UNIVERSITÀ
POLITECNICA
DELLE MARCHE

UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea triennale in Ingegneria Meccanica

MISURA DELLA DEFORMAZIONE DI UN SOLAIO TENSEGRALE TRAMITE ANALISI DI IMMAGINE

MEASUREMENT OF THE DEFORMATION OF A TENSEGRAL FLOOR BY IMAGE ANALYSIS

Relatore:

Prof. Ing. Marco Rossi

Tesi di Laurea di:

Andrea Falcioni

A.A. 2019 / 2020

Indice:

1. INTRODUZIONE 4

2. PRELIEVO DELLE IMMAGINI IN LABORATORIO 5

3. ANALISI DELLE IMMAGINI 9

3.1 STEREOSCOPIA 9

3.2 SOFTWARE MATLAB 10

3.3 CALIBRAZIONE DELLE TELECAMERE 11

***3.3.1. Parametri intrinseci* 11**

***3.3.2. Procedura di calibrazione* 12**

***3.3.3. Parametri estrinseci* 19**

***3.3.4 Procedura di stereo – calibrazione* 20**

3.4 ESTRAZIONE DELLE COORDINATE 2D 23

3.5 CODICE DI PASSAGGIO DI COORDINATE 27

4. RISULTATI 31

5. CONCLUSIONI 36

6. BIBLIOGRAFIA 39

1. Introduzione

La seguente tesi è volta alla spiegazione di un procedimento per misurare la deformazione che un modello di solaio ad elementi tesi riceve sotto un carico statico.

La misura della deformazione avviene tramite stereo-triangolazione di alcuni punti del piano. Partendo da delle immagini, verrà calcolata la posizione di questi punti nello spazio, sia in fase scarica che in fase carica del modello. Sovrapponendo in seguito le due stereo-triangolazioni si otterranno delle differenze lungo l'asse perpendicolare al piano; queste differenze di coordinate indicheranno la freccia subita dal solaio.

Il procedimento sarà diviso in due fasi:

- Una prima fase verrà svolta in laboratorio, e sarà volta al prelievo delle immagini che verranno poi analizzate in seguito; il piano da analizzare verrà deformato sotto diverse disposizioni di carico, e sì studierà il suo comportamento per ognuna di esse.
- La seconda fase consiste nell'analisi delle immagini prese in laboratorio, con l'utilizzo del software MATLAB. In linea generale tutta l'analisi punta sul passaggio di coordinate 2D dei punti sul piano, in coordinate 3D, così da poter confrontare le coordinate lungo l'asse Z, perpendicolare al piano.

2. Prelievo delle immagini in laboratorio

Il modello da analizzare è un piano in vetro, sorretto da elementi di acciaio tesi.

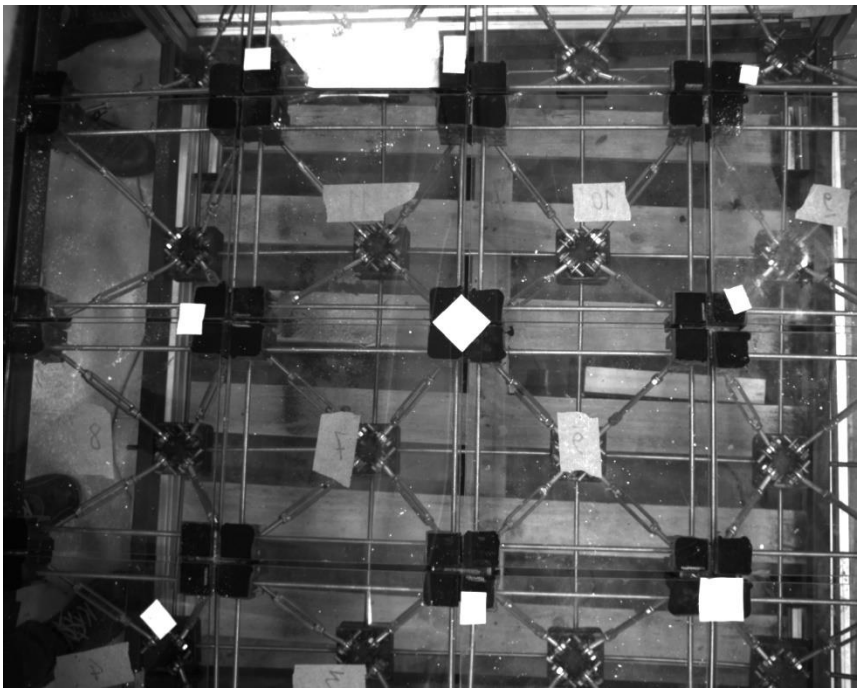


Figura 1 - modello di piano in vetro con i 9 marker fissati

Sulla superficie del piano sono stati fissati 9 marker: gli spigoli di questi marker saranno i punti che verranno poi analizzati su MATLAB, trasformando le loro coordinate da 2D a 3D. La posizione di questi punti, infatti, varierà sotto l'azione dei carichi, e ci darà un'indicazione di come il piano si sia deformato.

Sopra il piano, a circa 1 metro di altezza, sono posizionate due telecamere fisse, che riprenderanno i 9 marker da due angolazioni diverse; la stereoscopia è basata sulla

visione binoculare, che è la stessa che utilizza il nostro cervello (sfruttando i due occhi) per elaborare dimensione e distanze del mondo che ci circonda. Allo stesso modo, questo setup ci permetterà di ottenere due immagini del medesimo oggetto, al medesimo istante, ma da due prospettive differenti; sarà poi nella fase di analisi che andremo a 'sommare' queste coppie di immagini prospettiche per avere un'unica immagine tridimensionale.

Durante tutta la fase in laboratorio la posizione reciproca del piano e delle telecamere deve rimanere invariata, per non portare nella fase di stereo-triangolazione degli errori di misura.

Prima di iniziare con le prove di carico si effettua una fase di calibrazione delle telecamere, che verrà spiegata in seguito.

Le disposizioni di carico da analizzare sono le seguenti:

- 200 kg disposti nella fascia centrale del piano;

Figura 2 - Piano carico con 200 kg centrali

· 200 kg disposti nella fascia laterale del piano;

Figura 3 - Piano carico con 200 kg laterali

· 400 kg disposti uniformemente su tutto il piano;

Figura 4 - Piano carico uniformemente con 400 kg

Per ogni disposizione di carico si procede nel seguente modo:

- viene scattata un'immagine da entrambe le telecamere a piano scarico, quindi non deformato;
- i carichi, formati da sacchi di calce, vengono poi posizionati sul piano, e si scattano quindi le foto del piano deformato. A questo punto avremo una coppia di immagini del piano non deformato e una coppia di immagini del piano deformato.
- Il modello viene scaricato e si ripete il procedimento altre due volte con la stessa disposizione di carico, così da avere tre prove uguali. Questo passaggio è volto a minimizzare errori quali lo spostamento accidentale del modello durante la fase di carico, che andrebbe a compromettere la riuscita dell'intera analisi.

Una volta eseguito questo procedimento per ogni disposizione di carico siamo pronti a iniziare la fase di analisi sul software MATLAB.

3. Analisi delle immagini

3.1 Stereoscopia

Come già accennato in precedenza, il principio che sta alla base di questa analisi è la visione stereoscopica, che permette una misura delle distanze.

La visione monoscopica non ci permette di apprezzare la profondità, o la terza dimensione. Questo succede perché un qualsiasi punto P viene proiettato sulla retina della fotocamera seguendo l'asse che unisce P e il punto focale della fotocamera. Non avendo altre informazioni, la posizione di P non è univocamente determinata, in quanto esso può trovarsi in ogni punto di quest'asse, e generare la stessa immagine sulla retina della fotocamera.

Aggiungendo un'altra fotocamera affianco alla precedente la situazione cambia sensibilmente: il punto P, infatti, verrà proiettato sulla retina della seconda fotocamera seguendo un asse differente dal primo, poiché i punti focali delle fotocamere hanno posizioni differenti. Possiamo quindi ora stabilire qual è la posizione di P nello spazio, semplicemente procedendo a ritroso: unendo le proiezioni di P sulle telecamere coi rispettivi punti focali e prolungando i due assi fino ad intersezione, avremo la posizione esatta di P.

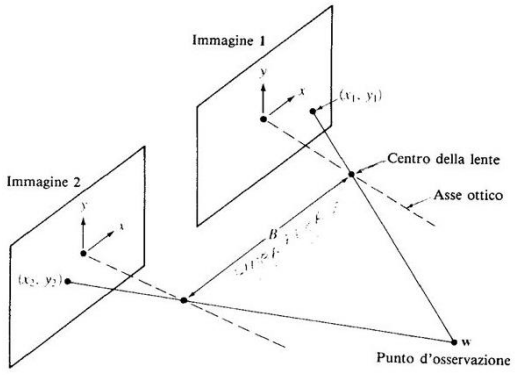


Figura 5 - Visione stereoscopica

3.2 Software MATLAB

Per l'analisi delle immagini è stato scelto il software MATLAB. Nello specifico sono state utilizzate le funzioni '*imtool*' per ordinare le coordinate 2D, 'calib' e '*stereo_gui*' per la calibrazione e la stereo-calibrazione delle telecamere.

MATLAB ci ha permesso di scrivere il codice di passaggio di coordinate in modo veloce ed efficace, avendo la possibilità di utilizzare una funzione che automatizza l'intero processo di stereo-triangolazione.

Con questo software è stato inoltre possibile plottare le coordinate 2D, e quelle 3D risultanti, in appositi piani cartesiani, così da rendere meglio visibile la posizione degli spigoli dei marker nello spazio.

È stata anche possibile, in seguito, la sovrapposizione dei piani deformati e non, creando un'immagine che visualizza la differenza di quota tra essi, quindi la freccia generata dalla presenza dei carichi.

3.3 Calibrazione delle telecamere

3.3.1. Parametri intrinseci

La calibrazione delle telecamere serve a definire i parametri intrinseci delle telecamere, che saranno poi utili nella fase di stereo-calibrazione e stereo-triangolazione. Tali parametri sono:

- Principal Point (centro di proiezione); è il punto dell'obbiettivo nel quale passano tutti i raggi che vengono proiettati su piano di proiezione. Le coordinate di tale punto vengono immagazzinate dal software nel vettore 2x1 ***cc***;

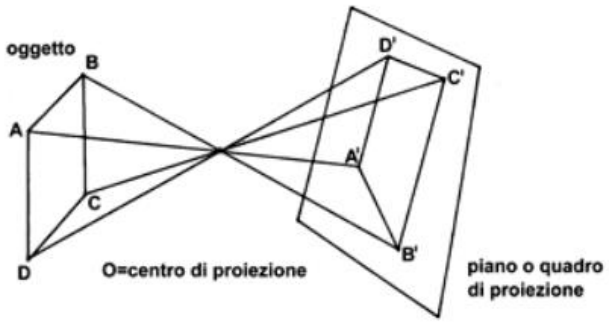


Figura 6 - Centro di proiezione

- Focal Lenght (distanza focale); è la distanza in pixel tra il centro di proiezione e il piano di proiezione. Questa sarà memorizzata nel vettore 2x1 ***fc***;

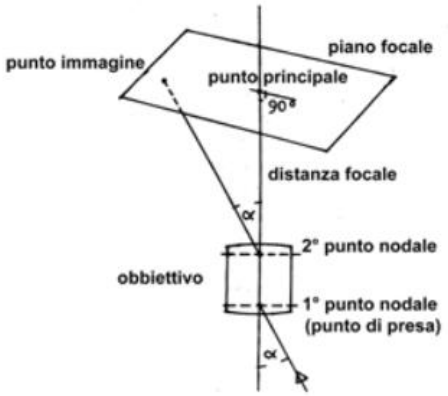


Figura 7 - Distanza focale

- Skew Coefficent (coefficiente di inclinazione); definisce l'angolo tra gli assi x e y dei pixel (in genere è nullo, il che indica un angolo di 90°). Tale coefficiente è definito dallo scalare ***alpha_c***;
- Distorsion (distorsione); le distorsioni radiali e tangenziali sono dovute alla non idealità nella fase di produzione e assemblaggio dell'ottica. Questi coefficienti sono memorizzati nel vettore 5x1 ***kc***.

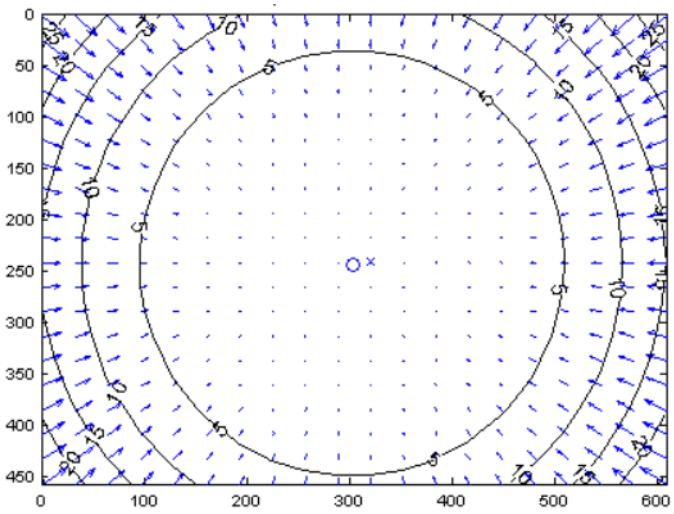


Figura 8 - Modello di distorsione di una lente

Ognuno di questi parametri è caratterizzato da un'incertezza che dipende dalla geometria stessa della telecamera e dalla precisione in fase di calibrazione.

3.3.2. Procedura di calibrazione

Qui di seguito viene riportata l'intera procedura che è stata effettuata per la calibrazione di una delle due fotocamere, con la funzione *'calib'* di MATLAB. Per la seconda fotocamera è stato utilizzato il medesimo procedimento.

Per iniziare vengono scattate, con la fotocamera da calibrare, delle foto di un pannello dotato di una griglia, in diverse posizioni. Nella prima foto il pannello è

adiacente al piano del modello; MATLAB, infatti, considera il piano del pannello nella prima foto della calibrazione come piano di riferimento delle coordinate spaziali. Avere il piano di riferimento affine a quello del modello ci permette di individuare la freccia dovuta ai carichi solamente lungo la coordinata Z, lasciando invariate le altre.

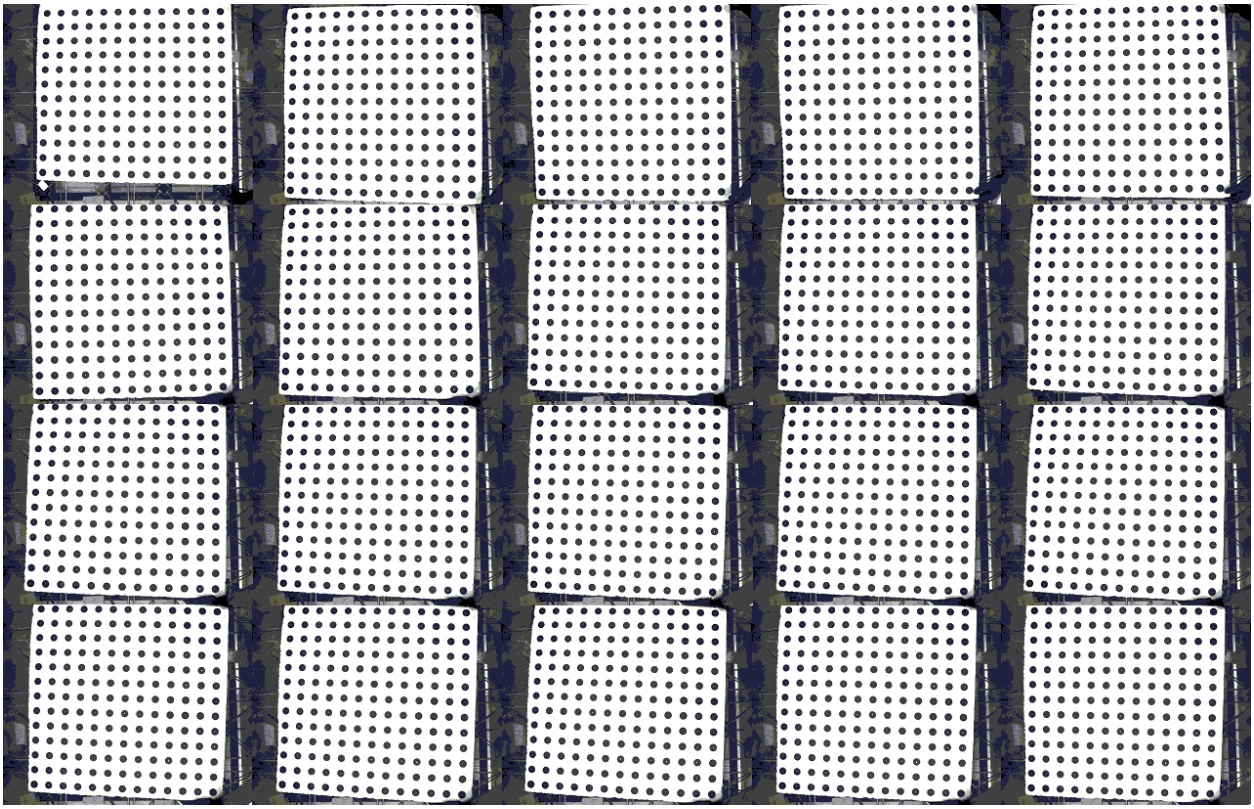


Figura 9 - Foto del pannello con griglia utilizzate per la calibrazione della fotocamera

Le immagini dovranno essere salvate nella stessa cartella, nominandole tutte con lo stesso suffisso, seguito da un numero (es. Calib_0, Calib_1, …).

Si può quindi procedere con la calibrazione vera e propria, digitando *'calib'* sulla *Command Window* di MATLAB.

Dopo aver scelto su che memoria vogliamo lavorare, si aprirà la finestra principale della toolbox, con la quale effettueremo le operazioni di calibrazione.

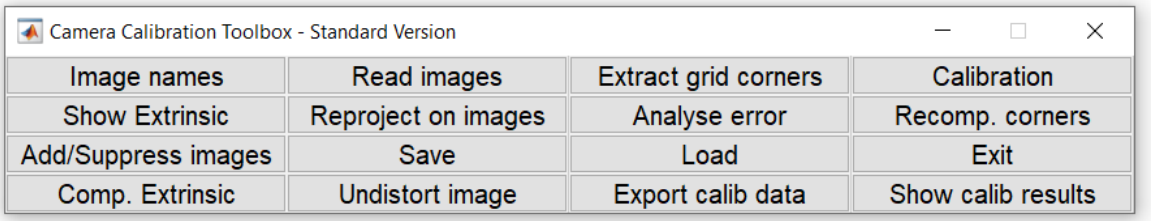


Figura 10 - Finestra principale di 'calib'

- ***Lettura delle immagini***

Una volta posizionati nella cartella dove sono presenti le immagini utili alla calibrazione, clicchiamo su *Image names*. Scriviamo quindi il suffisso delle immagini, e il formato dei file.

```
.                     CALIB_cam0_19.bmp     CALIB_cam0_27.bmp     CALIB_cam0_8.bmp
..                    CALIB_cam0_21.bmp     CALIB_cam0_28.bmp     Calib_Results.m
CALIB_cam0_0.bmp      CALIB_cam0_22.bmp     CALIB_cam0_29.bmp     Calib_Results.mat
CALIB_cam0_10.bmp     CALIB_cam0_23.bmp     CALIB_cam0_4.bmp      Calib_Results_old0.m
CALIB_cam0_14.bmp     CALIB_cam0_24.bmp     CALIB_cam0_5.bmp      Calib_Results_old0.mat
CALIB_cam0_15.bmp     CALIB_cam0_25.bmp     CALIB_cam0_6.bmp      calib_data.mat
CALIB_cam0_16.bmp     CALIB_cam0_26.bmp     CALIB_cam0_7.bmp

Basename camera calibration images (without number nor suffix): CALIB_cam0_
Image format: ([]='r'='ras', 'b'='bmp', 't'='tif', 'p'='pgm', 'j'='jpg', 'm'='ppm') b
Loading image 1...5...6...7...8...9...11...15...16...17...20...22...23...24...25...26...27...28...29...30...
done
>>
```

- ***Estrazione della griglia***

Caricate le immagini, possiamo cliccare su *Extract grid corners*, e premiamo *Invio* per selezionare tutte le immagini.

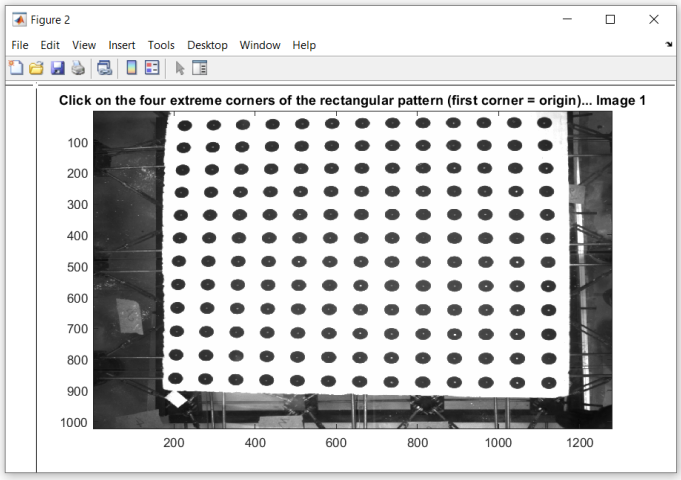
```
Extraction of the grid corners on the images
Number(s) of image(s) to process ([] = all images) =
Window size for corner finder (wintx and winty):
wintx ([] = 11) =
```

Ora decidiamo le dimensioni (in pixel) della finestra entro la quale ricercare lo spigolo (nel nostro caso il punto bianco al centro dei cerchi neri). Selezioniamo una finestra di 5x5 pixel.

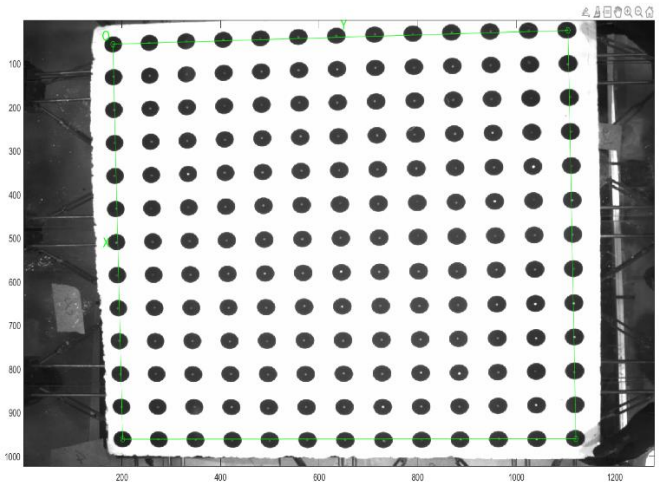
A questo punto MATLAB include un meccanismo di conteggio automatico del numero di quadrati presenti nella griglia. Siccome in alcuni casi potrebbe non predire il numero esatto di quadrati, scegliamo di scriverlo manualmente, premendo *1* e poi *Invio*.

```
Extraction of the grid corners on the images
Number(s) of image(s) to process ([] = all images) =
Window size for corner finder (wintx and winty):
wintx ([] = 11) = 5
winty ([] = 11) = 5
Window size = 11x11
Do you want to use the automatic square counting mechanism (0=[]=default)
  or do you always want to enter the number of squares manually (1,other)? 1

Processing image 1...
Using (wintx,winty)=(5,5) - Window size = 11x11      (Note: To reset the window size, run script clearwin)
Click on the four extreme corners of the rectangular complete pattern (the first clicked corner is the origin)...
```

Viene quindi mostrata la prima immagine della calibrazione, nella quale andiamo a cliccare gli angoli della griglia, cercando di essere il più precisi possibile, poiché abbiamo una finestra di ricerca di soli 5 pixel.



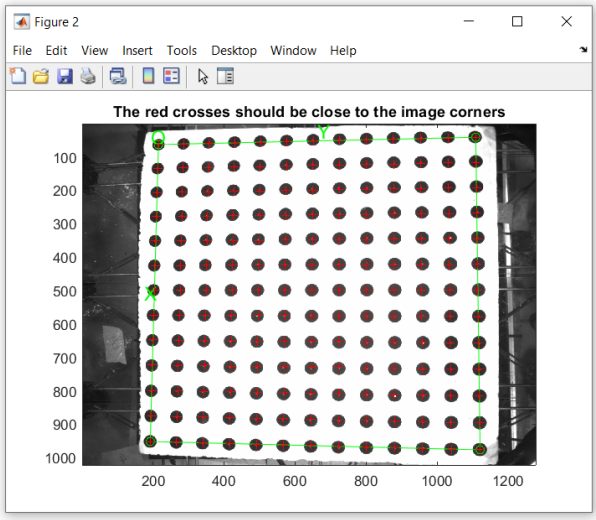
L’ordine con cui si cliccano gli angoli è fondamentale: il primo angolo selezionato sarà infatti l’origine del nostro sistema di riferimento, e dovendo fare una stereo-calibrazione, abbiamo bisogno che i due sistemi di riferimento abbiano l’origine nello stesso punto della griglia. Per convenzione scegliamo l’angolo in alto a sinistra come origine del sistema di riferimento, e procediamo in senso orario.



Ora scriviamo il numero di quadrati lungo la direzione X e Y, e la loro dimensione.

```
Number of squares along the X direction ([]=10) = 12
Number of squares along the Y direction ([]=10) = 12
Size of each square along the X direction: dX=59.5mm
Size of each square along the Y direction: dY=59.5mm
```

Il programma calcola quindi la griglia completa in assenza di distorsione.



Nel caso in cui la griglia calcolata non si adatti bene a quella reale, è possibile supporre un coefficiente di distorsione della lente.

```
If the guessed grid corners (red crosses on the image) are not close to the actual corners,
it is necessary to enter an initial guess for the radial distortion factor kc (useful for subpixel detection)
Need of an initial guess for distortion? ([]=no, other=yes)
Corner extraction...
```

Il caso in esame è sufficientemente preciso, quindi saltiamo questo passaggio premendo *Invio*.

Ripetiamo questi passaggi per tutte le immagini fino a completa operazione.

Al termine il programma salva tutti i dati nel file *calib_data.mat*, che verrà automaticamente modificato qualora venga ripetuto il procedimento di estrazione della griglia.

- **Calibrazione principale**

Ottenuto il file *calib_data.mat* possiamo procedere con la calibrazione della fotocamera. Clicchiamo quindi su *Calibration* nella finestra principale della toolbox, per avviare la calibrazione.

```
Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated (DEFAULT).
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point, set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
     Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .
Initialization of the principal point at the center of the image.
Initialization of the intrinsic parameters using the vanishing points of planar patterns.

Initialization of the intrinsic parameters - Number of images: 20


Calibration parameters after initialization:

Focal Length:          fc = [ 1879.90637   1879.90637 ]
Principal point:       cc = [ 639.50000   511.50000 ]
Skew:             alpha_c = [ 0.00000 ]   => angle of pixel = 90.00000 degrees
Distortion:            kc = [ 0.00000   0.00000   0.00000   0.00000   0.00000 ]

Main calibration optimization procedure - Number of images: 20
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...22...23...24...25
Estimation of uncertainties...done
```

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

Focal Length:          fc = [ 1839.76142   1837.64850 ] +/- [ 70.41831   69.85298 ]
Principal point:       cc = [ 691.08884   557.14108 ] +/- [ 9.69664   16.33168 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ -0.16020   0.29370   -0.00010   -0.00053  0.00000 ] +/- [ 0.01954   0.11152   0.00128   0.00089  0.00000 ]
Pixel error:          err = [ 1.07277   1.08679 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

Il programma rileva quindi i parametri intrinseci, le incertezze su di essi, e salva il tutto nelle relative variabili. Le variabili vengono immagazzinate nel file *Calib_Results.mat*. Questo sarà il file che, una volta rinominato, useremo nella stereo-calibrazione.

Proviamo ora a ridurre l'incertezza sulle variabili con il seguente metodo.

Clicchiamo su *Reproject on images* nella finestra principale della toolbox, e clicchiamo invio. Clicchiamo poi su *Recomp. Corners*, scegliamo la finestra di rilievo degli spigoli della griglia (5x5 nel nostro caso), e clicchiamo invio. Con queste due operazioni il programma ricalcolerà e ridisegnerà le griglie seguendo i parametri estratti nella calibrazione principale.

```
Loading image 1...6...7...13...22...23...24...25...26...27...31...32...33...43...44...45...49...50...51...52...
done
Number(s) of image(s) to show ([] = all images) =
Pixel error:      err = [1.07277   1.08679] (all active images)


Re-extraction of the grid corners on the images (after first calibration)
Window size for corner finder (wintx and winty):
wintx ([] = 5) =
winty ([] = 5) =
Window size = 11x11
Number(s) of image(s) to process ([] = all images) =
Use the projection of 3D grid or manual click ([]=auto, other=manual):
Processing image 1...6...7...13...22...23...24...25...26...27...31...32...33...43...44...45...49...50...51...52...
done
```

Avviamo quindi una seconda calibrazione, cliccando su *Calibration*, nella finestra principale della toolbox.

```
Aspect ratio optimized (est_aspect_ratio = 1) -> both components of fc are estimated (DEFAULT).
Principal point optimized (center_optim=1) - (DEFAULT). To reject principal point, set center_optim=0
Skew not optimized (est_alpha=0) - (DEFAULT)
Distortion not fully estimated (defined by the variable est_dist):
     Sixth order distortion not estimated (est_dist(5)=0) - (DEFAULT) .

Main calibration optimization procedure - Number of images: 20
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...18...19...20...21...22...23...24...25...26
Estimation of uncertainties...done


Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 1764.00628   1763.40303 ] +/- [ 65.85248   65.56337 ]
Principal point:       cc = [ 699.13708   534.12044 ] +/- [ 8.53520   13.82463 ]
Skew:             alpha_c = [ 0.00000 ] +/- [ 0.00000  ]   => angle of pixel axes = 90.00000 +/- 0.00000 degrees
Distortion:            kc = [ -0.18124   0.43606   0.00116   -0.00015  0.00000 ] +/- [ 0.01943   0.10867   0.00112   0.00084  0.00000 ]
Pixel error:          err = [ 1.03842   1.02240 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

Possiamo vedere una effettiva riduzione degli errori sui parametri.

Ora il programma sostituisce queste variabili nel file *Calib_Results.mat*, e crea un'ulteriore file, *Calib_Results_old0.mat*, dove troveremo i risultati della precedente calibrazione.

Ripetiamo ora tutto il procedimento di calibrazione per la seconda telecamera.

3.3.3. Parametri estrinseci

La procedura di stereo-calibrazione è molto più breve, ed è incentrata sull'estrazione dei due parametri estrinseci che descrivono la posizione relativa delle due fotocamere:

- Vettore rotazione, salvato come matrice 1x3, denominato ***om***;
- Vettore traslazione, salvato come matrice 1x3, denominato ***T***.

Sappiamo infatti, dalla calibrazione, che il sistema di riferimento rispetto alle due fotocamere è lo stesso; per passare dalle coordinate relative alla prima fotocamera, a quelle relative alla seconda, basterà utilizzare la semplice equazione:

$$X_R = R * X_L + T$$

Dove:

- $\boldsymbol{X_R}$ = coordinate rispetto alla fotocamera di destra
- $\boldsymbol{X_L}$ = coordinate rispetto alla fotocamera di sinistra
- ***T*** = vettore traslazione dato dalla stereo-calibrazione
- ***R*** = matrice di rotazione, dato dalla funzione ***R=rodrigues(om)***

La formula di Rodrigues è la seguente:

$$\boldsymbol{v_{rot} = v\cos\alpha + (k \times v)\sin\alpha + k(k \cdot v)(1-\cos\alpha)}$$

In questo caso:

- $\boldsymbol{v_{rot}}$ è il vettore ruotato
- ***v*** è il vettore iniziale
- $\boldsymbol{\alpha}$ è il vettore di rotazione (***om****)*
- ***k*** è l'asse di rotazione

3.3.4 Procedura di stereo – calibrazione

Vediamo ora come effettuare la stereo-calibrazione delle fotocamere grazie alla toolbox '*stereo_gui* '.

Come già accennato, da ora in poi parleremo di 'fotocamera di sinistra' e 'fotocamera di destra'; riportiamo quindi i risultati delle due calibrazioni in un'unica cartella e li rinominiamo come *Calib_Results_left* e *Calib_Results_right*.

Posizioniamoci quindi nella *Command Window* di MATLAB, e avviamo la toolbox, digitando *'stereo_gui'* e dando invio.

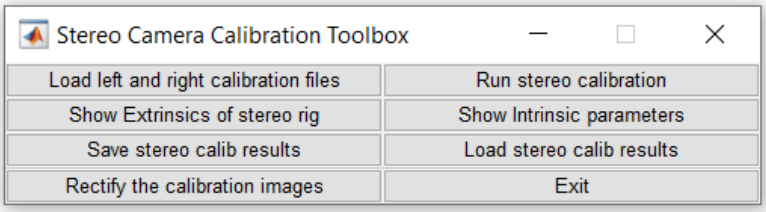


Figura 11 - Finestra principale di 'stereo_gui'

Posizioniamoci nella cartella dove sono presenti le due calibrazioni, e clicchiamo su *Load left and right calibration files*.

Immettiamo quindi i nomi dei due file '.mat'.

```
>> stereo_gui

Calib_Results_left.mat    Calib_Results_right.mat   Calib_Results_stereo.mat

Loading of the individual left and right camera calibration files
Name of the left camera calibration file ([]=Calib_Results_left.mat): C:\Users\USER\Desktop\tirocinio\16-12-20\CALIB\calib_stereo\Calib_Results_left.mat
Name of the right camera calibration file ([]=Calib_Results_right.mat): C:\Users\USER\Desktop\tirocinio\16-12-20\CALIB\calib_stereo\Calib_Results_right.mat
Loading the left camera calibration result file C:\Users\USER\Desktop\tirocinio\16-12-20\CALIB\calib_stereo\Calib_Results_left.mat...
Loading the right camera calibration result file C:\Users\USER\Desktop\tirocinio\16-12-20\CALIB\calib_stereo\Calib_Results_right.mat...


Stereo calibration parameters after loading the individual calibration files:


Intrinsic parameters of left camera:

Focal Length:          fc_left = [ 1661.27455   1660.42283 ] ± [  32.48776    32.41195 ]
Principal point:       cc_left = [ 655.67743   519.84498 ] ± [ 12.28215   13.70573 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ -0.18061   0.43396   -0.00019   -0.00148  0.00000 ] ± [ 0.03626   0.19845   0.00119   0.00117  0.00000 ]



Intrinsic parameters of right camera:

Focal Length:          fc_right = [ 1999.78847   1988.66603 ] ± [ 78.01235   75.42329 ]
Principal point:       cc_right = [ 632.36103   629.93012 ] ± [ 8.79037   28.67350 ]
Skew:             alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_right = [ -0.25605   0.90599   -0.00071   -0.00091  0.00000 ] ± [ 0.02845   0.25042   0.00256   0.00097  0.00000 ]


Extrinsic parameters (position of right camera wrt left camera):

Rotation vector:             om = [ 0.25718   -0.01544  0.01956 ]
Translation vector:           T = [ 23.72476   339.68602  334.31486 ]
```

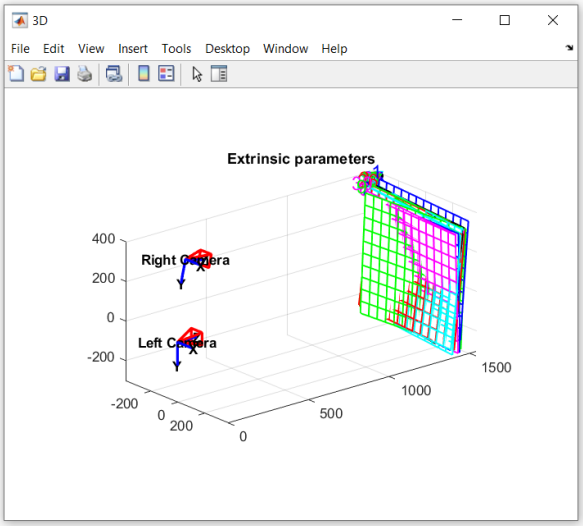
Vengono mostrati quindi i parametri intrinseci rispettivi alle due fotocamere e una stima dei parametri estrinseci relativi alla posizione reciproca di queste ultime.

Avviamo ora la stereo-calibrazione, cliccando su *Run stereo calibration* sulla finestra principale della toolbox.

```
Recomputation of the intrinsic parameters of the left camera (recompute_intrinsic_left = 1)

Recomputation of the intrinsic parameters of the right camera (recompute_intrinsic_right = 1)

Main stereo calibration optimization procedure - Number of pairs of images: 20
Gradient descent iterations: 1...2...3...4...5...6...7...8...9...10...11...12...13...14...15...16...17...done
Estimation of uncertainties...done



Intrinsic parameters of left camera:

Focal Length:          fc_left = [ 1916.18533   1915.35535 ] ± [ 68.47231   68.26382 ]
Principal point:       cc_left = [ 637.72947   501.20931 ] ± [ 9.82596   11.36285 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ -0.23665   0.74834   0.00151   0.00012  0.00000 ] ± [ 0.03251   0.26208   0.00119   0.00116  0.00000 ]


Intrinsic parameters of right camera:

Focal Length:          fc_right = [ 1853.03734   1844.98313 ] ± [ 59.48399   57.77483 ]
Principal point:       cc_right = [ 638.89468   590.97689 ] ± [ 8.37380   21.81723 ]
Skew:             alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_right = [ -0.22124   0.70597   0.00243   -0.00099  0.00000 ] ± [ 0.02884   0.22904   0.00179   0.00104  0.00000 ]


Extrinsic parameters (position of right camera wrt left camera):

Rotation vector:             om = [ 0.25209   -0.02725  0.01917 ] ± [ 0.00884   0.00582  0.00084 ]
Translation vector:           T = [ 25.00873   399.12455  24.53464 ] ± [ 5.67385   9.78967  34.06550 ]


Note: The numerical errors are approximately three times the standard deviations (for reference).
```

Il programma ha ricomputato quindi i parametri intrinseci ed estrinseci, ed i relativi errori.

Possiamo ora visualizzare la posizione delle telecamere nello spazio e dei piani di calibrazione, cliccando su *Show Extrinsics of stereo rig*.



Salviamo ora tutti i dati ottenuti, cliccando su *Save stereo calib results*; i risultati sono ora salvati nel file *Calib_Results_stereo.mat*.

La fase di calibrazione è ora terminata, quest'ultimo file è quello che useremo nel codice, per effettuare il passaggio di coordinate, da 2D a 3D, da pixel a millimetri.

3.4 Estrazione delle coordinate 2D

Come già detto in precedenza, i punti del piano analizzati, sono gli spigoli dei 9 marker fissati sul modello. La fase di estrazione delle coordinate 2D di tali punti è tanto importante quanto quella di calibrazione. Una precisione non adeguata porterebbe infatti ad errori importanti durante la stereo-triangolazione.

Per ogni immagine, alla fine di questa fase, avremo quindi una matrice 36x2 dove saranno localizzate le coordinate in pixel dei punti.

In un primo momento si è pensato di utilizzare una funzione specifica di MATLAB, '*Region Props*'. Questa funzione richiede di trasformare l'immagine a colori, in un'immagine a scala di grigi; il programma poi, in base ai contrasti tra i grigi, visualizza e contorna i vari oggetti che compongono l'immagine. Sarebbe quindi possibile, in questo modo, avere la precisione adeguata per eseguire questa operazione.

Il problema con l'utilizzo di questa funzione sono proprio le immagini in esame. Esse soffrono di svariati difetti, che non permettono di visualizzare i marker come unici oggetti, e di delinearli perfettamente. La superficie in vetro del modello crea riflessi nell'immagine, anche in prossimità degli spigoli; i sacchi di calce utilizzati

come carichi inombrano i marker, rendendo la luminosità di questi ultimi non uniforme.

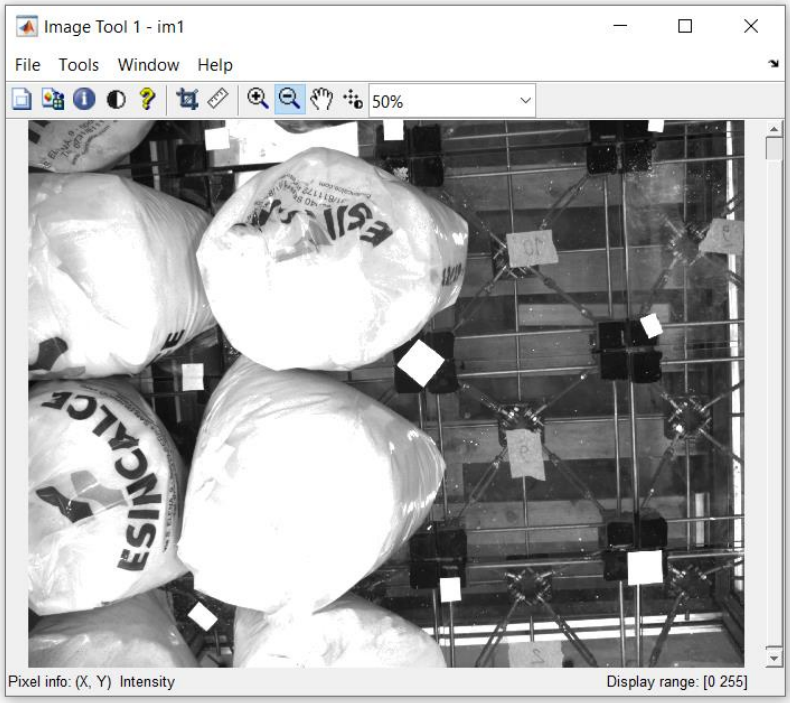


Figura 12 - immagine del piano, con riflessi ed ombre

Nella figura 12 si può chiaramente notare quanto il riflesso del vetro interferisca con la corretta visualizzazione degli spigoli del marker in alto a sinistra. I sacchi di calce, inoltre, adombrano leggermente il marker di sinistra, nella riga centrale.

Queste due importanti interferenze impediscono l'utilizzo della funzione sopra descritta, che non può infatti essere adattata a tutte le immagini che dobbiamo analizzare.

Si è deciso quindi di prelevare le coordinate in modo manuale, con la seguente procedura:

- ***Caricamento dell'immagine*** con la seguente operazione

Im0 = imread('nome_immagine.bmp');

- ***Trasformazione dell'immagine in bianco e nero*** tramite

Im0_bw = im2bw(im0,0.5);

dove ***Im0*** è l'immagine precedentemente caricata, e ***0.5*** è il coefficiente di illuminazione discriminante per il passaggio di ogni pixel, da 'colorato' a 'bianco/nero' (tale coefficiente è arbitrario, dipende dalla luminosità del marker e dello sfondo).

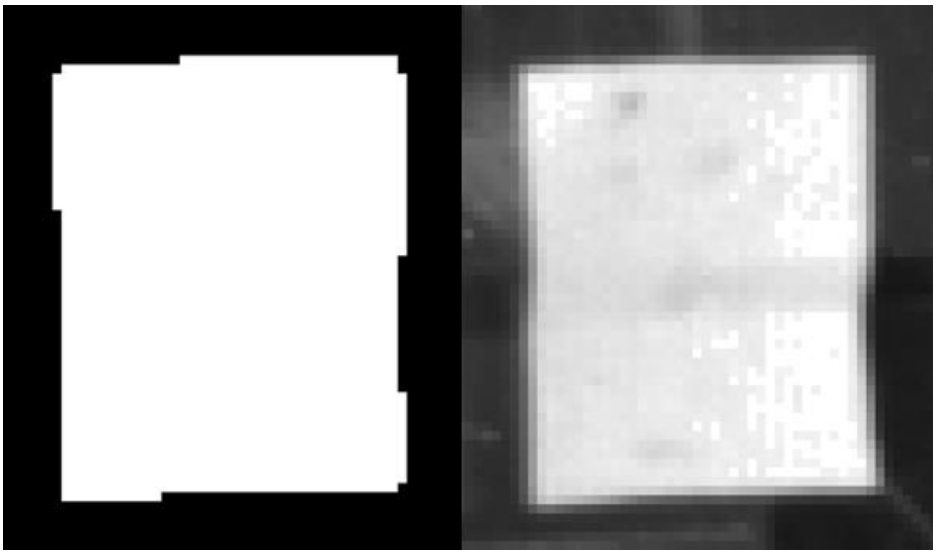


Figura 13 - marker bianco/nero e marker originale

È evidente che senza questo passaggio sarebbe stato più difficile individuare il pixel esatto che identifica lo spigolo del marker, avendo un contorno sfumato e non netto.

- ***Ricerca degli spigoli dei marker***

Selezioniamo l'immagine in bianco e nero sulla *Workspace*, e apriamo '*imtool*' dalle applicazioni di plottaggio. A questo punto cerchiamo gli spigoli dei marker e

posizioniamoci sopra col puntatore (cercando di essere il più precisi e coerenti possibile), ora cliccando il tasto destro del mouse, e poi su *copia*, abbiamo copiato le coordinate del pixel, e possiamo così costruirci la nostra matrice 36x2 nell'editor di testo.

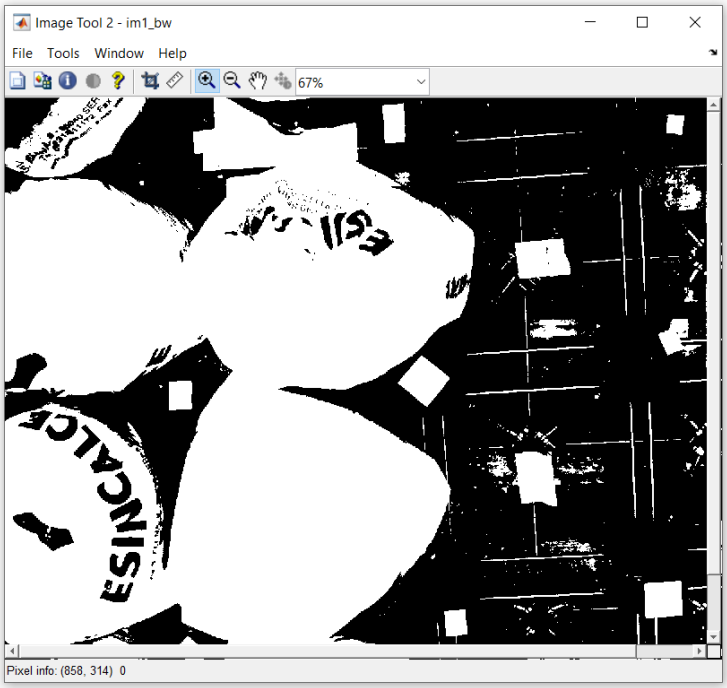


Figura 14 - imtool(im1_bw)

Alcuni spigoli non saranno definibili, in base alle stesse interferenze sopra descritte. In questo caso possiamo provare a plottare l'immagine non modificata, e da questa cercare di prendere le coordinate di tale spigolo. Se anche questa operazione non fosse possibile, possiamo non considerare lo spigolo, ricordandoci di farlo anche per la seconda fotocamera, così da avere due matrici di uguale dimensione. Nel caso in cui le due matrici avessero dimensioni differenti, infatti, non sarebbe possibile la stereo-triangolazione.

Una volta presi tutti i punti dell'immagine salviamo la matrice come file '.mat', nominandola, ad esempio, '*punti2D_nomeprova_cam0*'.

La matrice relativa alla seconda fotocamera (nella stessa prova) la chiameremo '*punti2D_nomeprova_cam1*'; ora salviamo le due matrici in un unico file '.mat', che chiameremo '*punti2D_nomeprova*'.

Quest'ultima è la fase più lunga dell'intero processo, ed è quella che richiede maggiore precisione, per evitare di ottenere, alla fine, delle frecce non coerenti con il carico statico applicato.

Abbiamo ora tutto ciò che ci serve per andare a scrivere ed eseguire il codice del passaggio di coordinate.

3.5 Codice di passaggio di coordinate

Il software contiene, come già accennato, la funzione di stereo-triangolazione; andremo quindi ad utilizzare tale funzione, adattando le nostre variabili a quelle richieste come input.

Nello specifico, la funzione richiede:

- **xL**, matrice Nx2 delle coordinate in pixel dell'immagine di sinistra;
- **xR**, matrice Nx2 delle coordinate in pixel dell'immagine di destra;
- **om**, **T**, vettore di rotazione e di traslazione tra la fotocamera di destra e sinistra;
- parametri intrinseci della fotocamera di destra e sinistra.

Gli ultimi due punti sono i risultati della stereo-calibrazione.

Come output avremo:

- **XL**: martice 3xN delle coordinate 3D dei punti rispetto alla fotocamera di sinistra;
- **XR**: matrice 3XN delle coordinate 3D dei punti rispetto alla fotocamera di destra.

Una volta avvenuta la stereo-triangolazione, riportiamo le coordinate **XL** (o **XR**) al sistema di riferimento relativo al piano di calibrazione, così da avere tutti i punti sul piano in fase di scarico, e gli spostamenti solamente lungo l'asse Z (perpendicolare al piano) in fase di carico.

Per questo passaggio utilizziamo le matrici di rotazione e traslazione della fotocamera di sinistra (o di destra), facendone l'inversa.

Il codice che ne risulta è il seguente (esempio di prova con 200 kg centrati):

```
clc;
clear;
close all;

load punti2D_200c_c

load Calib_Results_stereo

calib0.KK=KK_left;
calib0.alpha=alpha_c_left;
calib0.R=rodrigues(omc_left_1);
calib0.omc=omc_left_1;
calib0.T=Tc_left_1;
RT1=[rodrigues(omc_left_1) Tc_left_1];
RTc1=[RT1; 0 0 0 1];
calib0.RT4i=inv(RTc1);
calib0.H=KK_left*RT1;
calib0.kc=kc_left;
calib0.fc=fc_left;
calib0.cc=cc_left;
```

```
calib1.KK=KK_right;
calib1.alpha=alpha_c_right;
calib1.kc=kc_right;
calib1.fc=fc_right;
calib1.cc=cc_right;
calib1.om12=om;
calib1.T12=T;


figure
plot(punti2D_200c_c_cam0(:,1),punti2D_200c_c_cam0(:,2),'or')
title('camera1')

figure
plot(punti2D_200c_c_cam1(:,1),punti2D_200c_c_cam1(:,2),'ob')
title('camera2')



[XL,XR,xl,xr] = stereo_triangolazione(punti2D_200c_c_cam0',punti2D_200c_c_cam1',calib0,calib1);
XW2 = calib0.RT4i(1:3,:)*[XL;ones(1,size(punti2D_200c_c_cam0,1))];

figure
plot3(XW2(1,:)',XW2(2,:)',XW2(3,:)','o');
axis equal
title('punti3D')
```

dove:

- rig. 1-3 – pulizia della *Workspace*, della *Command Window*, e chiusura di tutte le applicazioni del software;
- rig. 5 – caricamento dei punti 2D (relativi a tutte due le fotocamere);
- rig. 7 – caricamento dei risultati della stereo-calibrazione;
- rig. 9-10, 18-28 – assegnazione delle variabili, nominate secondo quanto prevede la stereo-triangolazione;
- rig. 11-17 – preparazione della matrice per il cambio di sistema di riferimento, dalla fotocamera di sinistra al piano di calibrazione (quindi del modello);
- rig. 31-33 – plottaggio dei punti 2D rispetto alla fotocamera di sinistra;
- rig. 35-37 – plottaggio dei punti 2D rispetto alla fotocamera di destra;
- rig. 41 – stereo-triangolazione;
- rig. 42 – passaggio del sistema di riferimento dalla fotocamera di sinistra al piano di calibrazione;
- rig. 44-47 – plottaggio dei punti 3D nel sistema di riferimento del piano di calibrazione.

4. Risultati

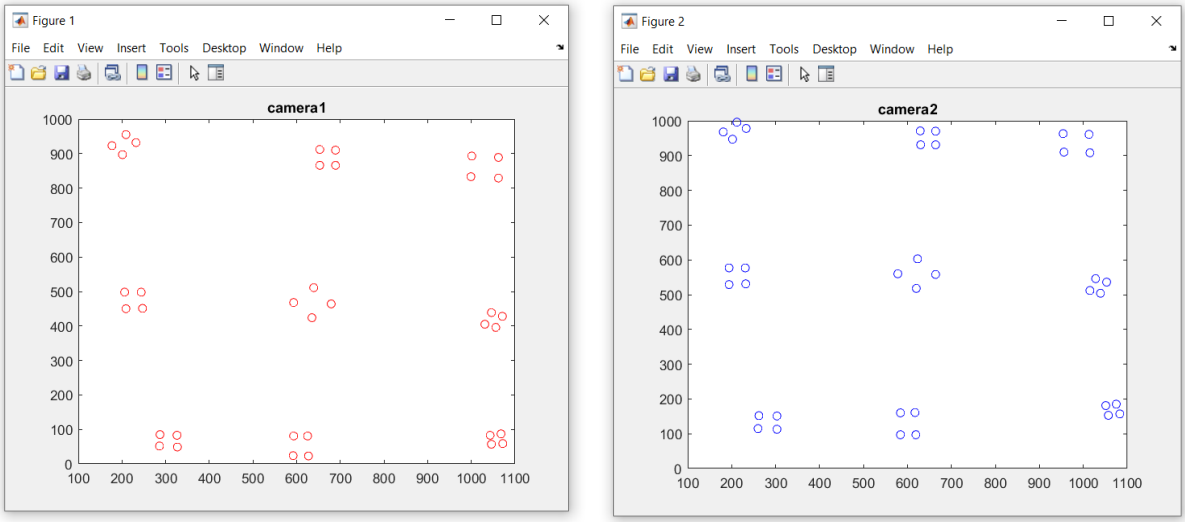
Vediamo quali sono i risultati dell'intero processo, in particolare le immagini 2D e 3D prodotte dal software.

Tali risultati, a primo impatto, risultano essenzialmente identici tra loro, tuttavia le differenze sono presenti, ma in scala molto piccola rispetto alla dimensione del piano (1-2% rispetto alla larghezza del modello). Queste differenze, non apprezzabili ad occhio, possono essere riscontrate nelle matrici 3xN.

Come già accennato, per ogni disposizione di carico sono state effettuare 3 prove a modello scarico, e poi 3 prove a modello carico, per ridurre al minimo la possibilità di trovare, nell'analisi, errori troppo grandi, che annullerebbero la riuscita del processo, e obbligherebbero a rieseguire la prova da capo.

Verranno mostrati i risultati di una sola prova per ogni disposizione di carico.

- ***Piano indeformato***



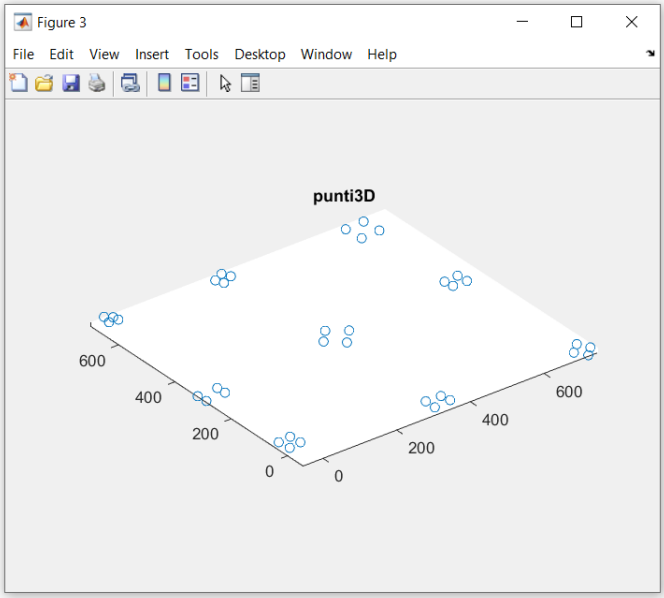


Figura 15 – Punti 2D e 3D piano indeformato

- ***Piano carico con 200 kg laterali***

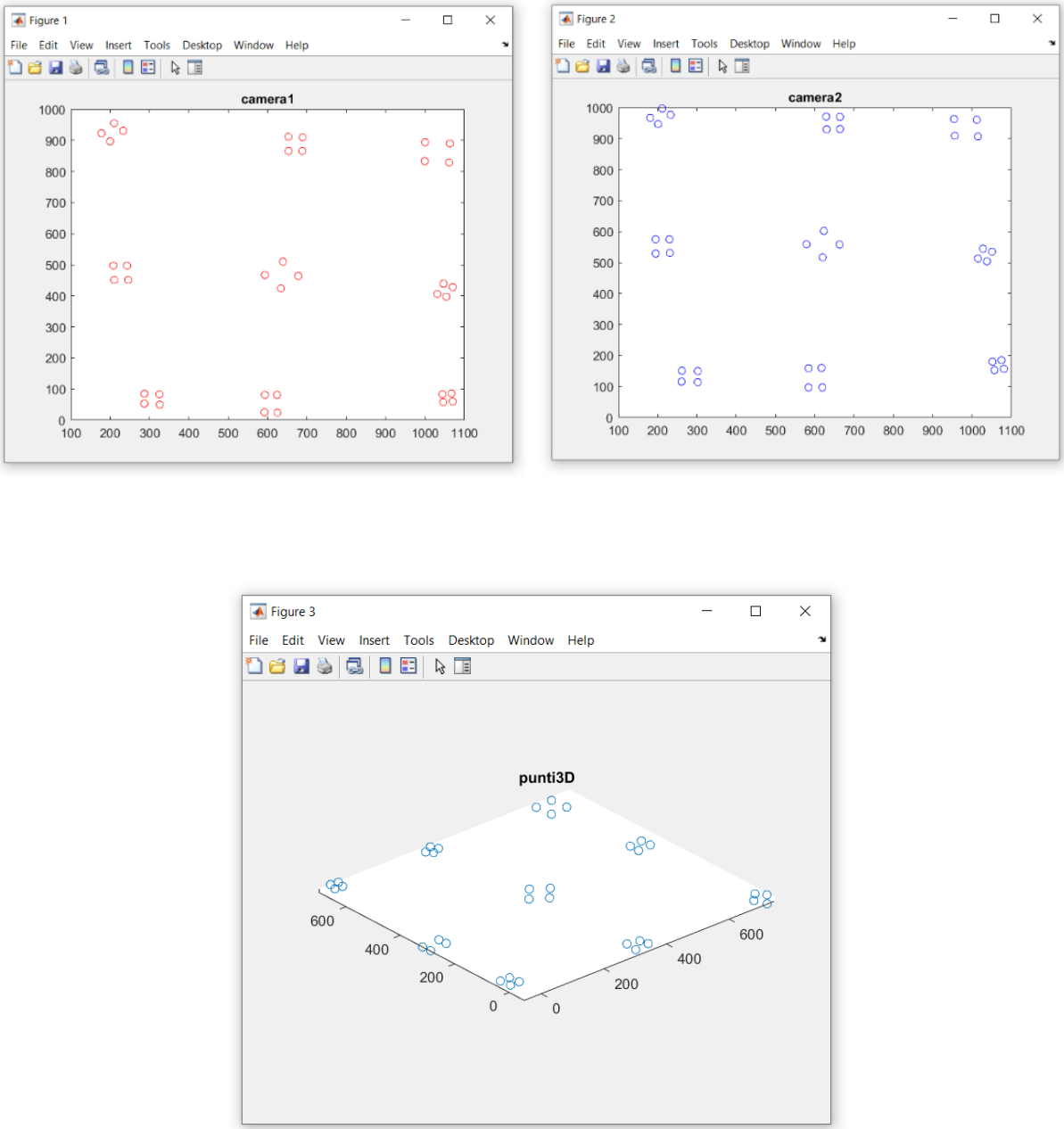


Figura 16 - Punti 2D e 3D con 200 kg laterali

- ***Piano carico con 200 kg centrali***

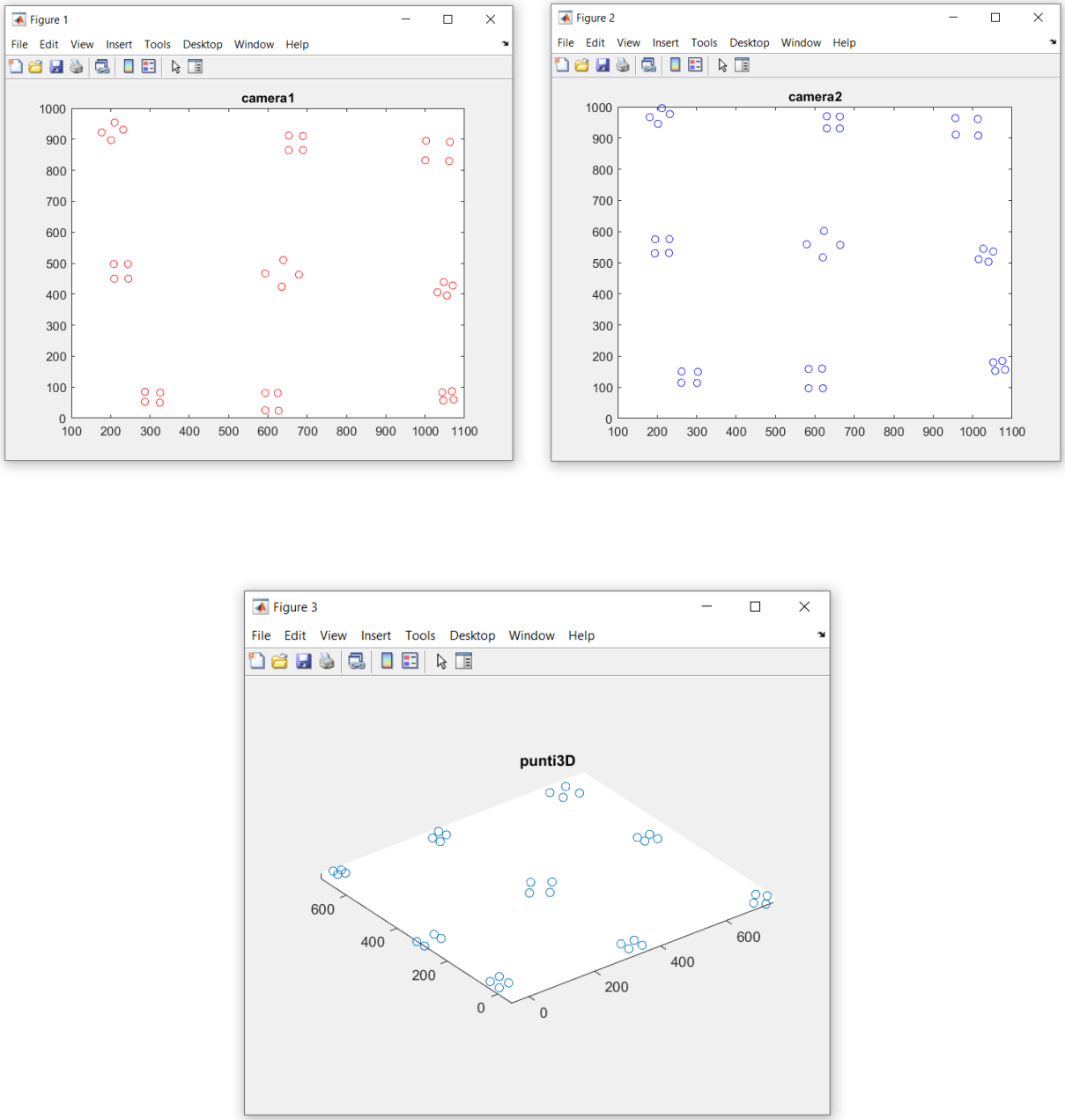


Figura 17 - Punti 2D e 3D con 200 kg centrali

- ***Piano carico con 400 kg uniformemente distribuiti***

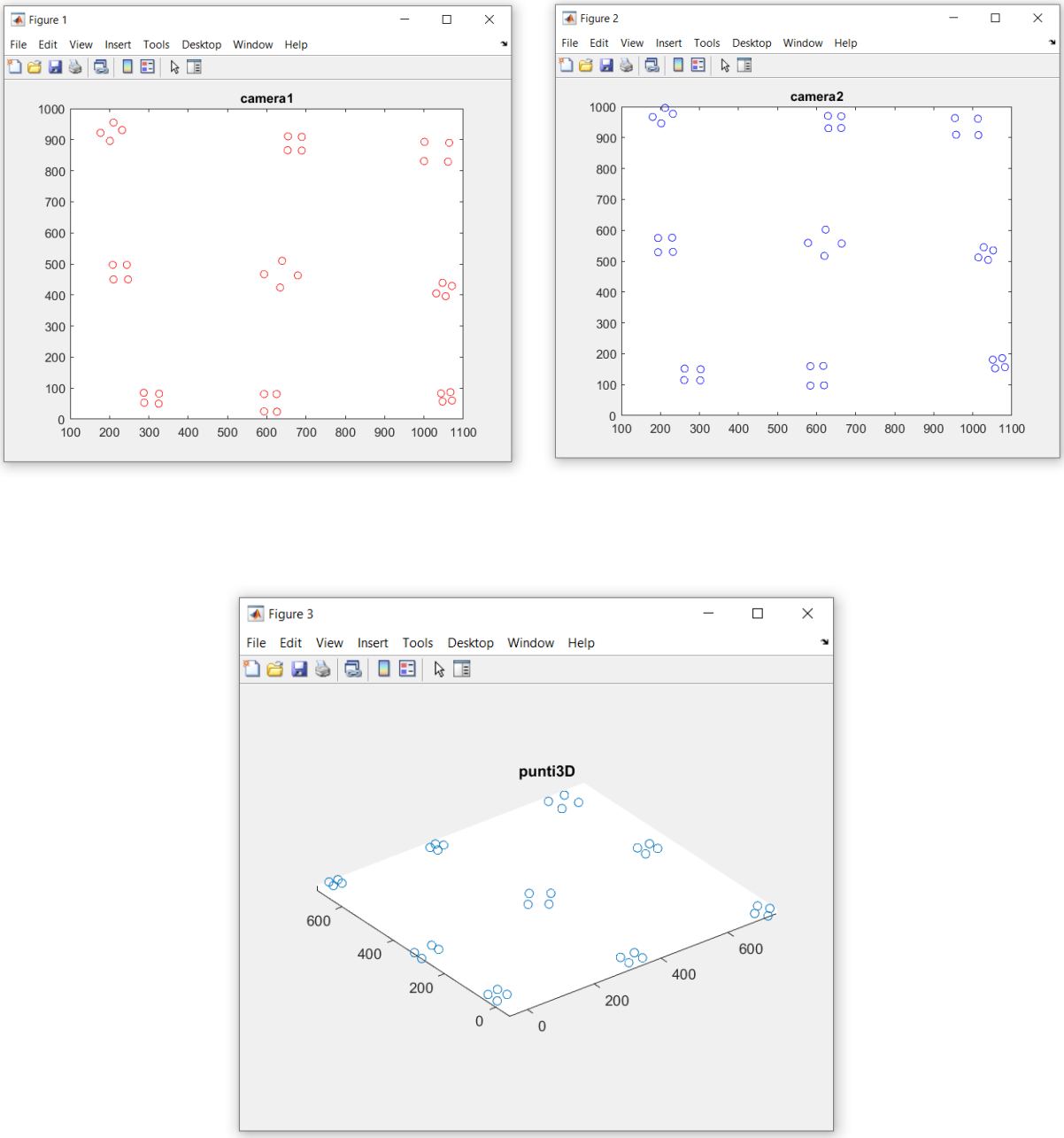


Figura 18 - Punti 2D e 3D con 400 kg uniformemente distribuiti

5. Conclusioni

Il presente trattato di tesi è finalizzato allo studio del comportamento di un modello di piano in vetro ad elementi tesi.

È stata spiegata quindi l'intera procedura di analisi adottata per raggiungere tale scopo.

Questa analisi è stata basata sul concetto di stereoscopia, che ha permesso di calcolare opportunamente dimensioni e distanze, partendo solamente da delle immagini prospettiche dello stesso modello.

Utilizzando il software MATLAB è stato possibile calibrare le fotocamere, ricavarne i parametri intrinseci ed estrinseci, e procedere poi con la stereo-triangolazione di alcuni punti del piano, fulcro del lavoro svolto.

I risultati esposti sono stati poi rielaborati, sovrapponendo le immagini 3D del piano carico (nelle diverse disposizioni) e scarico, così da poter confrontare le coordinate dei punti lungo l'asse perpendicolare al piano, e trovare la freccia che quest'ultimo subisce sotto l'effetto delle forze statiche.

Queste sovrapposizioni non coincidono però con i risultati che potevamo aspettarci: le frecce ottenute, infatti, non sono compatibili con i carichi, e sembra che il lavoro svolto sia fondamentalmente sbagliato.

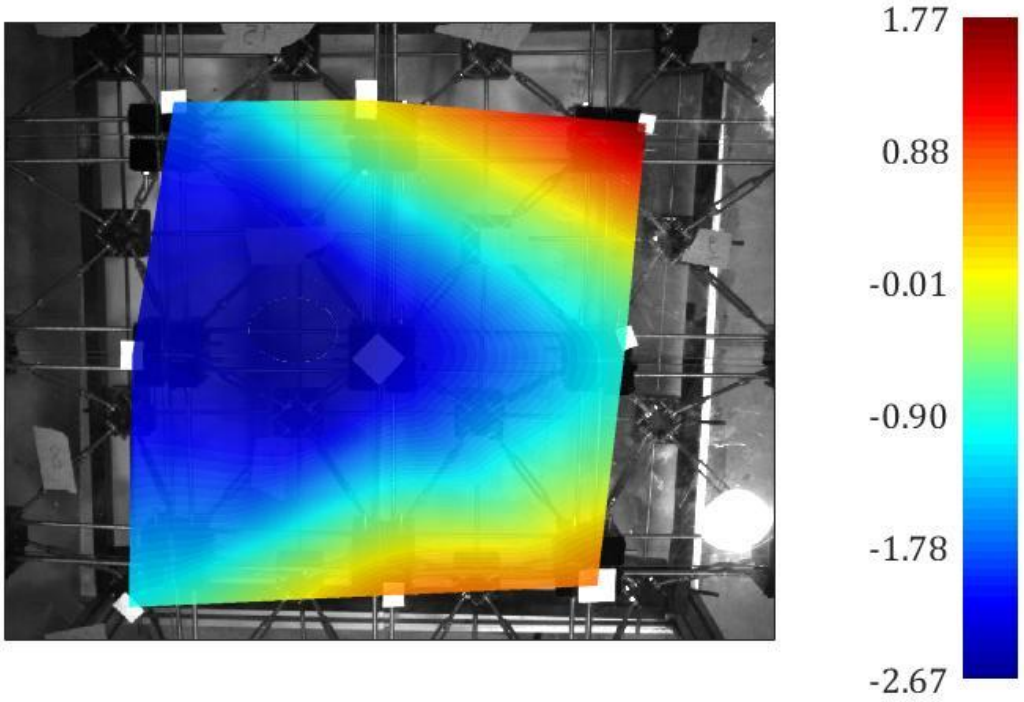


Figura 19 - Sovrapposizione tra piano indeformato e piano carico con 200 chili laterali

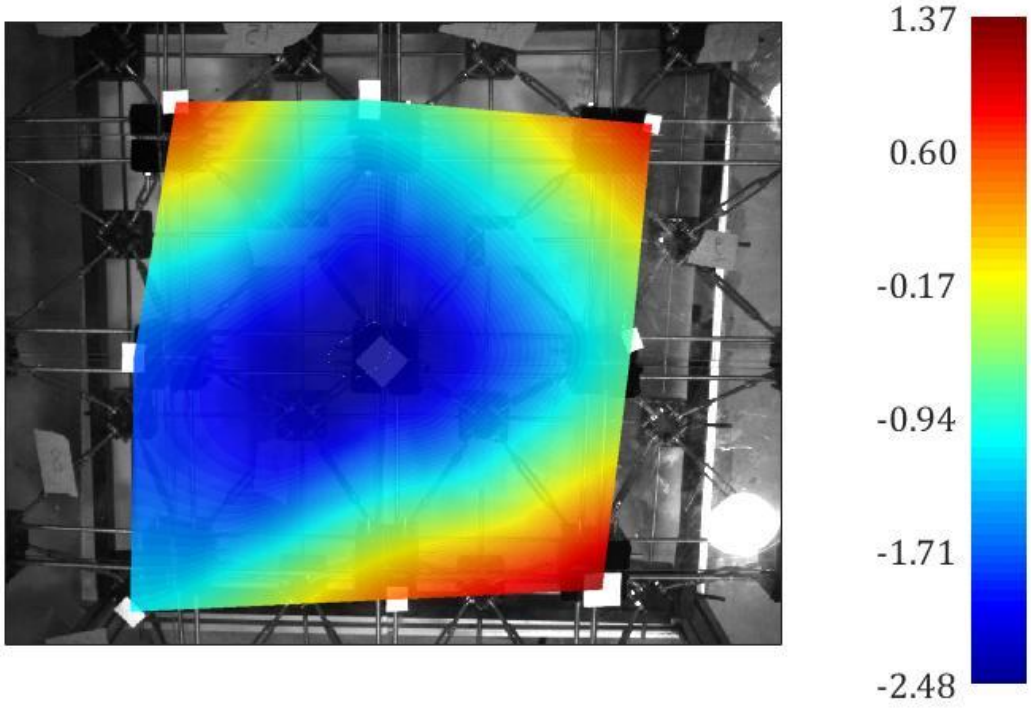


Figura 20 - Sovrapposizione tra piano indeformato e piano carico con 200 chili centrali

Nelle figure 19 e 20 le misure si riferiscono alla freccia subita dal modello, e sono espresse in *mm*.

A primo impatto è evidente che tali frecce non coincidono con quello che ci si poteva aspettare.

Rianalizzando il lavoro svolto è stato possibile comprendere la fonte di ciò che ha portato, infine, a dei risultati incomprensibili. La causa di ciò è attribuibile al piano di calibrazione non particolarmente adatto (in quanto il software non riconosce bene la griglia), al quale hanno seguito delle calibrazioni non precise.

Il resto del lavoro rimane sufficientemente preciso, ed il codice di passaggio delle coordinate da 2D a 3D non ha bisogno di correzioni.

È sufficiente quindi cambiare il piano di calibrazione e rieseguire le calibrazioni per ottenere dei risultati affidabili e coerenti.

6. Bibliografia

- http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html
- http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html
- http://rilievo.stereofot.it/studenti/tesi/diLauro/corso/tesi/fotogrammetria.html#parametri
- http://www.ce.unipr.it/~medici/geometry/node135.html