



UNIVERSITA' DEGLI STUDI DI SIENA

Facoltà di Ingegneria

Corso di Laurea in

INGEGNERIA INFORMATICA

Invarianti geometrici per la calibrazione di sistemi catadiottrici

Tesi di laurea di

Carlo Alberto Pascucci

29 Gennaio 2007

Relatore:

Prof. Domenico Prattichizzo

Correlatori:

Prof. Antonio Pasini

Ing. Gian Luca Mariottini

Anno Accademico 2005/2006

Al

Nonno Elio

ed allo

Zio Luigi

Indice

1. Nozioni Preliminari	
1.1 Gli invarianti Geometrici.....	1
1.2 I sistemi catadiottrici.....	2
1.3 La calibrazione di un sistema catadiottrico.....	4
2. La calibrazione secondo Ying ed Hu	
2.1 Perché le sfere.....	5
2.2 Dai contorni apparenti agli invarianti S_1 ed S_2	7
2.3 Dagli invarianti alle equazioni per i parametri intrinseci ed estrinseci da stimare.....	11
2.4 L' algoritmo in due fasi.....	11
2.4.1 Prima fase.....	12
2.4.2 Seconda fase.....	13
3. L' implementazione in MATLAB	
3.1 L' ambiente virtuale di simulazione.....	15
3.2 Estrazione e fittaggio dei contorni apparenti.....	18
3.3 La prima fase : implementazione.....	22

3.4 La seconda fase : implementazione.....	25
3.4.1 Critiche al metodo.....	30
4. Conclusioni e sviluppi futuri.....	31
A. Lo specchio iperbolico.....	32
B. Le funzioni implementate in MATLAB.....	34
Bibliografia.....	44

Introduzione

Questo lavoro di tesi si propone di implementare un metodo per la calibrazione di sistemi catadiottrici basandosi sull' utilizzo di invarianti geometrici. L' algoritmo e gli invarianti utilizzati sono stati esposti da Xianghua Ying e Zhanyi Hu in [1].

Utilizzare delle sfere per la calibrazione, piuttosto che delle linee parallele come proposto da Geyer e Daniilidis in [2] e da Barreto ed Araújo in [3], è molto vantaggioso, perché ci permette di stimare i parametri intrinseci ed estrinseci della telecamera, in maniera molto più precisa. Il principio che sta alla base, è piuttosto semplice. La proiezione di una linea retta su uno specchio parabolico o iperbolico e quindi sul piano immagine, rappresenta solo un arco di una conica (tipicamente $1/3$ di una ellisse). La proiezione di una sfera, è invece una conica chiusa. Quindi farne il fitting rappresenta un processo sicuramente più robusto agli errori, in quanto in questo caso i punti dell' ellisse da fittare sono già tutti disponibili e non devono essere stimati supponendo di trovarsi davanti ad un ellisse di grandi dimensioni.

Visto che per ogni sfera si hanno due invarianti e che in totale i parametri da stimare sono sei, bastano tre sfere per ottenere un sistema di sei equazioni in sei incognite da risolvere, per trovare la matrice di calibrazione del sistema catadiottrico. Tuttavia trattandosi di equazioni non lineari, Ying e Hu, per un calcolo più efficiente dal punto di vista computazionale, propongono uno schema di risoluzione in due fasi. Sfruttando la composizione degli invarianti ricavati, nella prima fase si stimano quattro dei cinque parametri intrinseci; nella seconda invece si calcolano l' ultimo parametro interno rimasto ed il parametro esterno (relativo alle dimensioni dello specchio) del sistema catadiottrico.

Nelle pagine seguenti, dopo aver introdotto alcuni concetti fondamentali, come quello di invariante, di sistema catadiottrico e di calibrazione nel Capitolo 1, si passerà nel secondo alla descrizione degli invarianti ottenuti dalla proiezione delle sfere e delle equazioni ricavate da questi, le quali verranno utilizzate per la stima dei parametri interni ed esterni. Nel terzo capitolo invece, verrà illustrato l'algoritmo di calibrazione in due fasi. In fine nel quarto capitolo verranno mostrati i risultati sperimentali ottenuti dall' implementazione in MATLAB con EGT dei concetti precedentemente esposti.

Capitolo 1

Nozioni preliminari

Qui di seguito verranno introdotti, in maniera basilare, alcuni concetti necessari per la comprensione del testo.

1.1 Gli Invarianti Geometrici

Gli invarianti sono relazioni geometriche legate ad un oggetto, che non cambiano valore a seguito di trasformazioni come potrebbero essere la rotazione, la traslazione o la scalatura dello stesso rispetto agli assi di riferimento. In altre parole sono delle formule, il cui valore rimane costante perché legato all' oggetto in esame e non ad altri fattori esterni. Consideriamo ad esempio una conica generica

$$f(x, y) = ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0 \quad (1.1)$$

e l' espressione

$$\delta = ac - b^2 \quad (1.2)$$

Questa rappresenta un valore costante e se la conica non è degenera, ovvero

$$D = \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & f \end{bmatrix} \neq 0 \quad (1.3)$$

si ha: $\delta > 0$ se la conica in questione è un' ellisse; $\delta < 0$ se è un' iperbole oppure $\delta = 0$ se ci si trova di fronte ad una parabola. La (1.2) infatti fu definita da Eulero come invariante quadratico. Gli invarianti sono quindi degli strumenti molto potenti, poiché consentono di fare importanti assunzioni sugli elementi geometrici che ci si trova ad osservare.

Non a caso in Computer Vision sono largamente utilizzati per il riconoscimento di oggetti, ed in questo lavoro di tesi sono la base per generare delle equazioni nelle incognite che dobbiamo stimare per calibrare la telecamera.

1.2 I Sistemi Catadiottrici

Un Sistema Catadiottrico si compone di due elementi: una telecamera *pin-hole* ed un specchio, il quale, in genere, nel caso di sistemi catadiottrici centrali, si ottiene (geometricamente) ruotando intorno al proprio asse centrale una parabola od un'iperbole. Si parla quindi di specchio parabolico oppure iperbolico. La telecamera così configurata, viene definita *panoramica*, in quanto consente di visualizzare la scena con un angolo di 360° rispetto al proprio asse centrale.

In Figura 1.1 sono mostrati gli schemi di principio nel caso di specchio parabolico ed iperbolico.

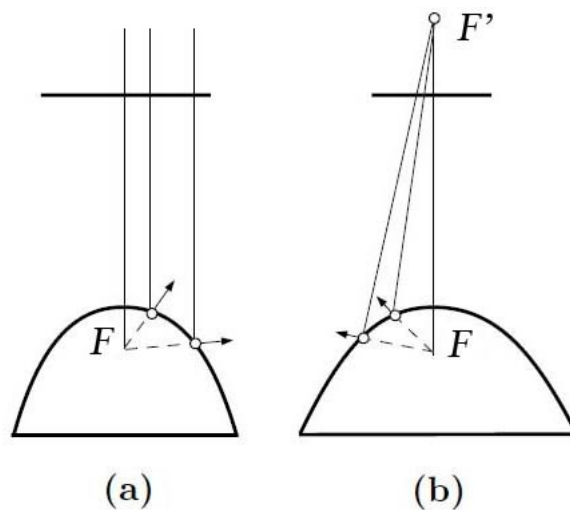


Fig. 1.1: Modelli di telecamere panoramiche: (a) specchio parabolico (b) specchio iperbolico

Nel caso in cui venga usato uno specchio parabolico come superficie riflettente, i raggi provenienti da quest' ultimo, vengono riflessi paralleli tra loro e perpendicolari rispetto al piano immagine. Se invece lo specchio è iperbolico, i raggi convergeranno nel punto F' che rappresenta il secondo fuoco dell' iperbole che genera lo specchio. Questa proprietà ci da un' indicazione precisa su dove porre la pin-hole, che infatti è posizionata proprio in F' . Così facendo si ottiene una chiara visualizzazione di qualsiasi punto riflesso dallo specchio.

Un sistema catadiottrico è caratterizzato da alcuni parametri *interni*, dipendenti dalla telecamera utilizzata e da altri parametri *esterni*, dipendenti dalle dimensioni fisiche dello specchio.

Per quanto riguarda i parametri intrinseci abbiamo: il fattore di scala (aspect ratio) r ; la distanza focale f_e ; il prodotto scalare tra i versori dell' asse x ed y del CCD della camera s ; le coordinate del punto principale (il centro del CCD) u_0 e v_0 ; in fine troviamo la distanza focale f_e , che in riferimento al modello unificante per i sistemi catadiottrici centrali esposto da Greyer e Daniilidis in [4], può essere espressa come $f_e = |O_c O_p|$, ovvero la distanza tra il centro del piano immagine O_p ed il centro della camera O_c .

I parametri estrinseci sono invece l' asse maggiore e minore della conica che rappresenta lo specchio, rispettivamente a e b (nel caso di specchio parabolico si ha solo a) ed altre grandezze derivanti da questi. Ne sono esempio l' eccentricità della sezione conica ε , che nel caso di specchio parabolico è uguale ad uno, mentre nel caso iperbolico è maggiore di uno ed è espressa dalla relazione

$$\varepsilon = \sqrt{1 + \frac{b^2}{a^2}} \quad (1.4)$$

ed il parametro

$$l = \frac{2 \cdot \varepsilon}{(1 + \varepsilon^2)} \quad (1.5)$$

il quale, sempre in riferimento al modello unificante per i sistemi catadiottrici centrali esposto da Greyer e Daniilidis in [4], può essere espresso come $l=|OO_c|$, ossia la distanza tra il centro della *viewing sphere* O ed il centro della telecamera O_c .

1.3 La Calibrazione di un Sistema Catadiottrico

Con la parola calibrazione, si intende la stima di tutti quei parametri intrinseci ed estrinseci descritti nel paragrafo precedente. Per fare questo si acquisiscono particolari immagini, rappresentanti scenari costruiti ad hoc, dalle quali estraendo alcune *features* e tramite manipolazioni algebriche è possibile risalire ai valori che caratterizzano inequivocabilmente il nostro sistema. La calibrazione, è necessaria per tutte quelle applicazioni che richiedono l'estrazione di informazioni geometriche sullo spazio tridimensionale a partire dalle immagini bidimensionali acquisite. Basti pensare a tutte le applicazioni di visual servoing, alla ricostruzione 3D degli ambienti oppure al riconoscimento di ostacoli, magari anche in movimento.

Capitolo 2

La calibrazione secondo Ying ed Hu

Mediante il sistema che andrò a descrivere tra breve, non si può calibrare qualsiasi tipo di telecamera panoramica. Quindi tratterò l' unico caso possibile: un sistema catadiottrico caratterizzato da specchio iperbolico, piuttosto che parabolico.

2.1 Perché le sfere

A causa della curvatura dello specchio sul quale si riflettono i punti della scena reale, le proiezioni di una linea o di una sfera sul piano immagine sono entrambe delle coniche. Il processo di calibrazione richiede di estrarre i parametri di queste figure geometriche a partire da ciò che si vede sul CCD. In una parola dobbiamo *fittare* i contorni apparenti. Sebbene in teoria bastino solo cinque punti per definire una conica, nella realtà più punti abbiamo a disposizione, maggiore sarà la precisione della nostra stima. In riferimento alla Fig. 2.1 si vede che inquadrando una scena con delle linee rette otteniamo delle curve, che sono solo una porzione della conica che rappresentano. Invece inquadrando delle sfere ed estraendone i contorni, come in Fig. 2.2, otteniamo delle coniche chiuse. Quindi abbiamo a disposizione tutti i possibili punti di cui possiamo aver bisogno per farne il fittaggio. Utilizzando quest' ultimo sistema otterremo una stima molto precisa delle coniche. Ciò si traduce in una base davvero solida dalla quale partire per il processo di calibrazione. E' bene ricordare infatti che la precisione delle stime, è fondamentale per ottenere valori il più possibile vicini alla realtà e quindi proficuamente utilizzabili nelle applicazioni che coinvolgeranno la telecamera panoramica.

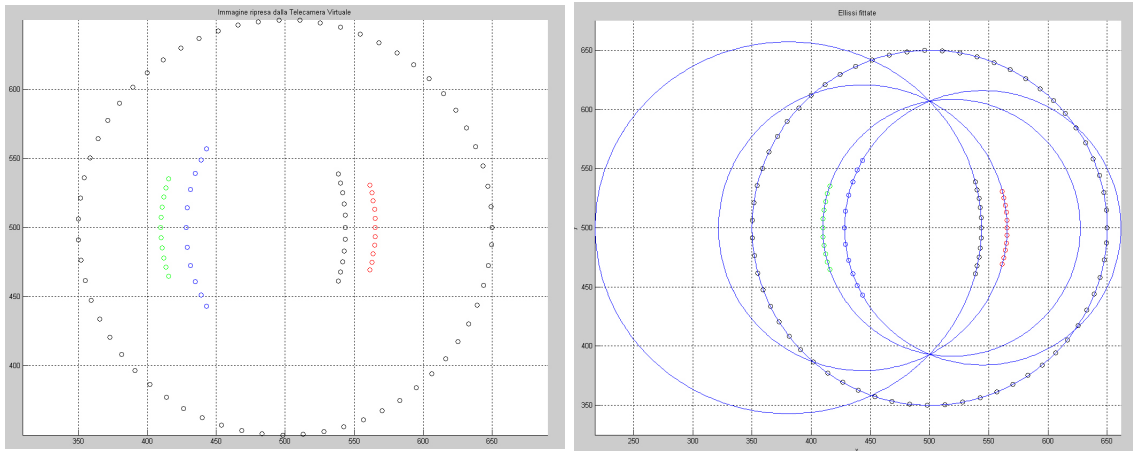


Fig. 2.1: Immagini acquisite partendo da linee rette nello spazio 3D e successivamente fittate

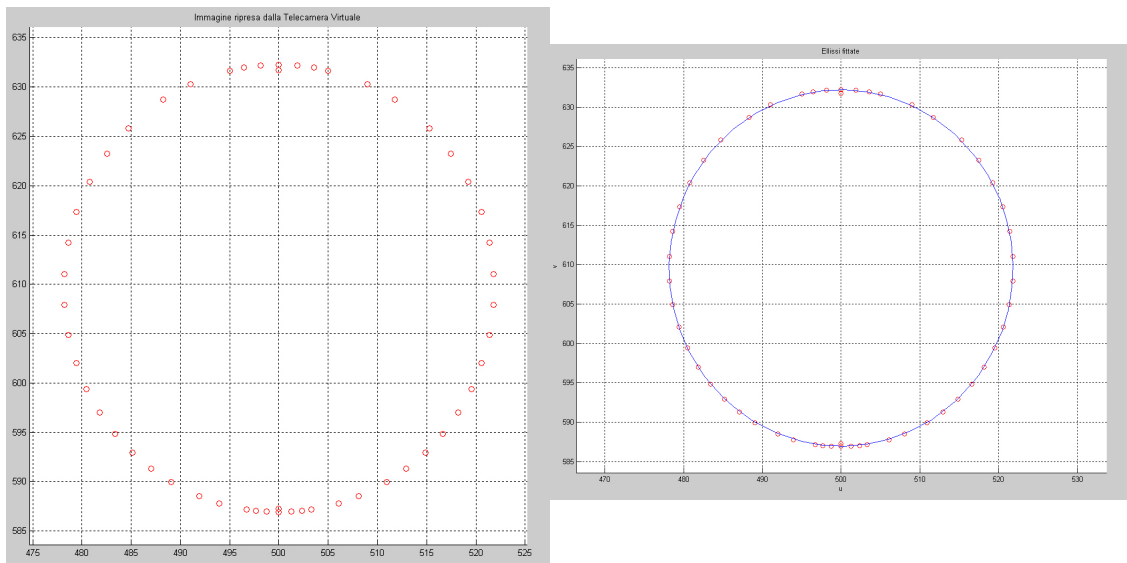


Fig. 2.2: Contorno chiuso estratto dall' immagine di una sfera e relativo fittaggio

2.2 Dai contorni apparenti agli invarianti S_1 ed S_2

Per ricavare gli invarianti, bisogna inizialmente porsi nel caso particolare della *metric catadioptric projection* in cui $r=1$, $s=0$, $u_0=0$ e $v_0=0$. La matrice di calibrazione

$$K = \begin{bmatrix} rf_e & s & u_0 \\ 0 & f_e & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

diventa quindi

$$K_M = \begin{bmatrix} f_e & 0 & 0 \\ 0 & f_e & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

Sotto questa condizione si ottiene che la forma quadratica della proiezione di una sfera sul piano immagine è

$$C_S = \begin{bmatrix} (l^2-1)n_x^2 + (d_0-l \cdot n_z)^2 & (l^2-1)n_x n_y & (ld_0-n_z)f_e n_x \\ (l^2-1)n_x n_y & (l^2-1)n_y^2 + (d_0-l \cdot n_z)^2 & (ld_0-n_z)f_e n_y \\ (ld_0-n_z)f_e n_x & (ld_0-n_z)f_e n_y & f_e^2(d_0^2-n_z^2) \end{bmatrix} \quad (2.3)$$

dove facendo riferimento a [4] $(n_x, n_y, n_z)^T$ è il vettore unitario delle normali del piano base e d_0 è la sua distanza dal centro della viewing sphere O . In questo caso l'immagine della conica che ne deriva può essere espressa come

$$m^T C_S m = 0 \quad (2.4)$$

dove

$$C_s = \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & f \end{bmatrix} \quad \text{ed} \quad m = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Per riportarci nel caso di una *generic catadioptric projection* (dove la (2.1) è la matrice di calibrazione) definiamo

$$K_A = \begin{bmatrix} r & s' & u_0 \\ 0 & 1 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

con $s' = s/f_e$, tale che

$$K = K_A K_M \quad (2.6)$$

la (2.4) diventa ora

$$m'^T C'_s m' = 0 \quad (2.7)$$

con

$$C'_s = \begin{bmatrix} a' & b' & d' \\ b' & c' & e' \\ d' & e' & f' \end{bmatrix} \quad \text{ed} \quad m = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Gli elementi di C'_s derivano dal fittaggio della conica, mentre u e v sono le coordinate in pixel dei punti dell'immagine.

Per come abbiamo definito m ed m' sappiamo che $m' = K_A m$ quindi

$$C_s = K_A^T C' K_A \quad (2.8)$$

Dopo aver definito la proiezione di una conica, data dall' estrazione del contorno apparente di una sfera, cerchiamo di scoprire a quali vincoli questa conica deve sottostare e quali equazioni possiamo ricavarne per ottenere in seguito anche una stima di l ed f_e .

A questo scopo, consideriamo la matrice di proiezione della telecamera nel caso della metric catadioptric projection

$$P = \begin{bmatrix} f_e & 0 & 0 & 0 \\ 0 & f_e & 0 & 0 \\ 0 & 0 & 1 & l \end{bmatrix}$$

il cono determinato da C_s e dal centro di proiezione O_c definito in [4]

$$Q = P^T C_s P = \begin{bmatrix} af_e^2 & bf_e^2 & df_e & df_e \\ bf_e^2 & cf_e^2 & ef_e & elf_e \\ df_e & ef_e & f & fl \\ dlf_e & elf_e & fl & fl^2 \end{bmatrix} \quad (2.9)$$

la forma quadratica della viewing sphere

$$V_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad (2.10)$$

ed una combinazione lineare di questi elementi

$$H \equiv Q + \lambda V_s = \begin{bmatrix} af_e^2 + \lambda & bf_e^2 & df_e & dlf_e \\ bf_e^2 & cf_e^2 + \lambda & ef_e & elf_e \\ df_e & ef_e & f + \lambda & fl \\ dlf_e & elf_e & fl & fl^2 - \lambda \end{bmatrix} \quad (2.11)$$

quello che abbiamo è un cono (2.9) che interseca la viewing sphere (2.10) in un cerchio. Quindi dovrebbe esistere un piano (base plane) che lo contiene. Siccome una coppia di piani distinti o coincidenti può essere considerata una quadrica degenera di rango rispettivamente 2 od 1, ciò che dobbiamo fare, è trovare un λ tale che il rango di H sia al più 2. In questa maniera avremo trovato i piani che contengono l'intersezione. Sostituendo il λ così trovato nella (2.11) otteniamo due equazioni in a, b, c, d, e, f, f_e, l , che rappresentano i due invarianti

$$S_1 = d(bd - ae) - e(be - cd) = 0$$

$$S_2 = b(bd - ae)f_e^2 - e(bf - de)(l^2 - 1) = 0$$

che utilizzeremo per calibrare la telecamera.

Bisogna innanzi tutto notare che questi invarianti non contengono n_x, n_y, n_z, d_0 , perché S_1 ed S_2 non cambiano a seconda della posizione delle sfere nello spazio. Inoltre per quanto riguarda il loro significato geometrico, possiamo dire che S_1 è un invariante di rotazione e di scala e ci dice che uno degli assi maggiori della conica C_s passa per l'origine del piano immagine. Invece S_2 è un invariante di rotazione.

2.3 Dagli invarianti alle equazioni per i parametri intrinseci ed estrinseci da stimare

Espandendo la (2.8) otteniamo

$$\begin{aligned}a &= r^2 a' \\b &= rs' a' + rb' \\c &= s'^2 a' + 2s' b' + c' \\d &= ru_0 a' + rv_0 b' + rd' \\e &= s' u_0 a' + u_0 b' + s' v_0 b' + v_0 c' + s' d' + e' \\f &= u_0^2 a' + 2u_0 v_0 b' + v_0^2 c' + 2u_0 d' + 2v_0 e' + f'\end{aligned}\tag{2.12}$$

Visto che gli elementi di C'_s possiamo ottenerli attraverso il fittaggio della conica sul CCD, sostituendo la (2.12) negli invarianti S_1 ed S_2 otteniamo delle equazioni vincolanti per la calibrazione della telecamera.

Per ogni sfera ho due invarianti ed i parametri da stimare sono in tutto sei: cinque interni ed uno esterno. E' facile capire allora come con solo tre sfere ottenendo un sistema di sei equazioni in sei incognite abbiamo materiale a sufficienza per la stima di tutti i parametri.

2.4 L' algoritmo in due fasi

Sebbene in teoria si potrebbe calibrare la telecamera in un "colpo solo" utilizzando tre sfere ed il sistema di sei equazioni in sei incognite che ne deriva, trattandosi di espressioni non lineari, Ying ed Hu, propongono un metodo in *due fasi*, computazionalmente piu leggero.

2.4.1 Prima fase

In questo stadio si trovano r , s' , u_0 e v_0 . Per farlo, sostituiamo la (2.12) nell'invariante S_1 ottenendo così un'equazione nelle sole incognite r , s' , u_0 e v_0 , giacché i parametri a' , b' , c' , d' , e' ed f' di C'_S li conosciamo a priori grazie al processo di fitting precedentemente effettuato. Abbiamo quindi bisogno di 4 sfere in questa fase, per generare un sistema di quattro equazioni in quattro incognite. Si utilizza quindi un metodo non lineare ai minimi quadrati, come il Levenberg-Marquardt, per la stima di questi valori.

Questo processo numerico, richiede un buon punto di partenza, coerente con quanto stiamo per calcolare. Dovremmo perciò poter fare delle ipotesi che ci permettano di trovarlo agevolmente. Ci viene incontro in questo senso la *bounding ellipse*, ossia la circonferenza formata dal bordo dello specchio che viene inquadrato dalla telecamera. La proiezione del bordo sul piano immagine rappresenta perfettamente un cerchio contenuto in un piano perpendicolare all'asse ottico della camera. Inoltre il centro di questa figura è in asse con il centro della telecamera. Quindi ci si può ricondurre nel caso particolare in cui il versore $n_z=1$ che ci dà

$$\begin{aligned} a=c & \quad b=0 \\ d=0 & \quad e=0 \end{aligned} \quad (2.13)$$

Sostituendo poi la (2.13) nella (2.12) e risolvendo per r , s' , u_0 e v_0

otteniamo

$$\begin{aligned}
 r &= \sqrt{\left(-\frac{b'^2}{a'^2} + \frac{c'}{a'}\right)} \\
 s' &= -\frac{b'}{a'} \\
 u_0 &= \frac{b'e' - c'd'}{a'c' - b'^2} \\
 v_0 &= \frac{b'd' - a'e'}{a'c' - b'^2}
 \end{aligned} \tag{2.14}$$

Valori da utilizzare come punto di partenza per la stima dei parametri intrinseci in questa fase.

2.4.2 Seconda fase

Arrivati a questo punto ci rimangono da stimare solo due dei sei parametri previsti:

f_e ed l . Ciò che abbiamo a disposizione è l'invariante S_2 ed r , s' , u_0 e v_0 che abbiamo appena stimato. Quindi sostituendo questi parametri interni nella (2.12) e successivamente in S_2 otteniamo un'equazione nelle sole incognite f_e ed l . Quindi con le coniche estratte dai contorni apparenti di due sfere dovremmo poter stimare anche questi ultimi due valori, in quanto otterremmo, ancora una volta, un sistema di due equazioni in due incognite. In questo caso però non sono necessari un punto di partenza od un sistema di risoluzione per equazioni non lineari ai minimi quadrati tipo Levenberg-Marquardt come nella prima fase. Perché se analizziamo più attentamente $S_2 = b(bd - ae)f_e^2 - e(bf - de)(l^2 - 1) = 0$ ci accorgiamo di essere di fronte ad una quadrica, nella fattispecie un cilindro.

Questo significa che per trovare f_e ed l basterebbe studiare l' intersezione di due quadriche ottenute dalle immagini di due sfere distinte.

Purtroppo non è così semplice!

Nella descrizione di questa seconda fase ho utilizzato non a caso il condizionale. Infatti come si vedrà nel capitolo successivo, durante l'implementazione di questi ultimi passaggi, sono emersi gravi ostacoli, che rimettono in discussione la reale applicabilità del metodo descritto da Ying ed Hu.

Ci troviamo infatti nella seguente situazione. Dall' intersezione di queste quadriche si ottiene la soluzione banale

$$l=1, \quad f_e=0 \quad \forall a, b, c, d, e, f \quad (2.15)$$

che non può essere accettata, visto che nel caso di specchio iperbolico $0 < l < 1$ e la distanza focale non può essere nulla (altrimenti non vedremmo niente sul CCD).

Le altre soluzioni (infinite!) sono date dai punti della curva, una quartica, generata dall' intersezione delle due quadriche. Sorge quindi un problema non banale:

Quale punto scegliere tra questi supponendo di non conoscere a priori ne l e tanto meno f_e ?

Capitolo 3

L' implementazione in MATLAB

Utilizzando MATLAB, con l' Epipolar Geometry Toolbox [5] ed il Robot toolbox [6] è stato possibile fare delle simulazioni per verificare l' efficacia del metodo di calibrazione attraverso gli invarianti geometrici proposti da Ying e da Hu.

3.1 L' ambiente virtuale di simulazione

Per iniziare è necessario creare uno spazio tridimensionale con un sistema di assi di riferimento, all' interno del quale inserire il sistema catadiottrico virtuale (con il proprio sistema di assi) e le sfere.

Per fare questo sono state utilizzate le funzioni di EGT

```
f_3Dwf (crea lo spazio con gli assi di riferimento)
f_3Dpanoramic (genera la telecamera panoramica)
f_3Dframe (posiziona il sistema di riferimento della telecamera)
f_3Dsurface (crea superfici tridimensionali)
```

Nelle seguenti figure è possibile vedere il risultato di queste operazioni.

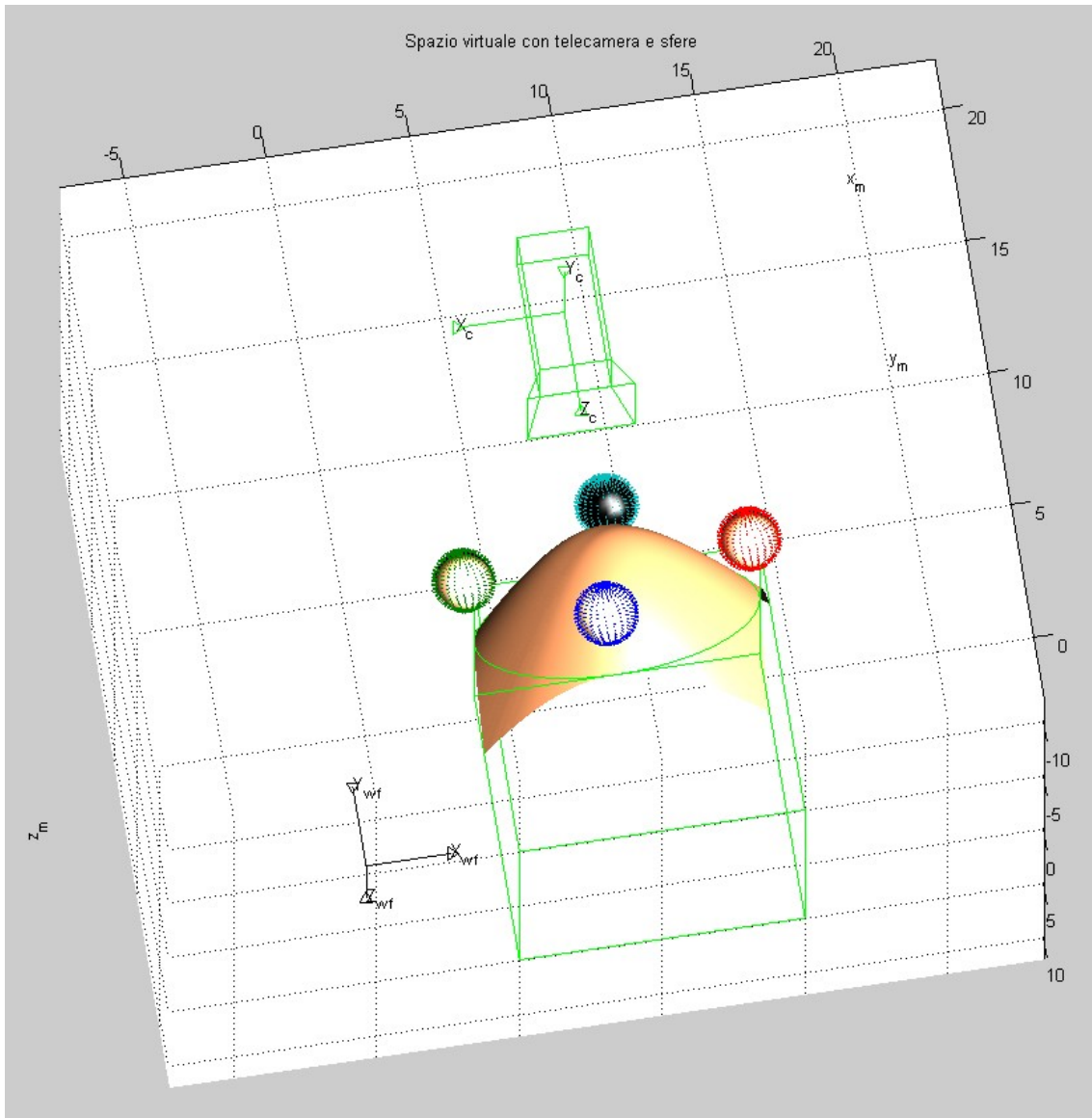


Fig. 3.1: vista della telecamera e delle sfere nello spazio tridimensionale

Nell' immagine è possibile distinguere il world frame (in nero), la telecamera ed il suo sistema di riferimento (in verde) e subito sotto, lo specchio iperbolico, con intorno le sfere. In fine la curva verde alla base dello specchio, altro non è che a bounding ellipse.

In fine, la matrice di calibrazione K , descritta nella (2.1), del sistema catadiottrico simulato è

$$K = \begin{bmatrix} 400 & 0 & 500; \\ 0 & 400 & 500; \\ 0 & 0 & 1 \end{bmatrix}; \quad (3.1)$$

ed il parametro estrinseco l , relativo alle dimensioni dello specchio è

$$l = 0.9660$$

3.2 Estrazione e fittaggio dei contorni apparenti

nella figura qui sotto, è possibile vedere il particolare di una sfera. Le piccole frecce colorate sono le normali alla sua superficie, necessarie per il calcolo del generatore di contorno e quindi per l'estrazione del contorno apparente.

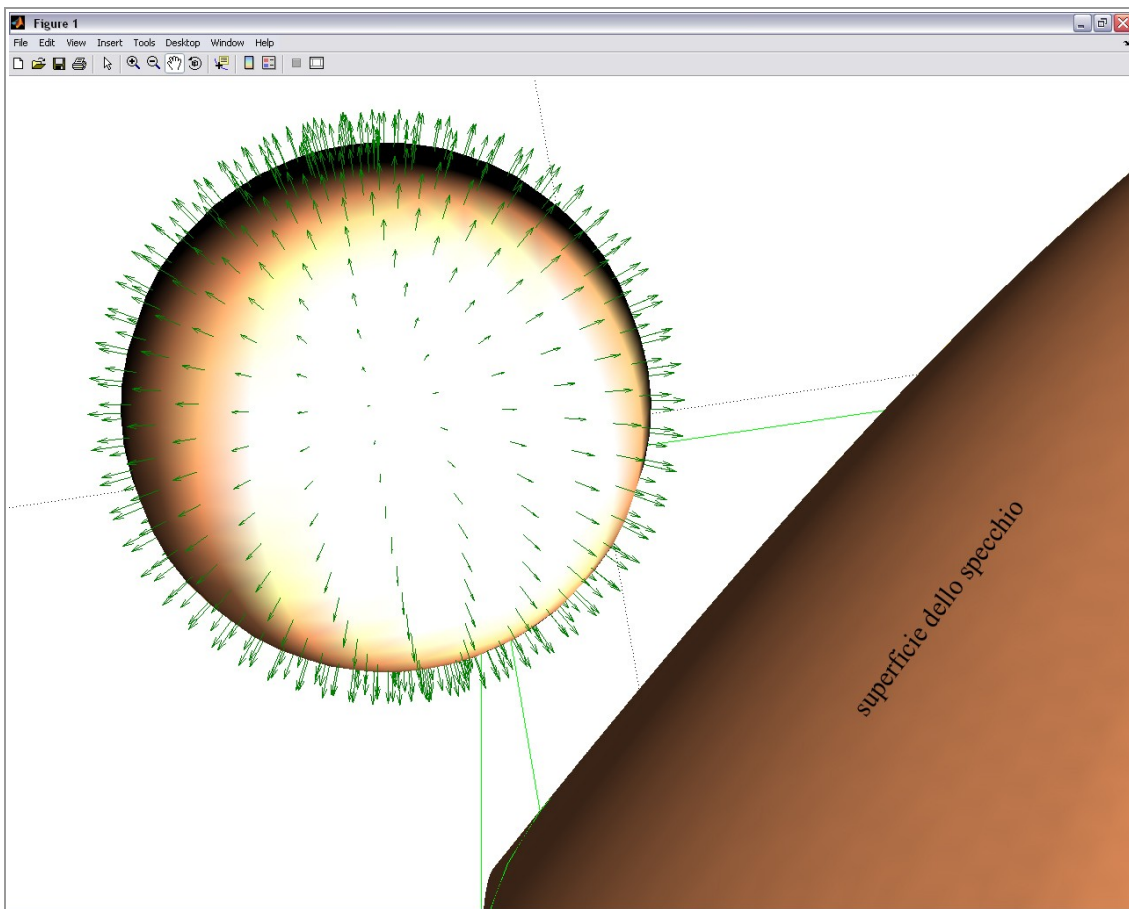


Fig. 3.2: Sfera con le normali alla superficie

Applicando alla situazione corrente quanto descritto da Roberto Cipolla in [7] infatti, il generatore di contorno è l'insieme dei punti (che si trovano sulla superficie della sfera), tali che, il prodotto scalare tra il vettore che rappresenta il raggio luminoso incidente sullo specchio e la normale alla superficie della sfera sia nullo. Questo metodo però richiederebbe il calcolo di infinite normali, quindi per l'esperimento si è dovuta scegliere una soglia (comunque vicina allo zero). Il generatore di contorno così calcolato, sarà perciò solo un'approssimazione di quello reale. In Fig. 3.3 è possibile vederne il risultato.

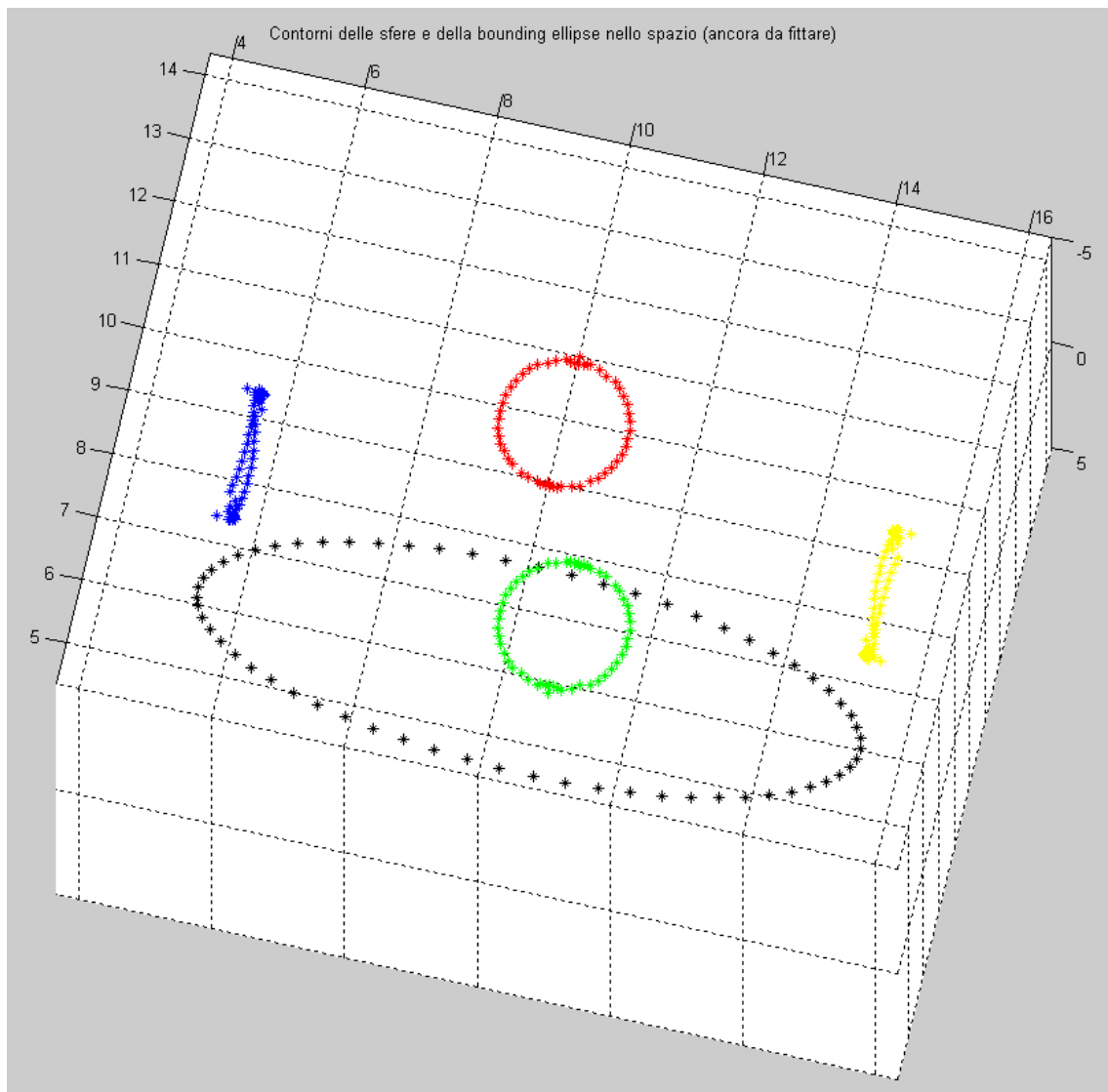


Fig. 3.3 Generatori di contorno

Per il calcolo di questi punti è stata necessaria la scrittura di una funzione apposita:

`f_gencon`

che mette in pratica quanto detto in [7] con in più la gestione del threshold per il prodotto con le normali diverso da zero. Per farlo mi sono appoggiato anche alla funzione `f_apparcontour` scritta precedentemente da Gian Luca Mariottini.

A questo punto come su può vedere in Fig. 3.4, utilizzando la funzione `f_panproj` si ottengono le coordinate dei pixel sul CCD dei punti che costituiscono i contorni apparenti delle sfere e della bounding ellipse.

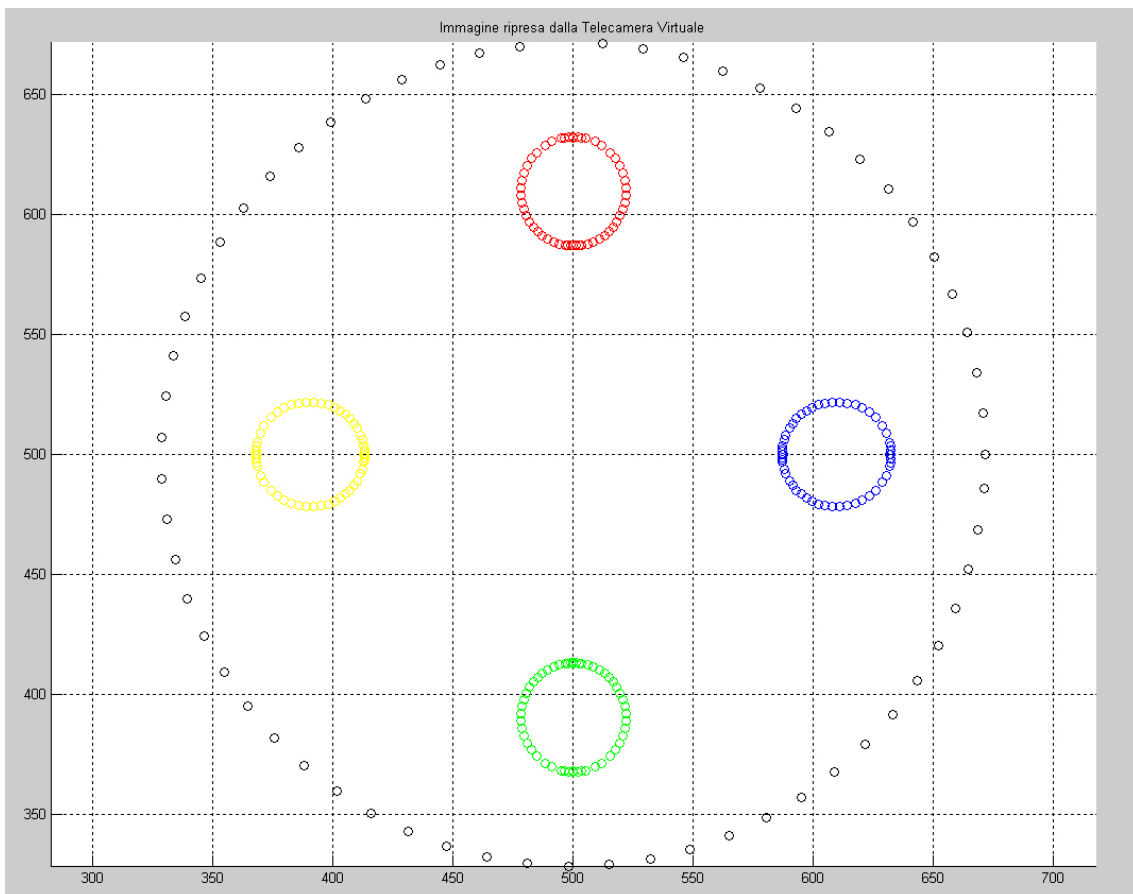


Fig. 3.4 Contorni apparenti

Quello che manca ora prima di arrivare all'implementazione dell' algoritmo vero e proprio, è il fittaggio delle ellissi rappresentate dai punti visibili nella precedente figura. Per quest' ultima operazione ho utilizzato il metodo di stima ai minimi quadrati implementato da Duane Hanselman [8] nella funzione `ellipsefit`. Il risultato è quanto segue.

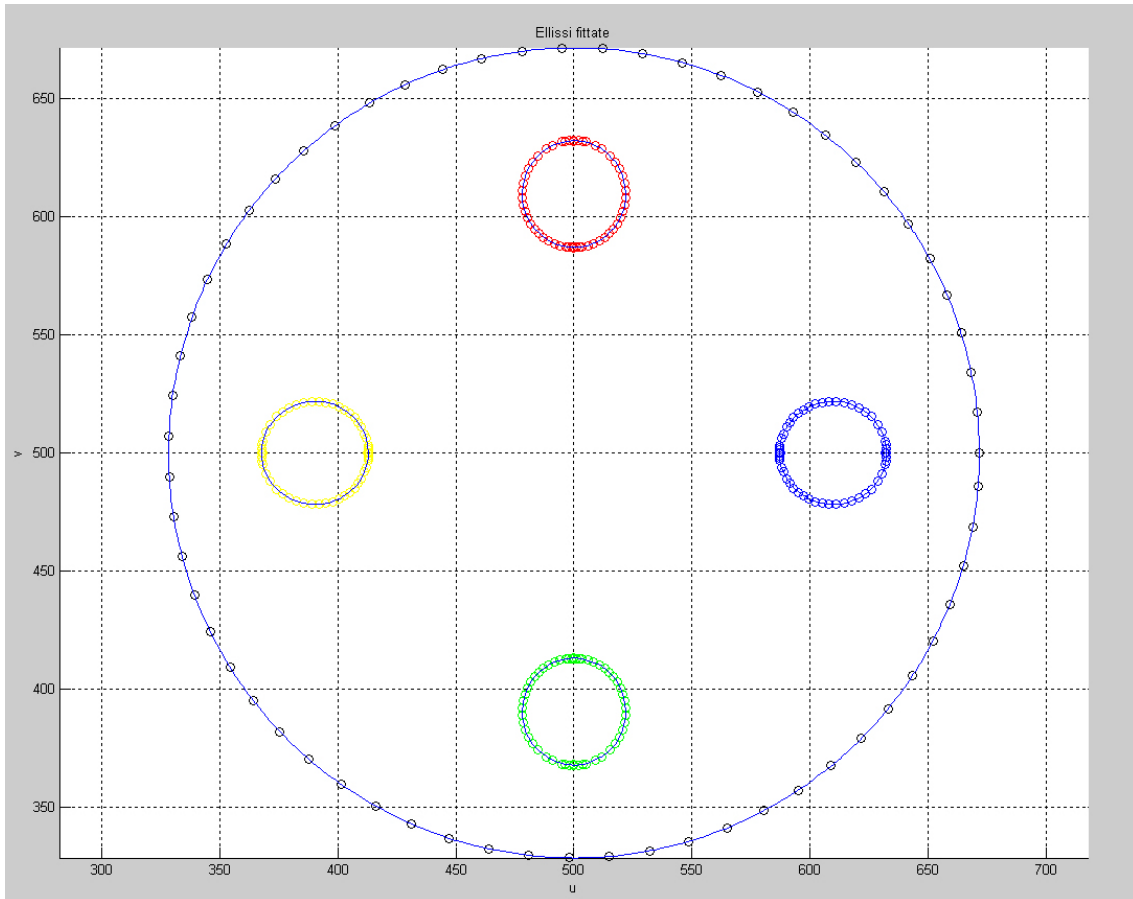


Fig. 3.5 Coniche fittate

3.3 La prima fase : implementazione

Tutto questo per cercare di rimanere il più possibile coerenti con le prove sperimentali fatte da Ying ed Hu.

Abbiamo ora tutto quello che serve per implementare la prima fase dell' algoritmo di Ying ed Hu. L'output della funzione `ellipsefit` infatti, ci da i valori degli elementi delle matrici C'_s . Quindi fittando la bounding ellipse e sostituendo ciò che si ottiene nelle (2.14) possiamo definire il vettore

```
pi0 = [r_Init s1_Init u0_Init v0_Init];
```

con i valori

```
pi0 = [1 2.0276e-16 500 500];
```

che costituisce il punto di partenza per la stima con Levenberg-Marquardt. Da qui si entra nella funzione

```
f_intrinseci
```

all' interno della quale, dopo aver passato i valori relativi al fittaggio degli altri contorni apparenti, si calcolano le soluzioni del sistema di quattro equazioni in quattro incognite

$$\begin{aligned} S1 = & [d*(b*d-a*e) - e*(b*e-c*d); \\ & d2*(b2*d2-a2*e2) - e2*(b2*e2-c2*d2); \\ & d3*(b3*d3-a3*e3) - e3*(b3*e3-c3*d3); \\ & d4*(b4*d4-a4*e4) - e4*(b4*e4-c4*d4)]; \end{aligned} \quad (3.2)$$

dove i coefficienti numerici sono ottenuti sostituendo passo per passo le soluzioni della (2.12) per ognuna delle quattro sfere.

Utilizzando il risolutore `lsqnonlin` di MATLAB si ottiene

```
piLsqnonlin =  
1.0000 -0.0000 500.0000 500.0000
```

Dove i valori mostrati sono rispettivamente

r , s' , u_0 e v_0

con

```
resnorm =  
4.6728e-011
```

```
residual =  
1.0e-005 *  
-0.3291  
-0.1382  
-0.4962  
-0.3060
```

In alternativa si può utilizzare il risolutore `fsolve`. In questo caso si ottengono risultati leggermente diversi, ma comunque molto buoni

```
piFsolve =  
  
0.9665 -0.0000 500.0010 500.0000
```

```
fval =  
  
1.0e-005 *  
  
-0.0926  
-0.2903  
-0.2907  
-0.4890
```

```
exitflag =  
  
1
```

Da notare l' output `exitflag = 1` che sta ad indicare la convergenza della stima. Come abbiamo visto, non ci sono grosse difficoltà nel ricavare r , s' , u_0 e v_0 , anche se è interessante sottolineare come le stime diventino più precise spostando le sfere verso la telecamera, piuttosto che verso la base dello specchio. Il motivo di questo è da ricercare in una maggiore efficienza del fitting in una situazione piuttosto che nell' altra. Come detto più volte infatti, la precisione nel fitting è fondamentale per ottenere stime corrette.

3.4 La seconda fase : implementazione

Come già accennato in precedenza, in questo passo si cerca di stimare l'unico parametro estrinseco l e l'ultimo parametro intrinseco, non stimato nello stadio precedente, f_e . Seguendo quanto suggerito in [1] inserendo i risultati ottenuti per i parametri interni calcolati con `lsqnonlin` oppure con `fsolve` nella funzione

`f_estrinseci`

che implementa quanto descritto nel paragrafo 2.4.2, si dovrebbero poter stimare univocamente questi ultimi due valori, necessari per la completa calibrazione della telecamera. Vediamo ora nel dettaglio perché non è stato possibile farlo.

Nella figura 3.6 è possibile vedere come l'intersezione delle due quadriche generate dal fittaggio dei contorni apparenti di due differenti sfere e dal sistema di due invarianti di tipo S_2

$$\begin{aligned} b * (b * d - a * e) * f_e^2 - e * (b * f - d * e) * (1^2 - 1) &= 0 \\ b2 * (b2 * d2 - a2 * e2) * f_e^2 - e2 * (b2 * f2 - d2 * e2) * (1^2 - 1) &= 0 \end{aligned}$$

sia una curva. Quindi in caso non si conosca la matrice di calibrazione K , come possiamo determinare quale punto, tra gli infiniti possibili appartenenti a questa curva, sia quello giusto? Purtroppo in [1] non si fa riferimento a niente del genere. In Oltre, considerando anche le Fig. 3.7, 3.8, la curva di intersezione è differente per ogni coppia di quadriche generate a partire da sfere diverse. Allora come si può sapere se ci sono curve *giuste* o *sbagliate*? In teoria la calibrazione non dovrebbe dipendere così fortemente dal posizionamento delle sfere nello spazio. Pertanto dovremmo ottenere risultati del tutto simili per ogni coppia di sfere in esame.

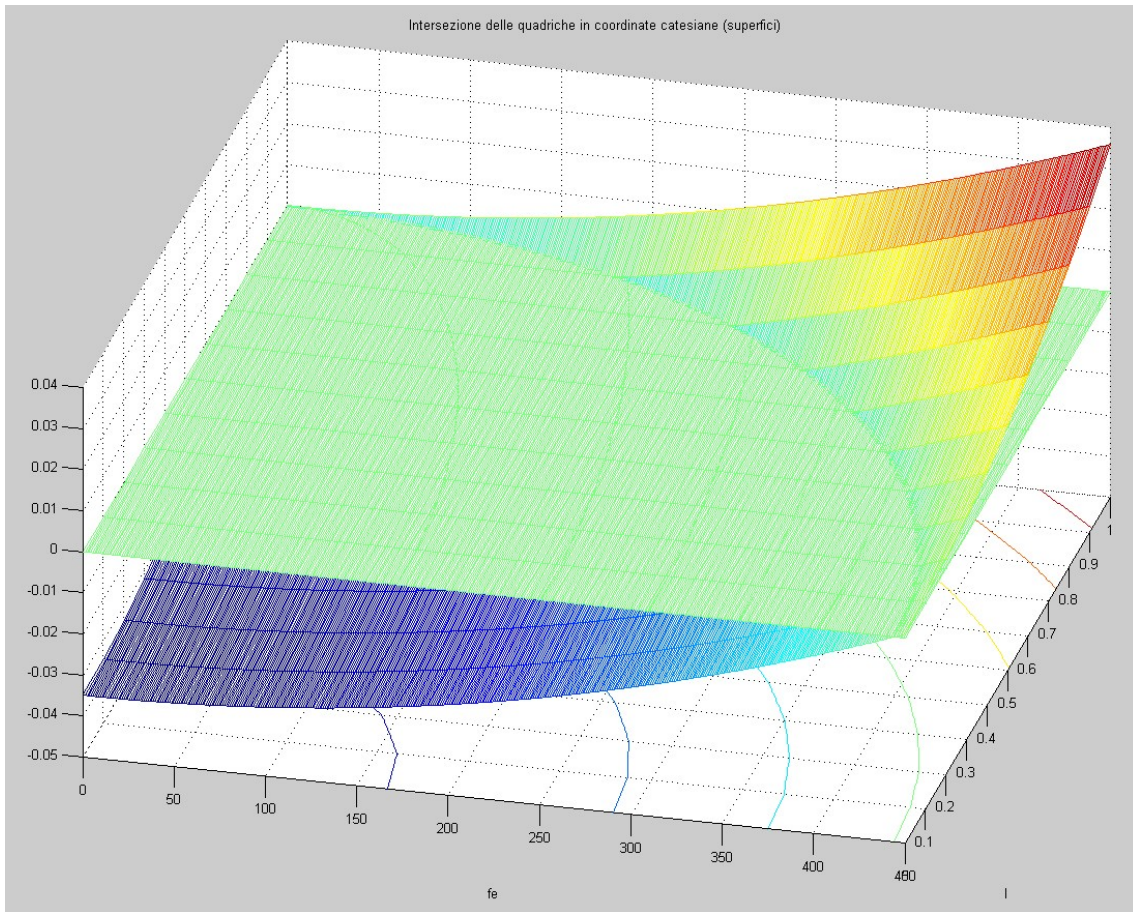


Fig. 3.6 Intersezione di due quadriche ottenute a partire da altrettante sfere

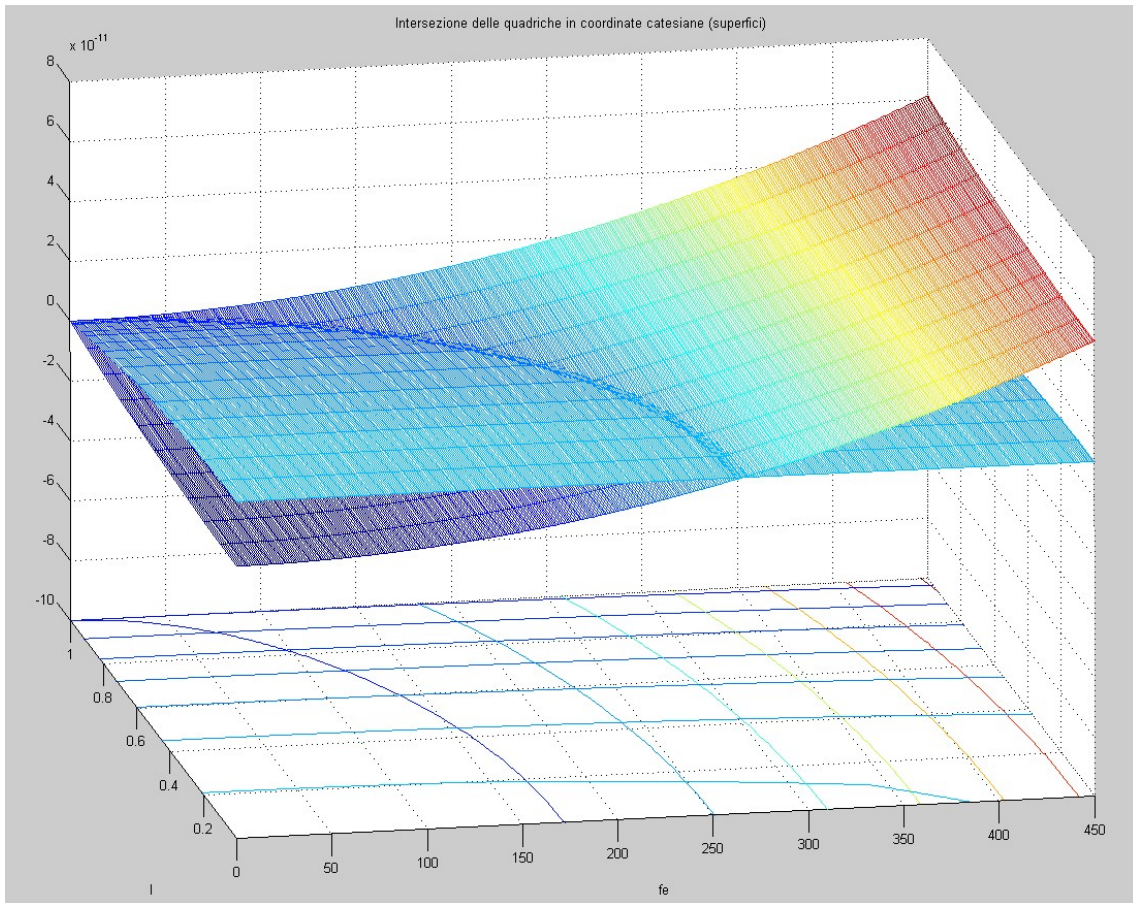


Fig. 3.7 Intersezione ottenuta da un'altra coppia di quadriche

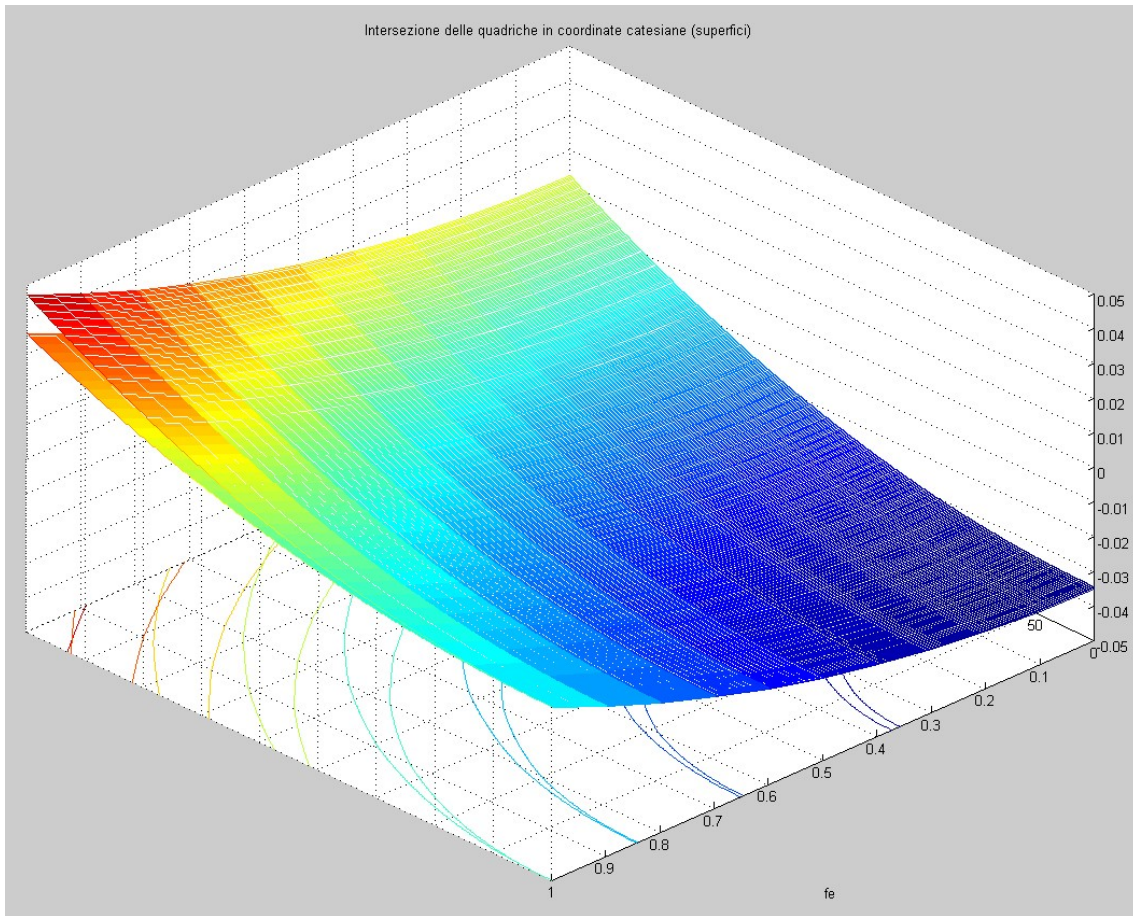


Fig. 3.8: In questa immagine le quadriche sono addirittura quasi coincidenti, sebbene vengano da due sfere distinte ed addirittura si ha intersezione solo $f_e=0$

Non essendoci informazioni in merito nell' articolo, ho mandato alcune e-mail ai due ricercatori, sperando che potessero darmi delle spiegazioni in merito. Ying (l' unico che ad oggi mi abbia risposto), sostiene, contrariamente a quanto scritto nell' articolo [1], come non sia possibile ricavare entrambe le variabili l ed f_e tramite l' invariante S_2 . Quindi mi ha suggerito di supporre l conosciuto a priori nel calcolo del secondo invariante. In effetti le dimensioni dello specchio possono essere misurate manualmente, per cui, sebbene sia restrittiva, è comunque una soluzione percorribile. Inserendo allora l come ulteriore valore noto in S_2 , si può ricavare f_e con l'utilizzo dell' immagine di una sola sfera e si ottiene

$$f_e = \sqrt{\frac{e(bf - de)}{b(bd - ae)}}(l^2 - 1) \quad (3.3)$$

Così facendo però sorge un altro problema di non poco conto. Il valore di f_e è variabile a seconda del contorno apparente utilizzato! In simulazione infatti otteniamo

fe_conoscendo_elle =

114.8012

fe2_conoscendo_elle =

467.6636

fe3_conoscendo_elle =

44.5241

fe4_conoscendo_elle =

99.5439

Ancora una volta ci troviamo nella condizione di non sapere quale tra i precedenti valori potrebbe essere quello corretto. In ogni caso (siccome stiamo svolgendo una simulazione) nessuno di questi lo è. Solo il secondo valore si avvicina (rispetto agli altri) a quello stabilito prima nella (3.1), ma date le circostanze, sembra essere solo una casualità. Ho fatto presente questa situazione ad Ying, ma non ho ricevuto risposte.

3.4.1 Critiche al metodo

Le possibili motivazioni possono essere ricercate nei passi cruciali per la definizione degli invarianti. Mi riferisco alla (2.3) che definisce la proiezione dell'immagine della sfera sul piano immagine, che potrebbe essere stata mal definita. Alternativamente l'errore potrebbe risiedere nella soluzione (del tutto non banale) rispetto a λ della (2.11). In effetti gli invarianti si ricavano proprio da quest'ultima operazione. Ancora, potrebbe esserci qualche errore nel fittaggio, e quindi nella sostituzione dei valori numerici in C'_s , ma ciò è da escludere per via degli ottimi risultati ottenibili nella prima fase dell' algoritmo. In fine si potrebbe ricercare l'errore nei valori di l ed f_e , ma ho usato gli stessi che utilizzano Ying ed Hu in [1], dove, seppur sotto la condizione di considerare come noto l , riescono a portare a termine la calibrazione.

4 Conclusioni e sviluppi futuri

Allo stato attuale, non è stato possibile portare a termine il processo di calibrazione, per i motivi sopra esposti. In conclusione, non è detto che gli invarianti S_1 ed S_2 siano del tutto sbagliati, ma nella migliore delle ipotesi, per ottenere dei risultati corretti, bisognerebbe porsi in condizioni molto particolari e restrittive, che non vengono adeguatamente indicate dai due ricercatori. Dico questo dopo aver fatto un numero molto elevato di prove e cambiamenti, per esempio nel posizionamento delle sfere, senza ottenere però alcun miglioramento. Un altro fatto da sottolineare è che nel loro articolo viene proposto un metodo di calibrazione, simile ma facente uso di linee parallele piuttosto che di sfere come esposto finora. Ebbene, nonostante la prima parte dell' algoritmo funzioni bene, anche se con risultati meno precisi, come d'altronde era lecito aspettarsi) per quanto detto riguardo ai problemi di fitting descritti nel paragrafo 2.1), ho riscontrato problemi analoghi per la determinazione dei due valori di l ed f_e : Anche in questo caso non è stato possibile stimarli correttamente.

Un metodo preciso ed affidabile per la calibrazione dei sistemi catadiottrici è di importanza fondamentale per tutte le applicazioni di computer vision, per cui visti i buoni risultati della prima fase, sarebbe interessante trovare un sistema per risolvere i problemi legati alla seconda parte dell' algoritmo di Ying ed Hu, anche se la soluzione non sembra essere del tutto a portata di mano. Mi riferisco in particolare alla correzione del secondo invariante (ammesso che sia possibile), oppure all' individuazione di quelle condizioni particolari che potrebbero renderlo valido, sperando che possano essere replicate anche nella realtà.

Appendice A

Lo specchio iperbolico

Si consideri un'iperbole come quella mostrata in figura

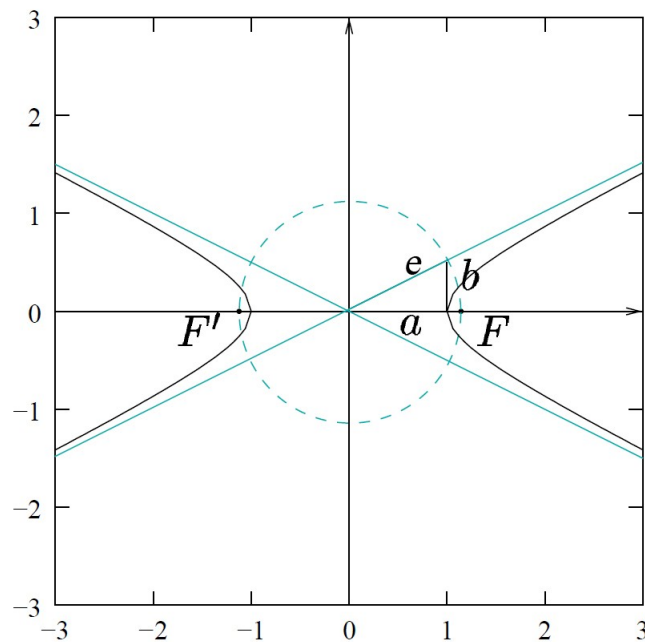


Fig. A.1: Iperbole sugli assi cartesiani

Analiticamente è esprimibile come

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \quad (\text{A.1})$$

Dove a rappresenta l'asse maggiore, b l'asse minore ed e la distanza del fuoco dall'origine. Lo specchio iperbolico, è definibile come la rotazione intorno all'asse x di uno dei due rami dell'iperbole. Per le nostre stime è utile definirne anche l'eccentricità ε

$$\varepsilon = \sqrt{1 + \frac{b^2}{a^2}} \quad (\text{A.2})$$

ed il parametro l funzione di essa

$$l = \frac{2\varepsilon}{1 + \varepsilon^2} \quad (\text{A.3})$$

Come si vede, la conoscenza di a e b , misurabili manualmente, permette di ricavare l , per esempio nel caso in cui debba essere sostituito nella (3.3) per trovare in maniera alternativa f_e .

Appendice B

Le funzioni implementate in MATLAB

```
%% Calibrazione in Simulazione

% Simulo la telecamera e quattro sfere per la calibrazione
% Inizializzazione
clear all
close all
clc
%Creo il World Frame
figure(1);
hold on
axis equal
grid on
title ('Spazio virtuale con telecamera e sfere')
f_3Dwf('k',3);
%% Scelgo una posizione arbitraria per la telecamera nello spazio
R=rotoy(pi);
t=[10,10,0]';
H=f_Rt2H(rotoy(pi),[10,10,0]'); % Matrice H coordinate omogenee per la
%rotazione e la traslazione della telecamera

%Creo la Telecamera
a=3.5924;
b=3;
r_rim=5;
f_3Dpanoramic(H,'g',1,a,b,r_rim); % Camera con specchio Iperbolico con
%a=3cm b=1cm r_rim=2cm

K=[400 0 500; % Matrice dei parametri interni K
0 400 500;
0 0 1];
f_3Dframe(H,'g',2); % Sys di rif. delle telecamera

%Creo le sfere
[x,y,z,nx,ny,nz]=f_3Dsurface(1,rotox(pi/2),[10 10 5]',[1 1],25,1);
[x1,y1,z1,nx1,ny1,nz1]=f_3Dsurface(1,rotox(pi/2),[5 10 0]',[1
1],25,1);
[x2,y2,z2,nx2,ny2,nz2]=f_3Dsurface(1,rotox(pi/2),[15 10 0]',[1
1],25,1);
[x3,y3,z3,nx3,ny3,nz3]=f_3Dsurface(1,rotox(pi/2),[10 10 -5]',[1
1],25,1);
```

```

% Una volta creata la telecamera e le sfere plotto la bounding
% ellipse della telecamera ed i contorni apparenti delle sfere nello
% spazio
figure (2);
hold on
axis equal
grid on
title ('Contorni delle sfere e della bounding ellipse nello spazio
(ancora da fittare)')
% Calcolo e plotto i punti della bounding ellipse nello spazio
theta = 0:.1:2*pi;
z_rim = (a/b)*sqrt(r_rim^2+b^2)-sqrt(a^2+b^2);
Rrpy = H([1:3],[1:3]); % Rrpy = Rwf2mir
circle = [r_rim*cos(theta); r_rim*sin(theta);
z_rim*ones(1,length(theta))];
circleMAT = [Rrpy , t; 0 0 0 , 1]*[circle; ones(1,length(theta))];
plot3(circleMAT(1,:),circleMAT(2,:),circleMAT(3,),'k*');

% Calcolo e plotto i punti del generatore di contorno delle sfere
nello spazio
[x_genc,y_genc,z_genc]=f_gencon(x,y,z,nx,ny,nz,t,0.3);
[x_genc1,y_genc1,z_genc1]=f_gencon(x1,y1,z1,nx1,ny1,nz1,t,0.3);
[x_genc2,y_genc2,z_genc2]=f_gencon(x2,y2,z2,nx2,ny2,nz2,t,0.3);
[x_genc3,y_genc3,z_genc3]=f_gencon(x3,y3,z3,nx3,ny3,nz3,t,0.3);
plot3(x_genc,y_genc,z_genc,'g*');
plot3(x_genc1,y_genc1,z_genc1,'b*');
plot3(x_genc2,y_genc2,z_genc2,'y*');
plot3(x_genc3,y_genc3,z_genc3,'r*');

%%Plotto quello che vedo sul CCD della Telecamera. Questa immagine è
ciò
%%che devo considerare per la calibraizone
figure (3);
hold on
axis equal
grid on
title ('Immagine ripresa dalla Telecamera Virtuale')
%%proiezione e plottaggio per la bounding ellipse
Xb=[circleMAT(1,);circleMAT(2,);circleMAT(3,)];
qb= f_panproj(Xb,H,K,a,b);
plot(qb(1,:),qb(2,),'ko');

%%proiezione e plottaggio per le sfere
Xs=[x_genc;y_genc;z_genc];
qs= f_panproj(Xs,H,K,a,b); %% contorno apparente della prima sfera
plot(qs(1,:),qs(2,),'go');

Xs1=[x_genc1;y_genc1;z_genc1];
qs1= f_panproj(Xs1,H,K,a,b); %% contorno apparente della seconda sfera
plot(qs1(1,:),qs1(2,),'bo');

Xs2=[x_genc2;y_genc2;z_genc2];
qs2= f_panproj(Xs2,H,K,a,b); %% contorno apparente della terza sfera
plot(qs2(1,:),qs2(2,),'yo');

```

```

Xs3=[x_genc3;y_genc3;z_genc3];
qs3= f_panproj(Xs3,H,K,a,b); %% contorno apparente della quarta sfera
plot(qs3(1,:),qs3(2:),'ro');
title ('Immagine ripresa dalla Telecamera Virtuale')
%% Ordino,fitto e plotto le ellissi a partire dai punti del contorno
%% apparente
figure (3);
hold on
axis equal
grid on
xlabel ('u')
ylabel('v')
title ('Ellissi fittate')
%% Ordinamento
[qb_u_ord,qb_v_ord]=f_ordinamento(qb(1,:),qb(2,:)); %% Bounding
%%ellipse

[qs_u_ord,qs_v_ord]=f_ordinamento(qs(1,:),qs(2,:)); %% Prima sfera
[qs1_u_ord,qs1_v_ord]=f_ordinamento(qs1(1,:),qs1(2,:)); %% Seconda
%%sfera
[qs2_u_ord,qs2_v_ord]=f_ordinamento(qs2(1,:),qs2(2,:)); %% Terza sfera
[qs3_u_ord,qs3_v_ord]=f_ordinamento(qs3(1,:),qs3(2,:)); %% Quarta
%%sfera

%% Fittaggio
[Xc,Yc,A,B,Phi,P5]=ellipsefit(qb_u_ord,qb_v_ord); %% Bounding ellipse

[Xc1,Yc1,A1,B1,Phi1,P1]=ellipsefit(qs_u_ord,qs_v_ord); %% Prima sfera
[Xc2,Yc2,A2,B2,Phi2,P2]=ellipsefit(qs1_u_ord,qs1_v_ord); %% Seconda
%%sfera
[Xc3,Yc3,A3,B3,Phi3,P3]=ellipsefit(qs2_u_ord,qs2_v_ord); %% Terza
%%sfera
[Xc4,Yc4,A4,B4,Phi4,P4]=ellipsefit(qs3_u_ord,qs3_v_ord); %% Quarta
%%sfera

%% Plotting
syms x y
C5=P5(1)*x^2 + P5(2)*x*y + P5(3)*y^2 + P5(4)*x + P5(5)*y + P5(6); %%
Bounding ellipse
ezplot (C5,[0,700]);

C1=P1(1)*x^2 + P1(2)*x*y + P1(3)*y^2 + P1(4)*x + P1(5)*y + P1(6); %%
Prima sfera
ezplot (C1,[0,700]);
C2=P2(1)*x^2 + P2(2)*x*y + P2(3)*y^2 + P2(4)*x + P2(5)*y + P2(6); %%
Seconda sfera
ezplot (C2,[0,700]);
C3=P3(1)*x^2 + P3(2)*x*y + P3(3)*y^2 + P3(4)*x + P3(5)*y + P3(6); %%
Terza sfera
ezplot (C3,[0,700]);
C4=P4(1)*x^2 + P4(2)*x*y + P4(3)*y^2 + P4(4)*x + P4(5)*y + P4(6); %%
Quarta sfera
ezplot (C4,[0,700]);
title ('Ellissi fittate')
xlabel ('u')
ylabel('v')

```



```

function S1=f_intrinseci(pi, Pc1, Pc2, Pc3, Pc4)

apr=Pc1(1);
bpr=Pc1(2)/2;
cpr=Pc1(3);
dpr=Pc1(4)/2;
epr=Pc1(5)/2;
fpr=Pc1(6);

apr2=Pc2(1);
bpr2=Pc2(2)/2;
cpr2=Pc2(3);
dpr2=Pc2(4)/2;
epr2=Pc2(5)/2;
fpr2=Pc2(6);

apr3=Pc3(1);
bpr3=Pc3(2)/2;
cpr3=Pc3(3);
dpr3=Pc3(4)/2;
epr3=Pc3(5)/2;
fpr3=Pc3(6);

apr4=Pc4(1);
bpr4=Pc4(2)/2;
cpr4=Pc4(3);
dpr4=Pc4(4)/2;
epr4=Pc4(5)/2;
fpr4=Pc4(6);

%soluzioni di C=Ka'*C1*Ka per la prima conica
a= pi(1)^2 * apr;
b= pi(1) * pi(2) * apr + pi(1) * bpr;
c= pi(2)^2 * apr + 2 * pi(2) * bpr + cpr;
d= pi(1) * pi(3) * apr + pi(1) * pi(4) * bpr + pi(1) * dpr;
e= pi(2) * pi(3) * apr + pi(3) * bpr + pi(2) * pi(4) * bpr + pi(4) *
cpr + pi(2) * dpr + epr;
f= pi(3)^2 * apr + 2 * pi(3) * pi(4) * bpr + pi(4)^2 * cpr + 2 * pi(3)
* dpr + 2 * pi(4) * epr + fpr;

%soluzioni di C=Ka'*C2*Ka per la seconda conica
a2= pi(1)^2 * apr2;
b2= pi(1) * pi(2) * apr2 + pi(1) * bpr2;
c2= pi(2)^2 * apr2 + 2 * pi(2) * bpr2 + cpr2;
d2= pi(1) * pi(3) * apr2 + pi(1) * pi(4) * bpr2 + pi(1) * dpr2;
e2= pi(2) * pi(3) * apr2 + pi(3) * bpr2 + pi(2) * pi(4) * bpr2 + pi(4)
* cpr2 + pi(2) * dpr2 + epr2;
f2= pi(3)^2 * apr2 + 2 * pi(3) * pi(4) * bpr2 + pi(4)^2 * cpr2 + 2 *
pi(3) * dpr2 + 2 * pi(4) * epr2 + fpr2;

```

```

%soluzioni di C=Ka'*C3*Ka per la terza conica
a3= pi(1)^2 * apr3;
b3= pi(1) * pi(2) * apr3 + pi(1) * bpr3;
c3= pi(2)^2 * apr3 + 2 * pi(2) * bpr3 + cpr3;
d3= pi(1) * pi(3) * apr3 + pi(1) * pi(4) * bpr3 + pi(1) * dpr3;
e3= pi(2) * pi(3) * apr3 + pi(3) * bpr3 + pi(2) * pi(4) * bpr3 + pi(4)
* cpr3 + pi(2) * dpr3 + epr3;
f3= pi(3)^2 * apr3 + 2 * pi(3) * pi(4) * bpr3 + pi(4)^2 * cpr3 + 2 *
pi(3) * dpr3 + 2 * pi(4) * epr3 + fpr3;

%soluzioni di C=Ka'*C4*Ka per la quarta conica
a4= pi(1)^2 * apr4;
b4= pi(1) * pi(2) * apr4 + pi(1) * bpr4;
c4= pi(2)^2 * apr4 + 2 * pi(2) * bpr4 + cpr4;
d4= pi(1) * pi(3) * apr4 + pi(1) * pi(4) * bpr4 + pi(1) * dpr4;
e4= pi(2) * pi(3) * apr4 + pi(3) * bpr4 + pi(2) * pi(4) * bpr4 + pi(4)
* cpr4 + pi(2) * dpr4 + epr4;
f4= pi(3)^2 * apr4 + 2 * pi(3) * pi(4) * bpr4 + pi(4)^2 * cpr4 + 2 *
pi(3) * dpr4 + 2 * pi(4) * epr4 + fpr4;

S1=[d*(b*d-a*e)-e*(b*e-c*d);           %% Sistema da risolvere
d2*(b2*d2-a2*e2)-e2*(b2*e2-c2*d2);
d3*(b3*d3-a3*e3)-e3*(b3*e3-c3*d3);
d4*(b4*d4-a4*e4)-e4*(b4*e4-c4*d4)];

```

```

%% Stima dei parametri Estrinseci l ed fe

%% Prendo i parametri interni appena stimati ed i parametri di due
%%coniche fittate, in questo caso la terza e la quarta. Sostituendo
%% questi valori nelle espressioni di  $C=Ka'*Cpr*Ka$  e nelle invarianti
%%S2, ottengo due quadriche, la cui intersezione mi dovrebbe dare l ed
fe.

function [S2,S2_2,fe,l]=f_estrinseci(pi,Pc1,Pc2)
%% Rinomino i parametri interni
r=pi(1,1);
s1=pi(1,2);
u0=pi(1,3);
v0=pi(1,4);
%% Rinomino i parametri delle coniche
apr=Pc1(1);           %% Prima conica
bpr=Pc1(2)/2;
cpr=Pc1(3);
dpr=Pc1(4)/2;
epr=Pc1(5)/2;
fpr=Pc1(6);

apr2=Pc2(1);         %% Seconda conica
bpr2=Pc2(2)/2;
cpr2=Pc2(3);
dpr2=Pc2(4)/2;
epr2=Pc2(5)/2;
fpr2=Pc2(6);

%%soluzioni di  $A=Ka'*Apr*Ka$  per la seconda conica
a= r^2 * apr;
b= r * s1 * apr + r * bpr;
c= s1^2 * apr + 2 * s1 * bpr + cpr;
d= r * u0 * apr + r * v0 * bpr + r * dpr;
e= s1 * u0 * apr + u0 * bpr + s1 * v0 * bpr + v0 * cpr + s1 * dpr +
epr;
f= u0^2 * apr + 2 * u0 * v0 * bpr + v0^2 * cpr + 2 * u0 * dpr + 2 * v0
* epr + fpr;

%%soluzioni di  $A=Ka'*Apr2*Ka$  per la seconda conica
a2= r^2 * apr2;
b2= r * s1 * apr2 + r * bpr2;
c2= s1^2 * apr2 + 2 * s1 * bpr2 + cpr2;
d2= r * u0 * apr2 + r * v0 * bpr2 + r * dpr2;
e2= s1 * u0 * apr2 + u0 * bpr2 + s1 * v0 * bpr2 + v0 * cpr2 + s1 *
dpr2 + epr2;
f2= u0^2 * apr2 + 2 * u0 * v0 * bpr2 + v0^2 * cpr2 + 2 * u0 * dpr2 + 2
* v0 * epr2 + fpr2;

%% Rinomino per semplicità i termini numerici
alfa=b*(b*d-a*e);
beta=e*(b*f-d*e);
gamma=b2*(b2*d2-a2*e2);
delta=e2*(b2*f2-d2*e2);

```

```

%%Plotto le quadriche per verificarne l'intersezione
figure(4)
hold on
grid on;
hidden off
xlabel('fe')
ylabel('l')
title('Intersezione delle quadriche in coordinate catesiane
(superfici)')

x=0:1:450;
y=0:.1:1;
[fe,l]=meshgrid(x,y);
S2=alfa.*(fe.^2)-beta.*((l.^2)-1);
S2_2=gamma.*(fe.^2)-delta.*((l.^2)-1);
meshc(fe,l,S2);
meshc(fe,l,S2_2);

figure(5)
hold on
grid on;
hidden off
xlabel('fe')
ylabel('l')
title('Intersezione delle quadriche in coordinate catesiane
(contorni)')

contour3(fe,l,S2,20);
contour3(fe,l,S2_2,20);

%%Se conosco già l
elle=0.966
fe_conoscendo_elle=sqrt((beta/alfa)*((elle^2)-1))
fe2_conoscendo_elle=sqrt((delta/gamma)*((elle^2)-1))

```

```

%%Contorno apparente
%%devo trovare i punti del contorno apparente dell'oggetto utilizzando
le
%%normali
function [x_cg,y_cg,z_cg]=f_gencon(x,y,z,nx,ny,nz,t,soglia)

i=1;
j=1;
k=1;

while i<=size(x,1)
    while j<=size(x,2)
        if (abs([x(i,j),y(i,j),z(i,j)]-
t')*( [nx(i,j),ny(i,j),nz(i,j)]')) < soglia)
            x_cg(k)=x(i,j);
            y_cg(k)=y(i,j);
            z_cg(k)=z(i,j);
%           k=k+1;
            if k>=2 && x_cg(k)==x_cg(k-1) && y_cg(k)==y_cg(k-1) &&
                z_cg(k)==z_cg(k-1)
                k=k-2;
            end;
            k=k+1;
        end;
        j=j+1;
    end;
    j=1;
    i=i+1;
end;
%%Elimino l'ultimo elemento che è sempre ridondante

x_cg(:,k-1)=[];
y_cg(:,k-1)=[];
z_cg(:,k-1)=[];

```

```

%ordinamento punti contorni sfere
function [X_ord,Y_ord]=f_ordinamento(X_fit,Y_fit)

%inizializzo i vettori ordinati con il primo elemento dei vettori dei
punti sparsi
k=1;
X_ord(k)=X_fit(1);
Y_ord(k)=Y_fit(1);
X_fit(1)=100000;
Y_fit(1)=100000;
%hold on
%plot(X_ord(k),Y_ord(k),'g*')
for kk=1:(size(X_fit,2)-1)
    j=1;
    clear distanza;
    i=2;
    %%Scorro il vettore e calcolo le distanze
    while i<=size(X_fit,2)
        while i<=size(X_fit,2) && j<=size(X_fit,2) && X_fit(i)==100000
            distanza(j)=500000;
            i=i+1;
            j=j+1;
        end;
        if i<=size(X_fit,2)
            distanza(j)=sqrt((X_fit(i) - X_ord(k))^2 + (Y_fit(i) -
Y_ord(k))^2);
            if distanza(j)== 0
                distanza(j)=100000; %pongo ad un valore assurdo se la
distanza è uguale a zero
            end;
            j=j+1;
            i=i+1;
        end;
    end;
    d_min=min(distanza);

    %% conto quale è l'elemento a distanza minima
    count=1;
    while distanza(count)>d_min
        count=count+1;
    end;
    count;
    %%Aggiorno il vettore X ed Y ord
    k=k+1;
    X_ord(k)=X_fit(count+1);
    Y_ord(k)=Y_fit(count+1);
    %plot(X_ord(k),Y_ord(k),'g*')
    X_fit(count+1)=100000;
    Y_fit(count+1)=100000;
end;

```

Bibliografia

- [1] Xianghua Ying, Zhanyi Hu, Catadioptric Camera Calibration Using Geometric Invariants, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 26(10), pp. 1260-1271, 2004. (SCI/EI)
- [2] C. Geyer and K. Daniilidis, “Catadioptric Camera Calibration,” Proc. Seventh Int’l Conf. Computer Vision, vol. I, pp. 398-404, 1999.
- [3] J. Barreto and H. Araujo, “Geometric Properties of Central Catadioptric Line Images,” Proc. Seventh European Conf. Computer Vision, pp. 237-251, 2002.
- [4] C. Geyer and K. Daniilidis, “A Unifying Theory for Central Panoramic Systems and Practical Implications ” Long Presentation at ECCV 200, Dublin Ireland.
- [5] Gian Luca Mariottini and Domenico Prattichizzo, “Epipolar Geometry Toolbox” <http://egt.dii.unisi.it/>
- [6] Peter Corke, “Robotics Toolbox” <http://www.petercorke.com/Robotics%20Toolbox.html>
- [7] Roberto Cipolla and Peter Giblin, “Visual Motion of Curves and Surfaces” Phil. Trans. R. Soc. Lond, 356(1740):1103-1121, May 1998.
- [8] Duane Hanselman, “[Stable Direct Least Squares Ellipse Fit](#) ”

Ringraziamenti

Alla fine di ogni lavoro, grande o piccolo che sia, non si può fare a meno di ringraziare quelle persone che ci hanno accompagnato e sostenuto durante il percorso. Per questo ci tengo a dire Grazie a tutta la mia famiglia, ma soprattutto ai miei genitori, che anche se in maniera non invasiva, sono stati sempre presenti. Hanno costituito per me una solida base dalla quale procedere ed hanno così contribuito significativamente a darmi quella fiducia e quella sicurezza senza la quale non avrei potuto e non potrei fare tutte le cose per me più importanti. Grazie alle mie sorelline Alessandra e Valentina, che hanno saputo aiutarmi sempre e sulle quali so di poter contare. Dico Grazie anche ai miei compagni di casa Alessandro, Amedeo e Luigi che mi hanno sopportato (ma che a volte sono stati anche sopportati) durante questi anni di convivenza universitaria. Posso dire che abbiamo vissuto veramente come una famiglia ed avrò piacere di continuare così il più a lungo possibile. Grazie ai miei Amici, tutti! Un ringraziamento particolare va al mio Team di ricerca e sviluppo Ingegneristico e/o affini, costituito da Giovanni e Paolo. Veramente poche persone sarebbero disposte come loro ad ascoltarmi ed a perdere tempo dietro ai miei (assurdi?) progetti. In fine ci tengo a ringraziare molto sentitamente anche il Professore Antonio Pasini e la sua assistente Ilaria Cardinali, poiché hanno cercato di aiutarmi a risolvere un problema, legato allo svolgimento di questa tesi, con un tale impegno ed una tale passione, che mi hanno lasciato del tutto senza parole.

Grazie ancora a tutti quanti voi.

Carlo Alberto