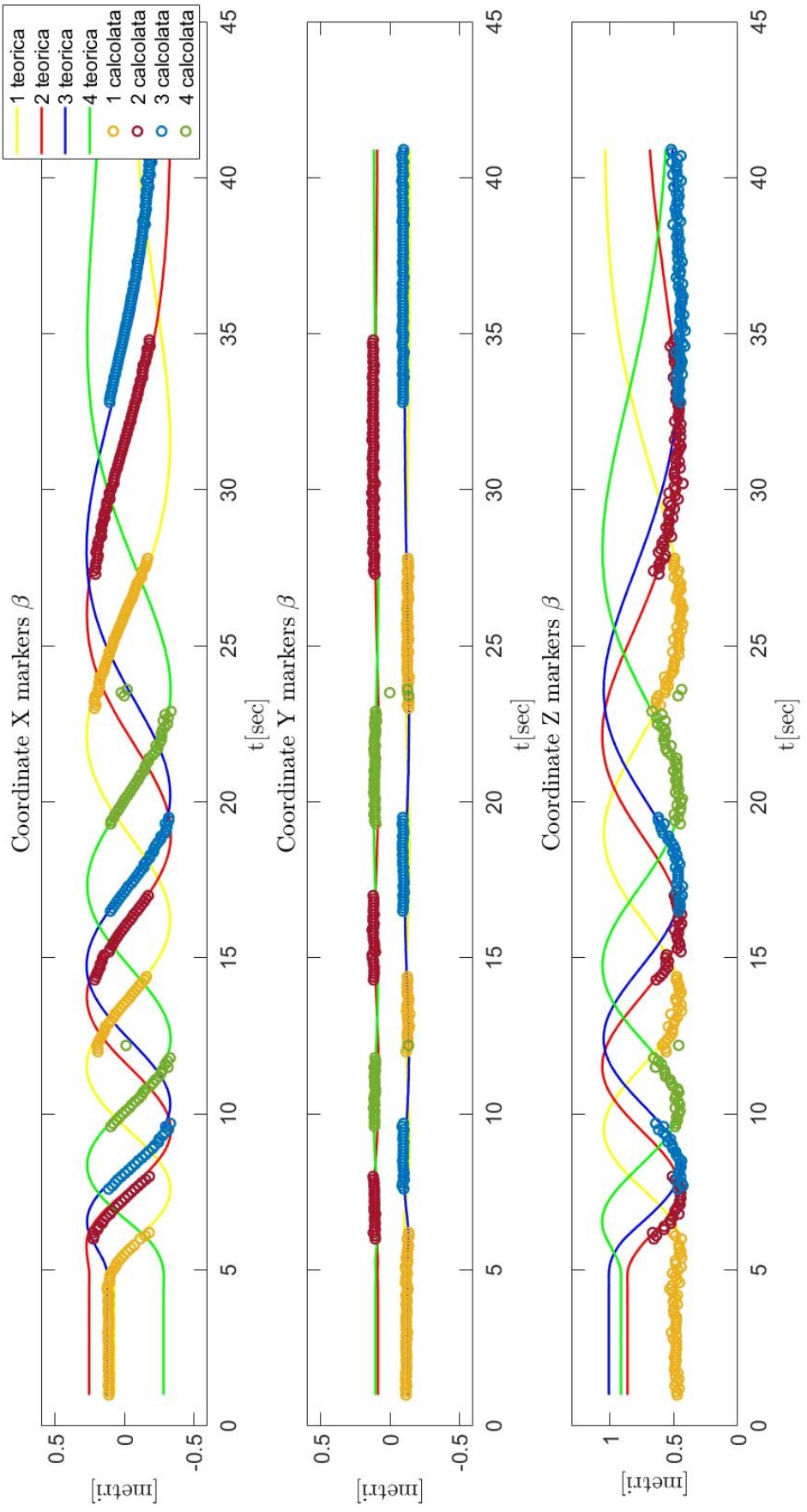


Figura 4.21: Test laboratorio posizioni in terna *camera* markers *tipo* β

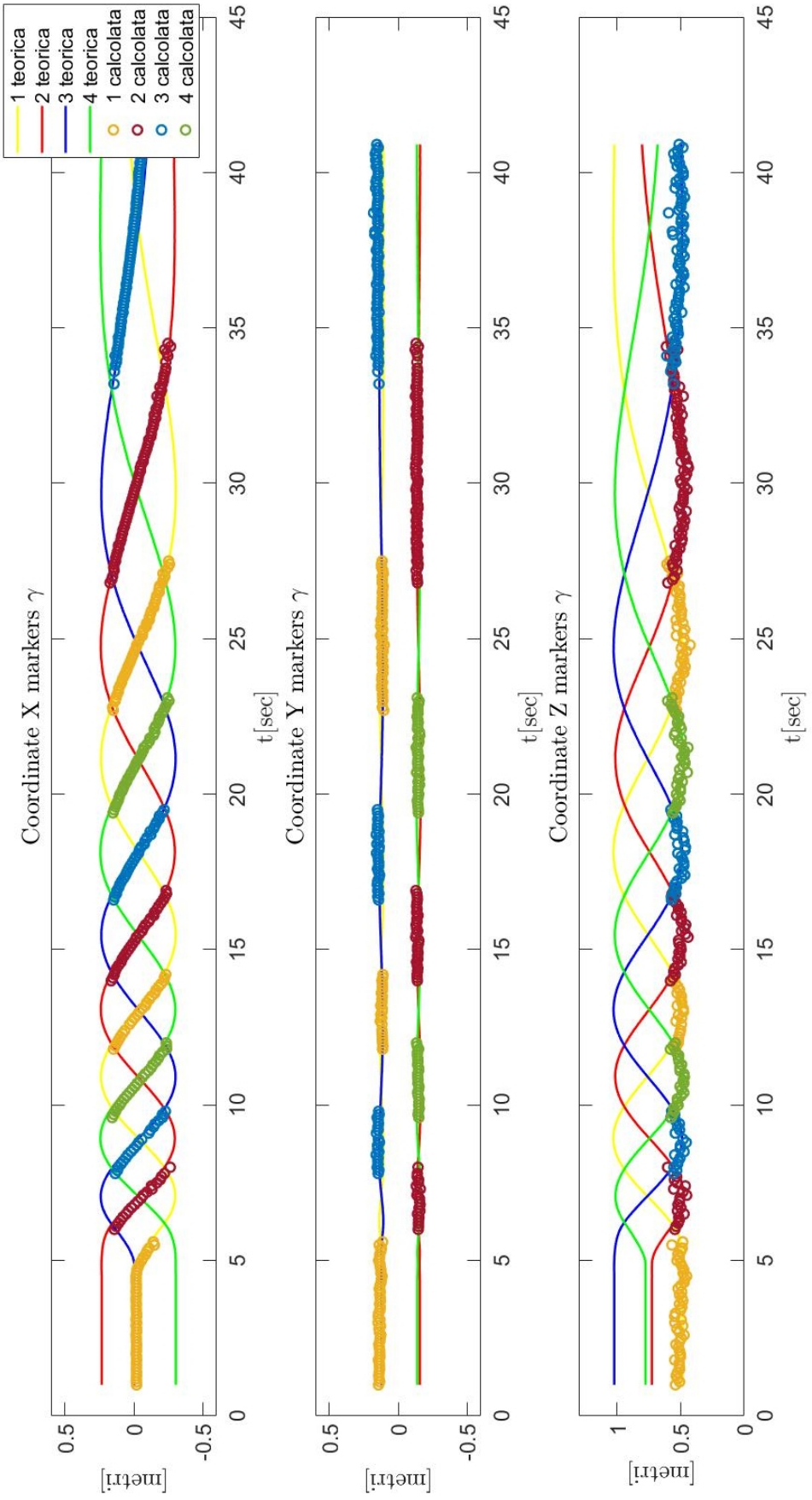


Figura 4.22: Test laboratorio posizioni in terna *camera* markers *tipo* γ

Marker α	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
X	0.0542	0.0259	0.0303
Y	0.0335	0.0111	0.0145
Z	0.0592	0.0176	0.0220
Marker β	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
X	0.0626	0.0261	0.0293
Y	0.0309	0.0122	0.0139
Z	0.0841	0.0167	0.0223
Marker γ	ε_{max} $[metri]$	ε_{mean} $[metri]$	σ $[metri]$
X	0.0432	0.0205	0.0234
Y	0.0317	0.0131	0.0145
Z	0.0742	0.0215	0.0215

Tabella 4.9: Errori nella stima della posizione dei markers in terna *camera* in test di laboratorio

- la risoluzione della stereocamera DUO M è più bassa della risoluzione utilizzata per i video Solidworks, 640×480 contro 800×618, il che rende le figure meno definite accetuando il rumore di misura sui centroidi;

- le condizioni di luminosità dell'ambiente virtuale erano ideali, diversamente da quelle dell'ambiente reale di laboratorio;

- gli stessi materiali esterni, pur essendo di egual colore hanno proprietà riflettenti differenti tra il caso reale ed il modello virtuale;

- pur essendo certamente più accurate e veritiere delle traiettorie calcolate, le traiettorie ideali derivanti dal GNS sono frutto di una ricostruzione tramite una trasformazione del sistema di riferimento, che può aver influito a sporcare le misure;

- ultimo ma non ultimo, anzi forse motivo di più rilievo, l'errore costruttivo di cui sono affette le pareti esterne del modulo d'assetto e soprattutto di cui è affetta l'applicazione manuale dei markers sulle stesse. Questi due aspetti insieme influiscono sulle reali coordinate in terna *target* dei markers e quindi sul processo di stima.

A fronte della somma e propagazione di tutti questi errori insieme, un errore medio in un intervallo $[1, 2]cm$ è da ritenersi accettabile per questo primo lavoro di calcolo diretto.

4.4.2 Risultati attitudeDetect()

Sono presentati in *Figura 4.9* i parametri d'assetto stimati. I simboli circolari rappresentano gli angoli calcolati mentre le linee continue le traiettorie ricostruite dalle misurazionei provenienti dal GNS. Qualitativamente le rilevazioni risultano seguire le traiettorie di confronto, ma come conseguenza degli errori di misura ottenuti nella stima della posizione spaziale dei markers, essi risultano numericamente meno accurati rispetto al caso del video virtuale. Gli errori rispetto alle linee teoriche sono riportati in *tabella* .

Angoli di Eulero	ε_{mean} $[deg]$	σ $[deg]$
$Roll$	4.7446	5.8933
$Pitch$	2.0252	2.4126
Yaw	7.0034	8.6590

Tabella 4.10: Errori nella stima degli angoli di Eulero in test di laboratorio

Pur accostandosi bene alla linea teorica, l'angolo di *Yaw* risulta essere il più rumoroso, mentre si conferma il più accurato l'angolo di *Pitch*. Considerando questo come l'output di una lunga catena di processi di calcolo e somma quindi di una vasta propagazione di errori, è da ritenersi come primo lavoro un buon risultato. Ma a questo punto della trattazione ha più senso andare a fare un ultima analisi generale sul processo di stima, andando a svolgere un'analisi d'incertennza descritta infatti in *Sezione 4.5*.

4.5 Analisi d'incertezza

Nelle sezioni precedenti del presente capitolo, sono stati presentati ed analizzati i risultati del processo di stima dell'assetto per diversi test, svolti sia in ambiente virtuale che in laboratorio. Quello che si vuole fare in questa sezione è però dare una stima generale sulla bontà del software sviluppato, indipendente dalle condizioni particolari dei test svolti.
Si decide allora di andare a svolgere un'*analisi di Montecarlo* per ottenere una banda

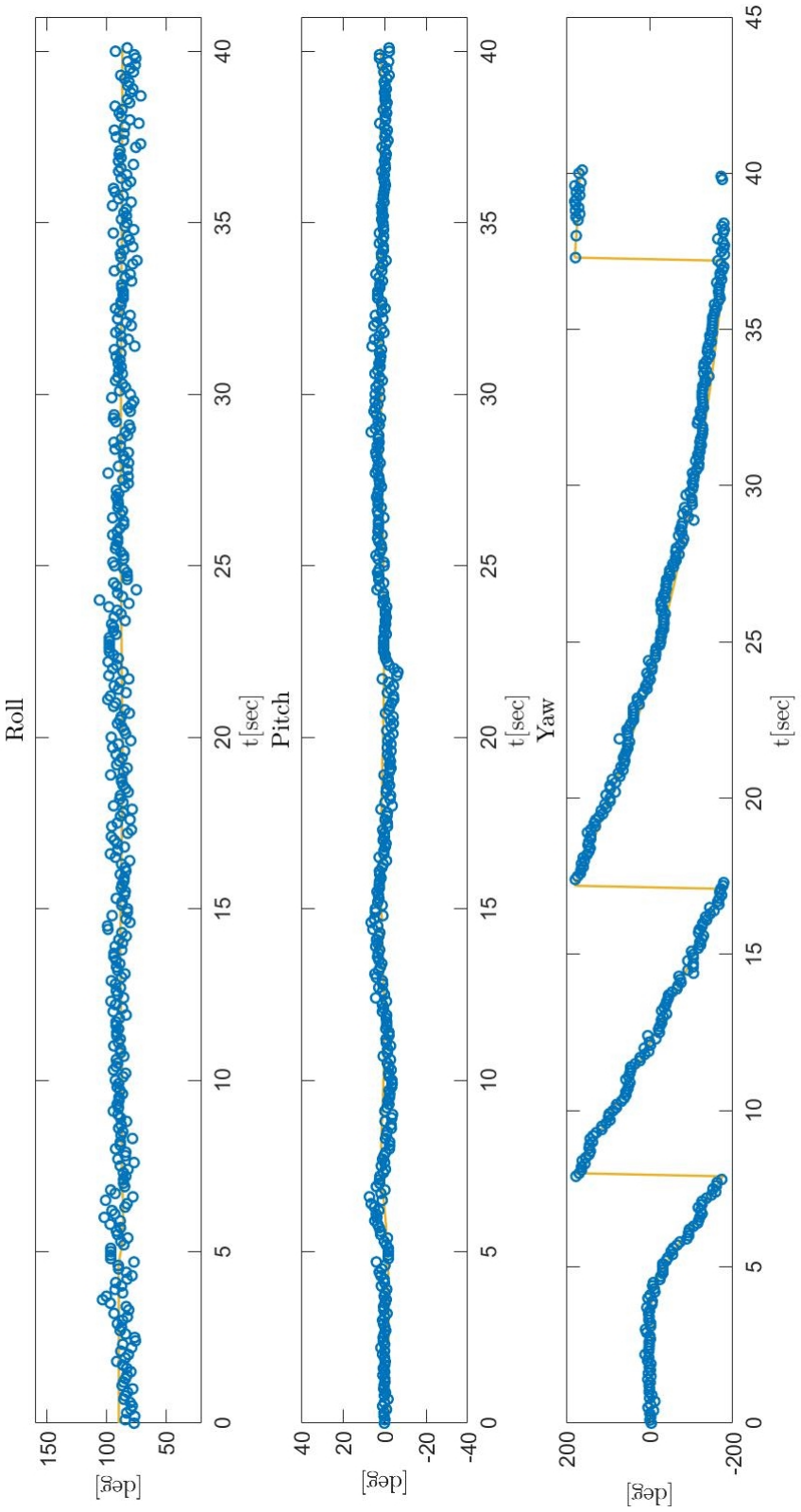


Figura 4.23: Test in laboratorio risultati angoli di Eulero

d'incertezza al variare del rumore sui dati in input. Nella pratica quello che si fa è generare un dataset di coordinate ideali riferite ad una data famiglia di markers nel compimento di una traiettoria ideale prefissata, quindi sporcarle con un rumore gaussiano e darle in ingresso al software di stima di posizione e assetto per valutarne gli errori confrontando l'output generato con un output teorico. Per poter coprire l'intera banda d'errore in modo accettabile si sceglie di processare un dataset con un certo valore di scarto quadratico medio per un numero di 1000 iterazioni.

L'analisi d'incertezza tiene conto dell'intero processo di calcolo, a partire dagli algoritmi di riconosicmento dei markers fino alla stima dell'assetto. Ma prima di procedere, si vuole poter capire quanto gli algoritmi di ricostruzione della distanza e ricostruzione dei parametri d'assetto siano effettivamente efficienti, per poter stimare quanto effettivamente l'errore dipenda da questi o dalla fase di elaborazione d'immagine. Per fare questo, prima di procedere a dare in ingresso il dataset sporcato del rumore, si da in ingresso il dataset di traiettorie ideali, dal quale ci si aspetta di ottenere un errore il più vicino possibile allo zero. Sono presentati in *Figura 4.24 - 4.25* i risultati. Con un errore inferiore al mezzo grado sessagesimale sulla stima di *Roll,Pitch,Yaw* ed un errore dell'ordine di grandezza del *micrometro* si può considerare efficiente il software di calcolo delle distanze posizionali e dei parametri d'assetto.
Questo conferma come l'errore sia dato principalmente dal rumore di misura sul rilevamento dei centroidi dei markers, quindi sulla parte del processo che si occupa dell'elaborazione d'immagine. Per capire il grado di dipendenza si sono quindi svolte quattro diverse analisi con scarti quadratici medi del rumore gaussiano generato via via crescenti, in particolare usando:

- $\sigma = 0.25$ $[pixel]$
- $\sigma = 0.5$ $[pixel]$
- $\sigma = 0.75$ $[pixel]$
- $\sigma = 1$ $[pixel]$

Sono riportate in *Figura 4.26 - 4.27 - 4.28 - 4.29* le variazioni più significative ovvero quelle tra $\sigma = 0$ $[pixel]$ e $\sigma = 0.5$ $[pixel]$ e poi per $\sigma = 1$ $[pixel]$, dove si nota come soprattutto nel secondo caso, oltre ad avere un aumento della banda d'errore nella stima della profondità si ha che l'errore sugli angoli di Eulero tende a divergere

drasticamente. I grafici dei casi con $\sigma = 0.25$ $[pixel]$ e $\sigma = 0.75$ $[pixel]$ si possono visionare in Appendice.

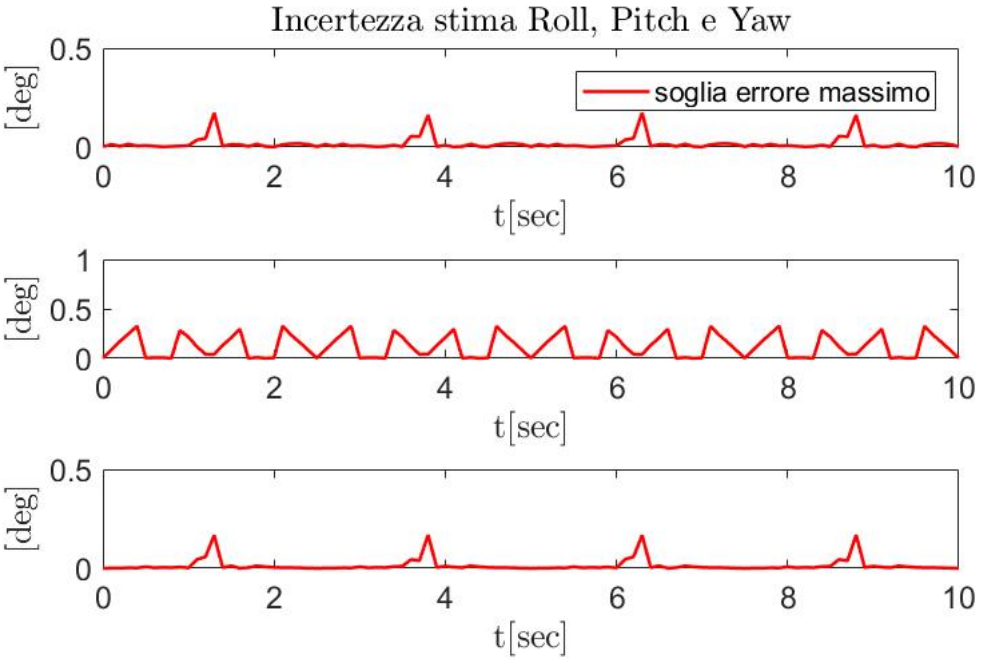


Figura 4.24: Analisi incertezza angoli di Eulero $\sigma = 0$ $[pixel]$

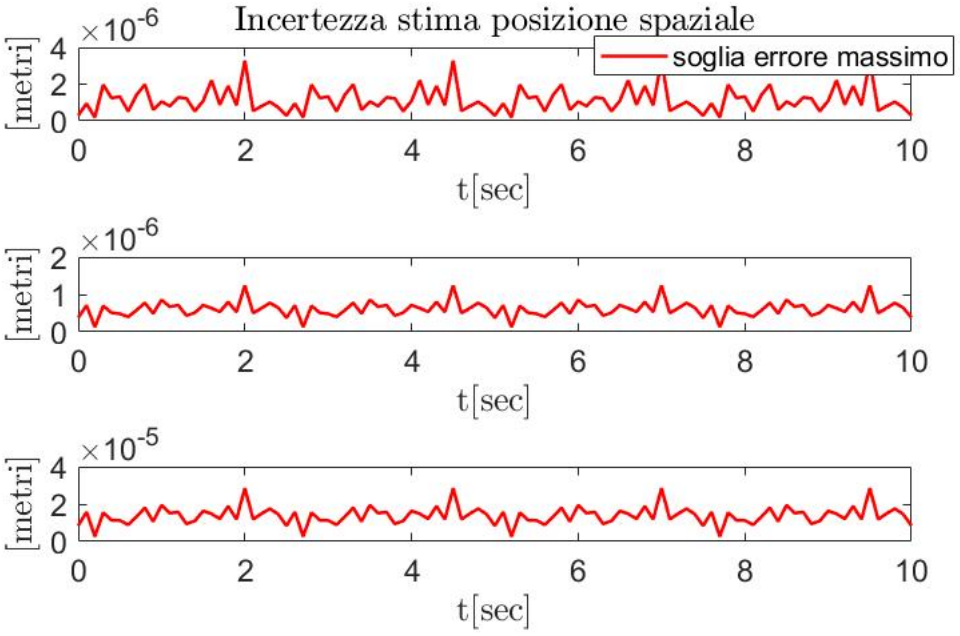


Figura 4.25: Analisi incertezza coordinate spaziali $\sigma = 0$ $[pixel]$

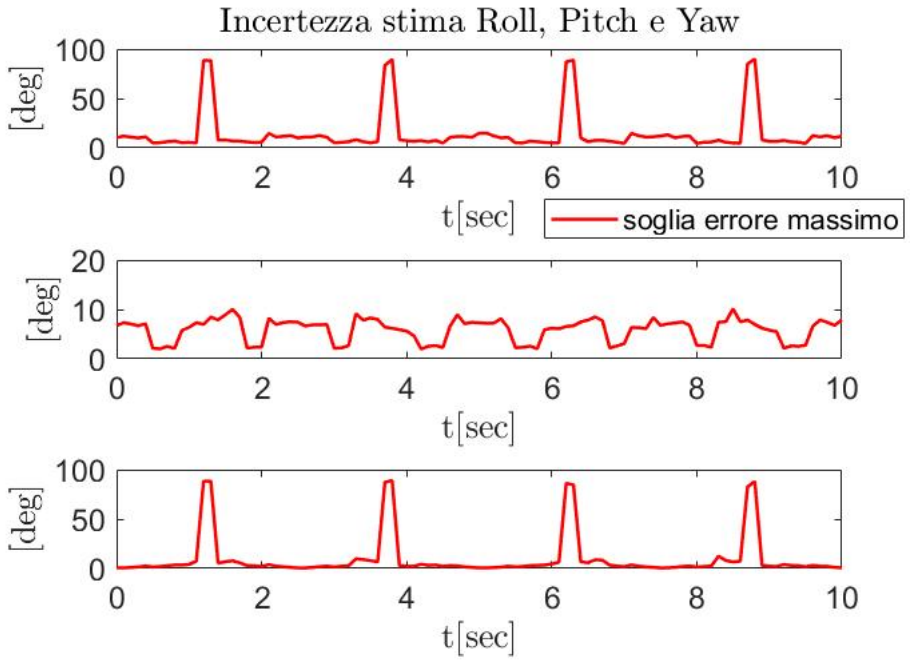


Figura 4.26: Analisi incertezza angoli di Eulero $\sigma = 0.5$ [*pixel*]

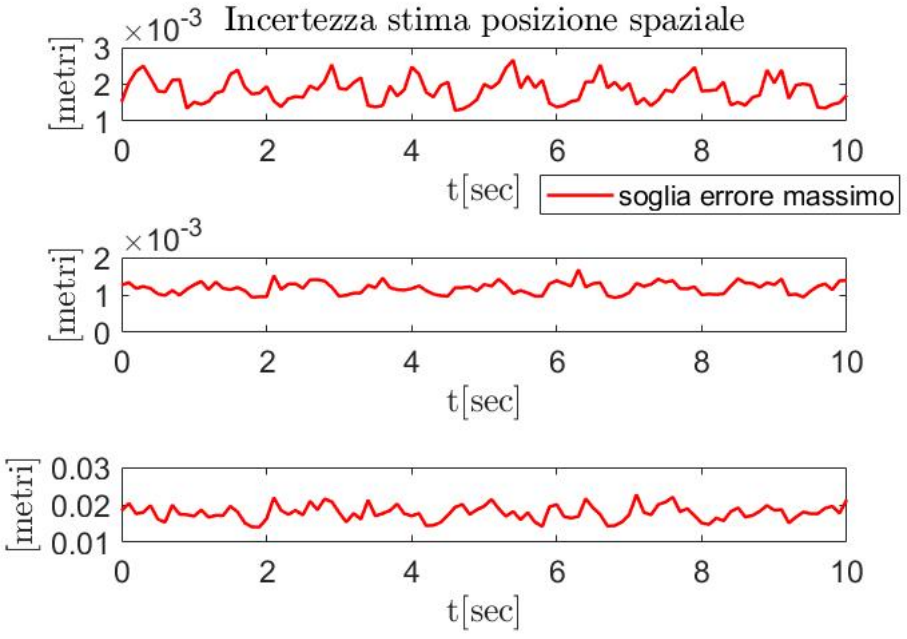


Figura 4.27: Analisi incertezza coordinate spaziali $\sigma = 0.5$ [*pixel*]

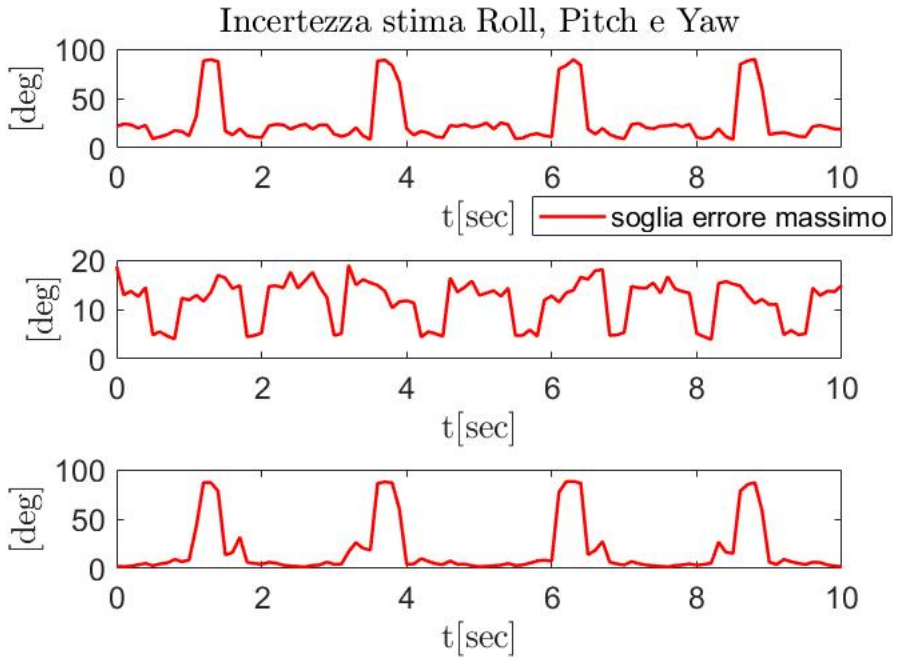


Figura 4.28: Analisi incertezza angoli di Eulero $\sigma = 1$ [*pixel*]

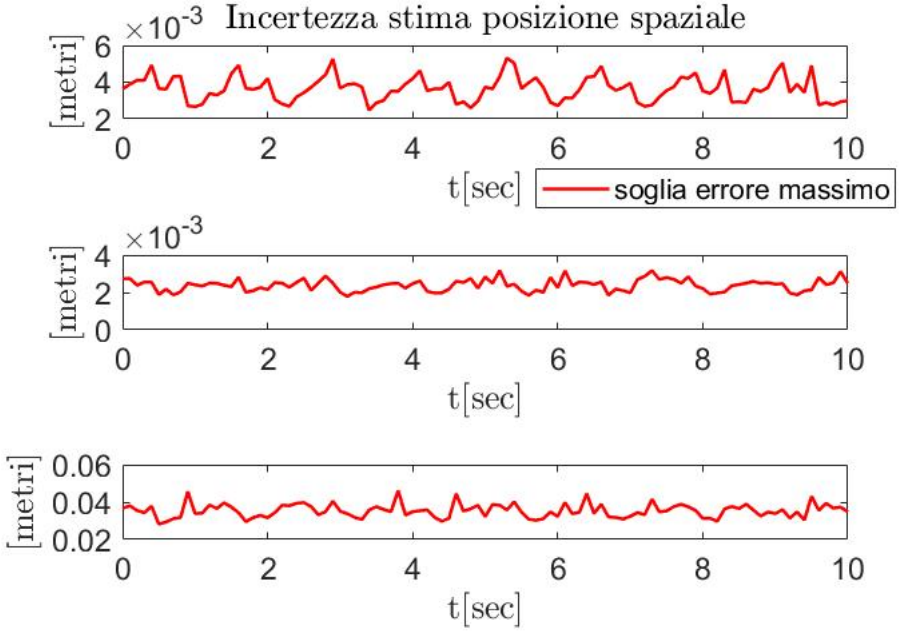


Figura 4.29: Analisi incertezza coordinate spaziali $\sigma = 1$ [*pixel*]

4.6 Velocità computazionale

Si analizza infine in quest'ultimo paragrafo la velocità computazionale riscontrata per l'intero processo d'analisi di un singolo frame, per poter ipotizzare nel futuro prossimo un'uso online del software sviluppato. Si prendono in considerazione i risultati di entrambi i test, quello virtuale e quello reale in laboratorio. Si riportano inoltre i tempi di calcolo dell'intero processo, dall'input del frame all'output degli angoli, ma anche del solo algoritmo di stima della posizione e dell'assetto. Si presentano in *Tabella 4.11* i tempi rilevati per il test virtuale mentre in *Tabella 4.12* quelli ottenuti per il test in laboratorio.

Tempi di processo	$t_{max}\ [sec]$	$t_{min}\ [sec]$	$t_{mean}\ [sec]$
Processo intero	2.04755	0.48831	1.17497
Processo di stima posizione/assetto	0.00029	0.00004	0.000068

Tabella 4.11: Tempi di calcolo test virtuale

Tempi di processo	$t_{max}\ [sec]$	$t_{min}\ [sec]$	$t_{mean}\ [sec]$
Processo intero	1.04664	0.23974	0.494704
Processo di stima posizione/assetto	0.000194	0.00003	0.00001

Tabella 4.12: Tempi di calcolo test in laboratorio

La prima cosa che si osserva è il divario fra i tempi di calcolo dell'intero processo e quelli del solo processo di stima, questo a dimostrazione del fatto che la gran parte del peso computazionale è data dall'elaborazione d'immagine. I tempi più elevati del test virtuale rispetto a quelli del test reale sono infatti imputabili in primis alla più alta risoluzione e quindi ad un più elevato numero di pixel da processare. Questo porta a fare inoltre delle considerazioni su di una possibile risoluzione ottimale, compromesso tra una più alta possibile in modo da avere la miglior definizione grafica per i markers ma che sia sufficientemente bassa da permettere dei tempi di calcolo accettabili.
Si può infine dire che mentre i tempi di calcolo rilevati per il test in ambiente virtuale non sarebbero utilizzabili, risultano interessanti quelli rilevati nel test reale in un'ottica di futura integrazione online, attualmente ipotizzabile con una frequenza di campionamento almeno intorno ad $1Hz$.

Conclusioni

Lo sviluppo del software di elaborazione dell'output di una stereocamera implementato in questo lavoro di tesi, permette la stima della posizione e dell'assetto di un micro satellite *target* rispetto ad un terna di riferimento voluta. Questo attraverso il risultato di tre step successivi quali:

- il riconoscimento nelle immagini in input di fiducial markers applicati al modulo d'assetto di un satellite *target*;
- la stima della posizione dei markers rilevati nello spazio rispetto ad un sistema di riferimento solidale alla camera;
- l'utilizzo della posizione calcolata dei markers per la stima della posa e l'assetto del *target*.

Il software è stato testato sia in ambiente virtuale che in laboratorio. I risultati ottenuti, assieme ad un'analisi d'incertezza svolta, permettono di trarre delle conclusioni sulla bontà del processo.

Il solo algoritmo di stima e ricostruzione di posizione ed assetto, si rivela efficace ed efficiente sia in termini di accuratezza che in termini di velocità computazionale, risultando però particolarmente sensibile al rumore di misura ottenuto nel rilevamento dei centroidi dei markers applicati.

Il riscontro dei markers tramite elaborazione d'immagine risulta invece efficace ma con un'accuratezza fortemente dipendente dalla qualità dell'immagine e quindi dalla risoluzione e dalle condizioni ambientali d'illuminazione. É inoltre la parte del processo che richiede il maggior tempo computazionale, attualmente questo risulta tale da poter ipotizzare un'applicazione online con frequenza di campionamento di $1Hz$.

Questo lavoro rappresenta il primo step verso un sistema di visione per il calcolo diretto di posa e assetto. Nell'ottica però di un futuro utilizzo integrato nel sistema interno di un micro satelite, sono ipotizzabili alcune migliorie e sviluppi. Oltre ad un'ottimizzazione del codice atta a migliorare ulteriormente i tempi di calcolo, si può pensare a delle soluzioni per migliorare e facilitare il rilevamento dei markers, come ad esempio:

- utilizzare per il bianco delle features dei markers delle vernici riflettenti per aumentarne il contrasto in immagine e diminuirne la dipendenza dalle condizioni ambientali di luminosità;
- nella stessa ottica, utilizzare per le pareti del modulo d'assetto una vernice nera opaca ad alto assorbimento luminoso in modo da ridurre al minimo possibili riflessi;
- aumentare la precisione dell'applicazione dei markers sulle pareti stesse, ad esempio ricorrendo ad un metodo di stampa a controllo numerico dei markers sulle superfici.

In un ambito di più alto livello invece, è pensabile per migliorare la precisione dell'intero sistema di visione, di poter accostare il metodo di calcolo diretto sviluppato in questo lavoro ad un metodo di stima predittiva come ad esempio un filtro di Kalman.

Appendice - Grafici test

Si riportano nel presente appendice i risultati grafici del test rototraslazionale omessi in *Sezione 4.3* ed i grafici delle analisi d'incertezza omessi in *Sezione 4.5*.

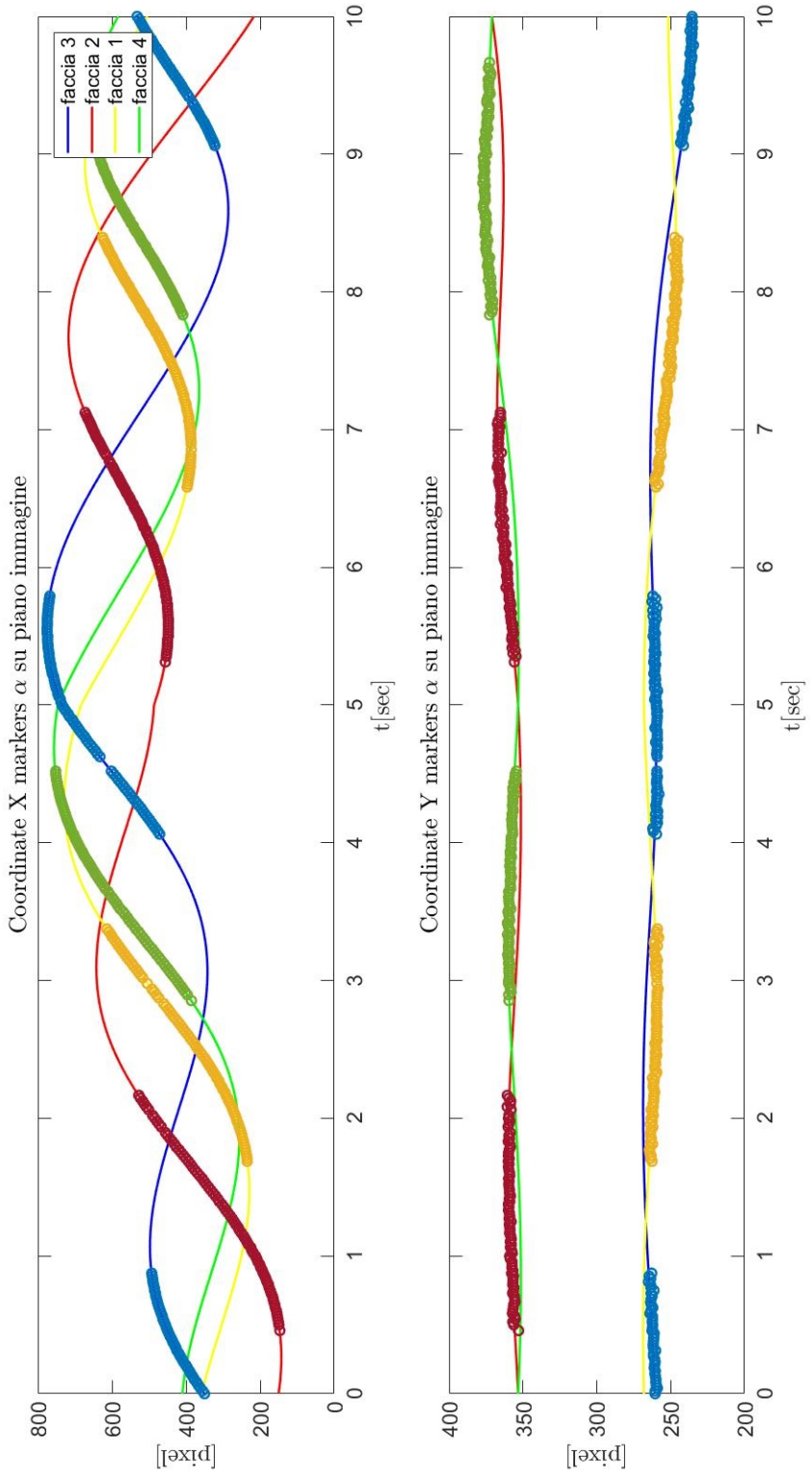


Figura 4.30: Test Rototraslazionale *Sezione 4.3.1* - Rilevamento coordinate centroidi su *piano immagine* - marker α

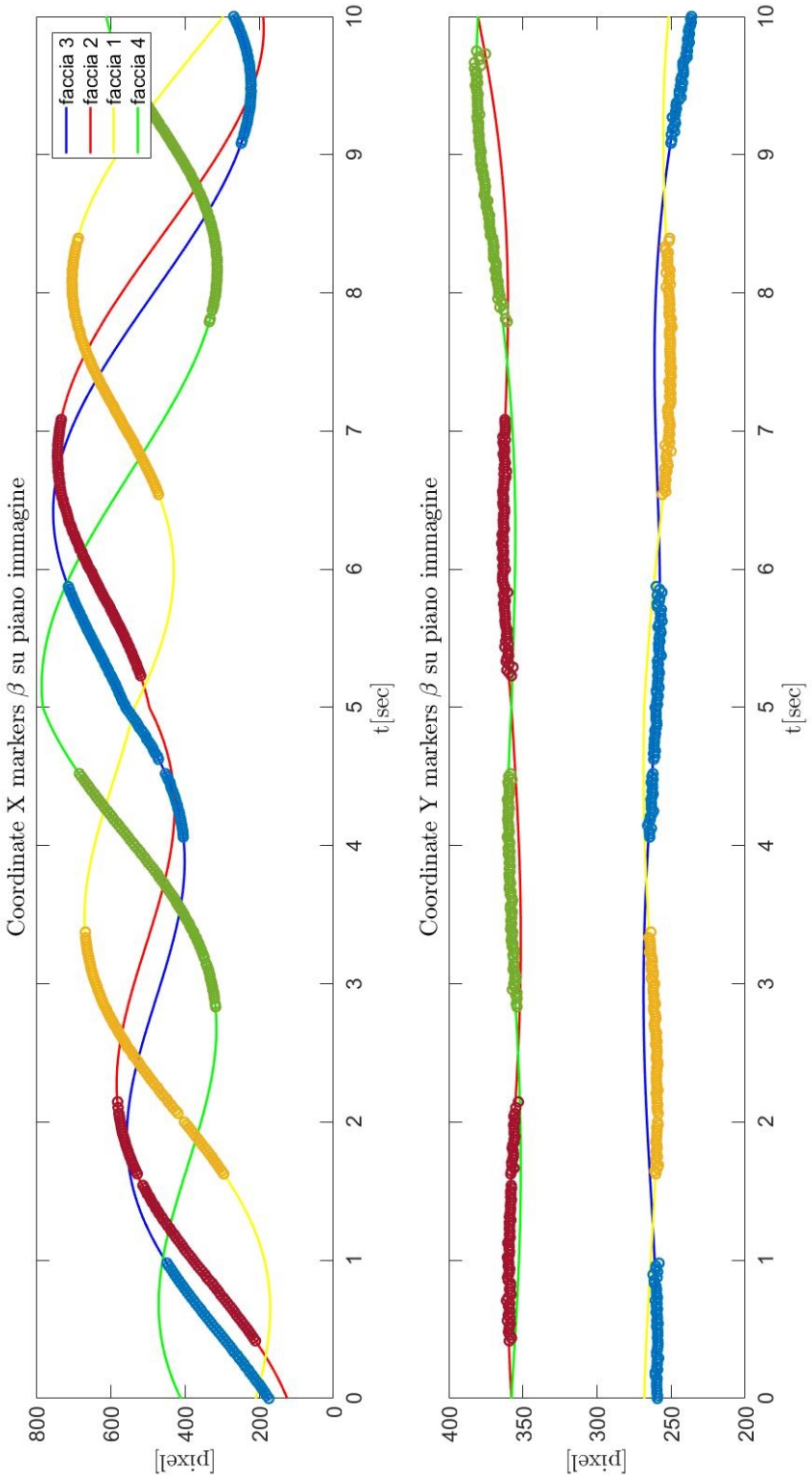


Figura 4.31: Test Rototraslazionale *Sezione 4.3.1* - Rilevamento coordinate centroidi su *piano immagine* - marker β

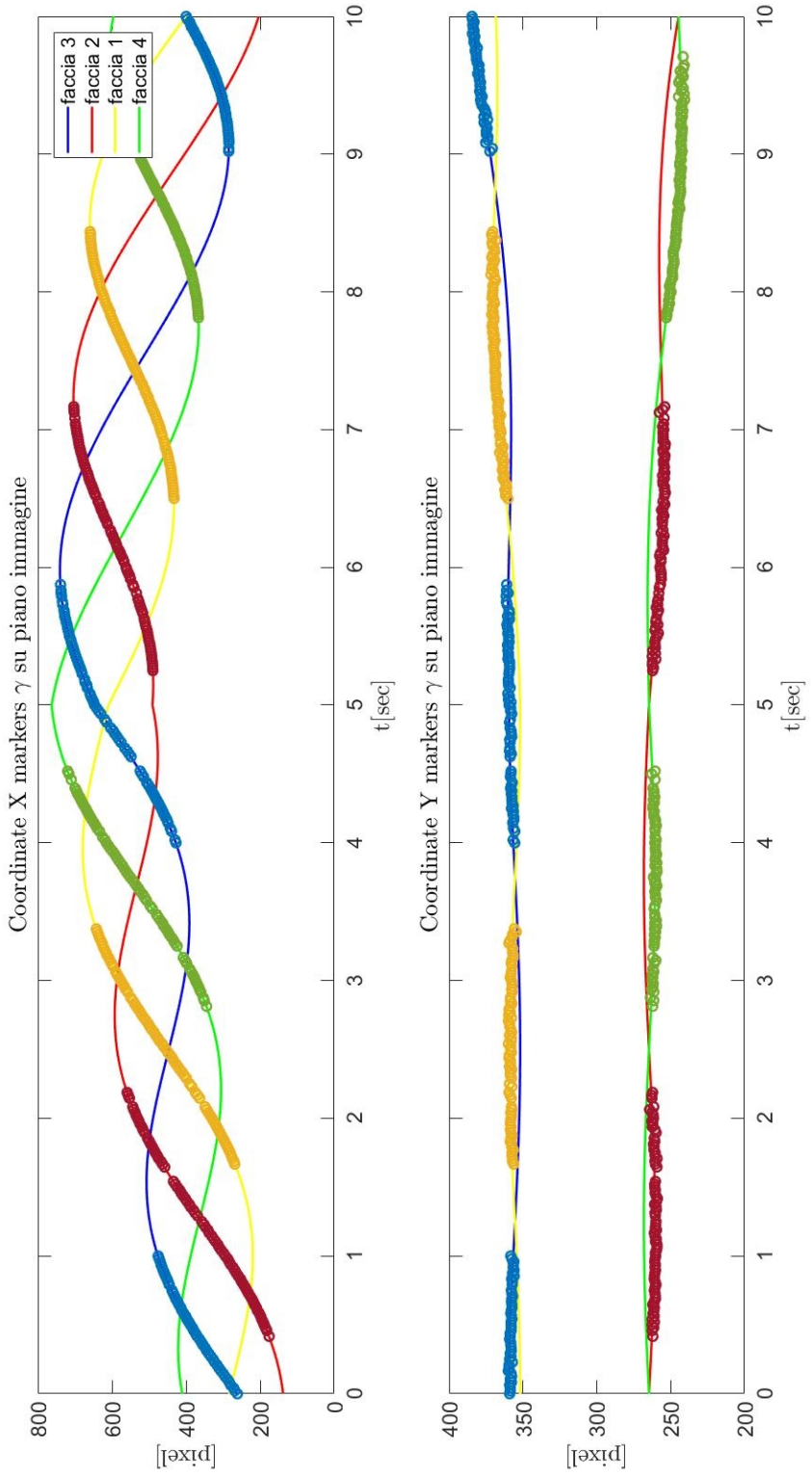


Figura 4.32: Test Rototraslazionale *Sezione 4.3.1* - Rilevamento coordinate centroidi su *piano immagine* - marker γ

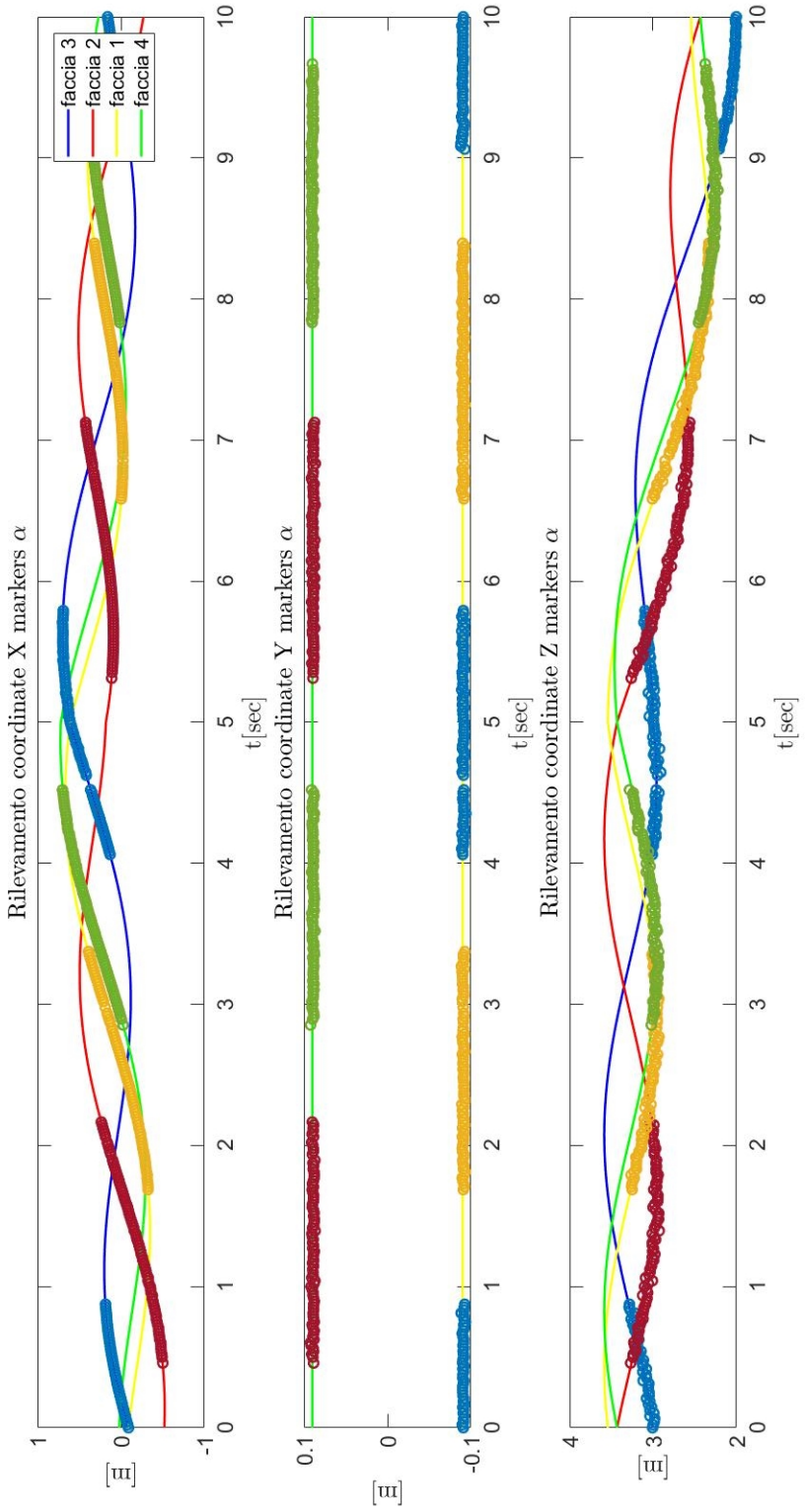


Figura 4.33: Test Rototraslazionale *Sezione 4.3.1* - Rilevamento coordinate markers in riferimento *camera* - marker α

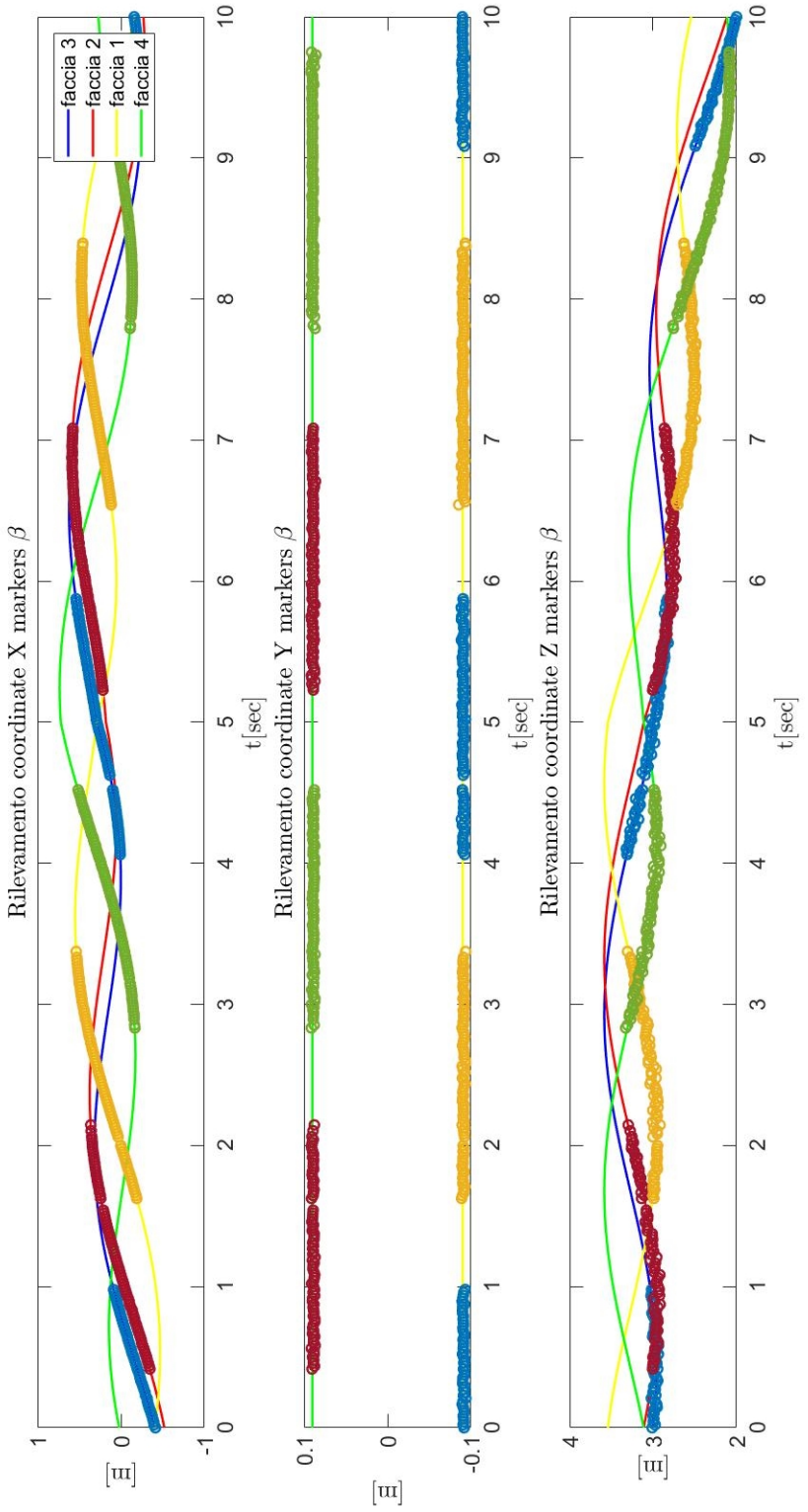


Figura 4.34: Test Rototraslazionale *Sezione 4.3.1* - Rilevamento coordinate markers in riferimento *camera* - marker β

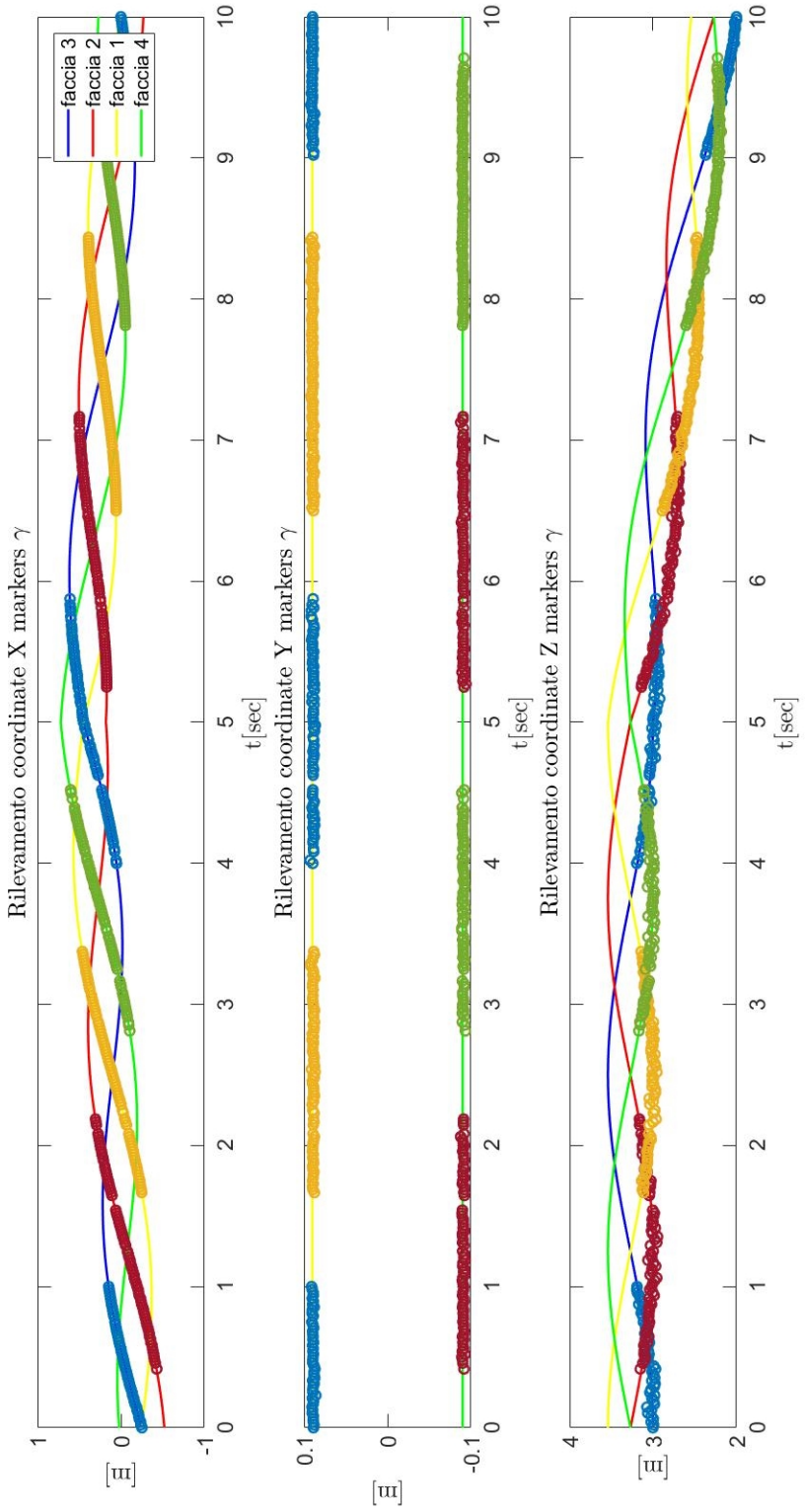


Figura 4.35: Test Rototraslazionale *Sezione 4.3.1* - Rilevamento coordinate markers in riferimento *camera* - marker γ

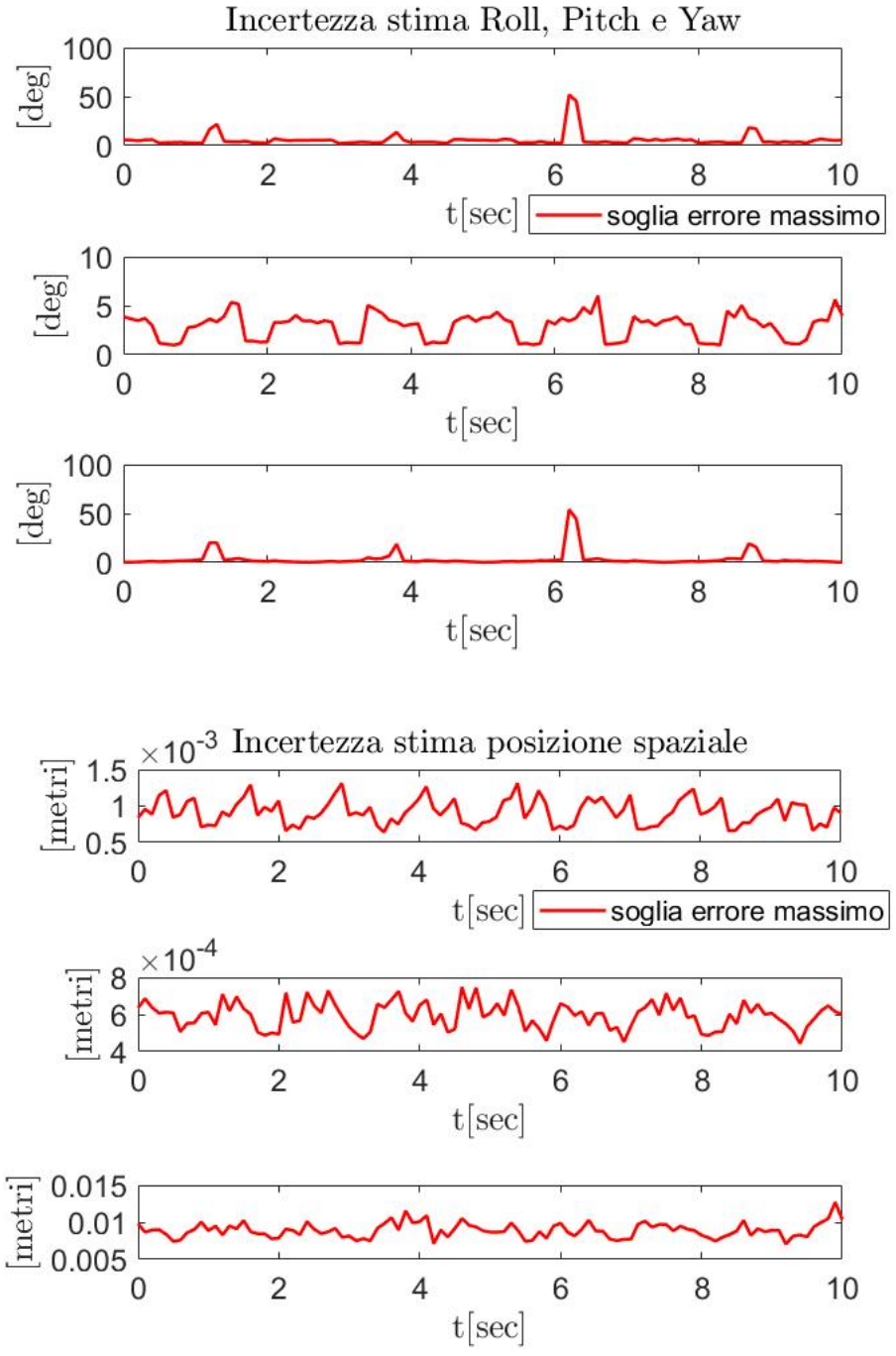


Figura 4.36: Analisi incertezza - $\sigma = 0.25$ $[pixel]$

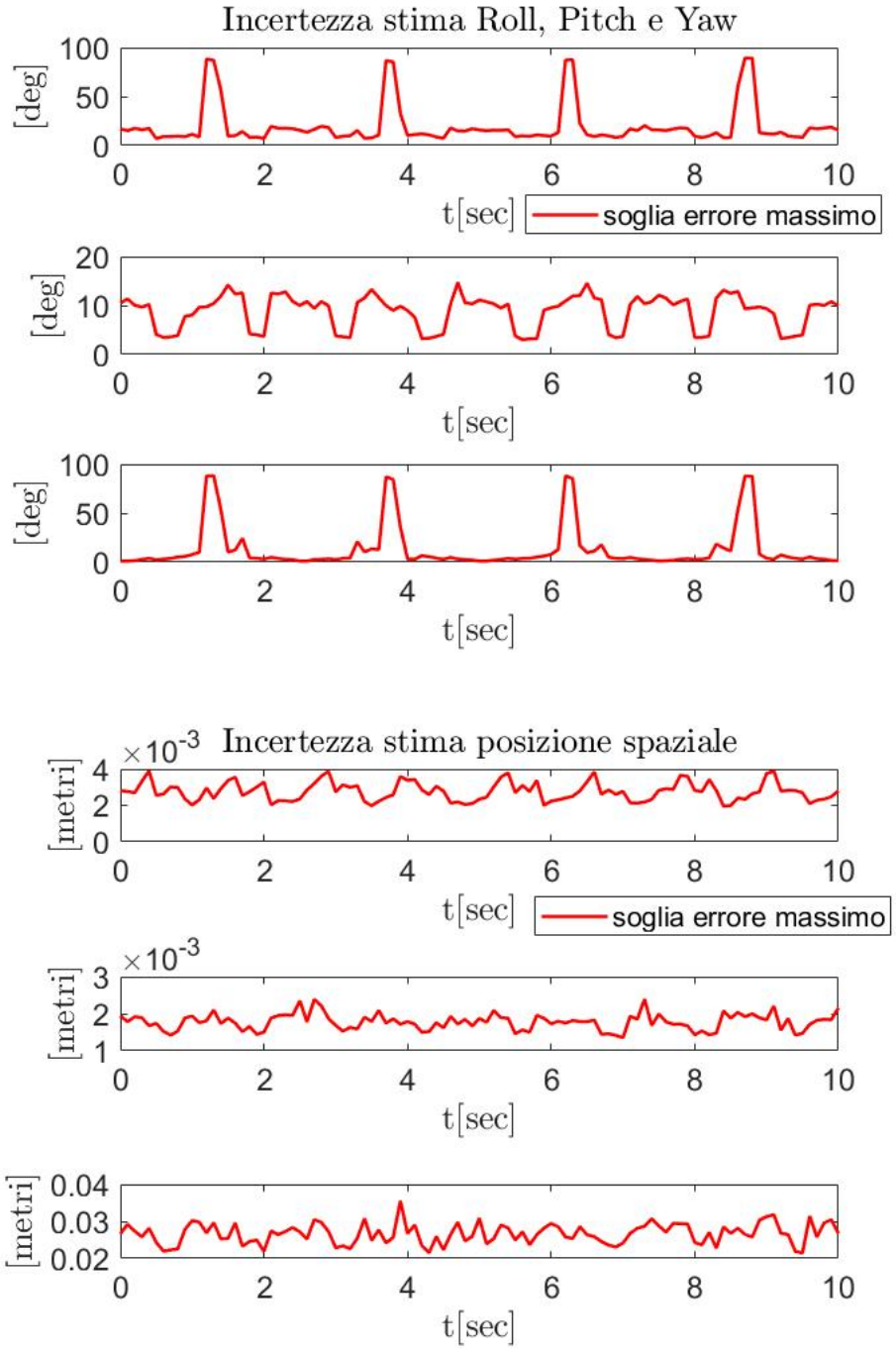


Figura 4.37: Analisi incertezza - $\sigma = 0.75$ $[pixel]$

Bibliografia

1. Sabatini, M., Palmerini, G. B., Monti, R., and Gasbarri, P. (2014). Image based control of the "PINOCCHIO" experimental free flying platform. Acta Astronautica, 94(1), 480-492.

2. Miller, David, et al. "SPHERES: a testbed for long duration satellite formation flying in micro-gravity conditions." Proceedings of the AAS/AIAA Space Flight Mechanics Meeting, Clearwater, FL, Paper No. AAS 00-110. 2000.

3. Formation Control Testbed *http* : *//dst.jpl.nasa.gov/test_beds/*

4. Sidi, Marcel J. Spacecraft dynamics and control: a practical engineering approach. Vol. 7. Cambridge university press, 1997.

5. Bottaro, Luca. "Calibrazione metrologica di un sistema di visione per la misura di posizione e assetto relativi di un satellite." (2015).

6. Oualline, Steve. Practical C++ programming. " O'Reilly Media, Inc.", 2003.

7. OpenCV *https://opencv.org/*

8. Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. S. (2012). An invitation to 3-d vision: from images to geometric models (Vol. 26). Springer Science and Business Media.

9. Szeliski, Richard. Computer vision: algorithms and applications. Springer Science and Business Media, 2010.

10. Fornaciari, Michele, and Andrea Prati. "Very fast ellipse detection for embedded vision applications." Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on. IEEE, 2012.

11. Mazzucato, Mattia. "Controllo d'assetto a tre assi di un simulatore per il volo in formazione-Software design e hardware test." (2014).

12. Valmorbida, Andrea, et al. "Design of a ground-based facility to reproduce satellite relative motions." Metrology for AeroSpace (MetroAeroSpace), 2017 IEEE International Workshop on. IEEE, 2017.

13. LeGrand, Keith. "Initial relative orbit determination using stereoscopic imaging and Gaussian mixture models." (2013).

14. Rimrott, Friedrich Paul Johannes. Introductory attitude dynamics. Springer Science and Business Media, 2012.

15. Kiryati, Nahum, Yuval Eldar, and Alfred M. Bruckstein. "A probabilistic Hough transform." Pattern recognition 24.4 (1991): 303-316.

Ringrazio il professor Enrico Lorenzini per avermi dato l'opportunità
di collaborare ad un progetto sperimentale e ringrazio infinitamente
l'Ing. Mattia Mazzucato per l'immensa pazienza, le ore spese, le conoscenze
e non da meno l'affetto che ha condiviso con me in questo lavoro.