Project 1
Camera calibration

Kevin Marzio

16 febbraio 2022

1 Problem Statement

The 3 main goals of this project are:

- calibrate camera using Zhang procedure by finding its intrinsic parameter and, for each image, its pose respect the world reference frame;
- refine the results obtained from the calibration procedure by taking into account radial distortion;
- superimpose an object in the images employed for the calibration.

To do so we have available a set of 20 images of a checkerboard, each taken at different angles and distances.
In the following sections we are going to describe the approach used to solve the problems along with the choices made during the implementation. Finally we dedicate section 6 on page 9 for results and observations.
All the steps have been implemented using Matlab in a script called *Kevin_ Marzio_ project_ function.m.*

2 Calibration

In this section we describe the theory that allowed us to solve the first problems mentioned in the previous section. All we need to find the intrinsic and extrinsic parameters are the projection matrices $P = K\,[R\,|\,t\,]$. We will get one for each image, but K, the intrinsic parameters matrix, will be in common between all the images. The ones that change between different images will be R and t, respectively the rotation matrix and the translation vector representing the orientation and the position of the camera respect the world reference frame.
The calibration approach used is the one based on the Zhang procedure [5].
The perspective projection model presented during the lectures, with a slight modification on matrix K, has been used throughout the work. The K matrix used is [1]:

$$K = \begin{bmatrix} f\,k_u & f\,k_\vartheta & u_0 \\ 0 & f\,k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & \alpha_\vartheta & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where:

[1] Thus, K has the same form used in [1]

- f is the focal length of the camera;
- k_u and k_v are the reciprocal of the pixel sizes;
- k_ϑ is the skewedness (diagonal distortion) of the image plane (which is usually negligible or zero);
- (u_0, v_0) is the location of the image center (that is the optical axis) with respect to the image coordinate system (expressed in pixel).

At first, in chapter 2.1, we will explain how to estimate K, then in 2.2 on page 4 we will focus in how to obtain P (thus finding also R and t). In the last chapter 2.3 on page 4, we will see in detail how to estimate an homography H (required also to estimate K). All the following chapter are in turn divided into explanation and implementation.

2.1 Estimate K

2.1.1 Approach description

The map from a 3D plane to its perspective image is seen to be a homography, in fact, if we assume to know the coordinates of a set of point that belongs to a plane, we can write:

$$\omega \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = [p1\, p2\, p3\, p4] \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \underbrace{[p1\, p2\, p4]}_{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

where:

- x, y and 0 are the coordinate of a real world point of a plane, assuming $z = 0$ (in our case the plane is the checkerboard's plane);
- u and v are the coordinates of its projection in the image;
- ω is a scale factor;
- p_i $(i = 1, 2, 3, 4)$ are the columns of the projection matrix;
- H is the homography matrix, notice that it is defined up to a scale factor. If we have n planes available we will obtain n different Hs, one for each different plane. We will use in the following these homography matrices to obtain K.

If we assume to know $H = \lambda\, [p1\, p2\, p4]$ and we notice that $P = K\, [R \,|\, t\,] = K\, [r_1\, r_2\, r_3\, t]$, we get $p_1 = K r_1$, $p_2 = K r_2$, $p_4 = K t$ and by substitution:

$$H = [h_1\, h_2\, h_3] = \lambda\, K\, [r_1\, r_2\, t] \tag{2}$$

Equating the columns we obtain:

$$\begin{cases} r_1 = \frac{1}{\lambda} K^{-1} h_1 \\ r_2 = \frac{1}{\lambda} K^{-1} h_2 \end{cases} \tag{3}$$

and exploiting the properties of orthogonality of the rotation matrix, we can write:

$$\begin{cases} h_1^\intercal \, (KK^\intercal)^{-1} \, h_2 = 0 \\ h_1^\intercal \, (KK^\intercal)^{-1} \, h_1 = h_2^\intercal \, (KK^\intercal)^{-1} \, h_2 \end{cases}$$

that can be rewritten as:

$$\begin{cases} h_1^\intercal \, B \, h_2 = 0 \\ h_1^\intercal \, B \, h_1 = h_2^\intercal \, B \, h_2 \end{cases} \tag{4}$$

where $B = (KK^\intercal)^{-1}$. By taking n planes with the corresponding homography we therefore obtain $2n$ linear constraints on the same B. What we have to do now is to solve for B and, once this is done, we will recover K using $B = (KK^\intercal)^{-1}$.
To estimate B let's notice first that B is symmetric by construction, thus we can write for $i, j \in \{1, 2\}$:

$$h_i^\intercal \, B \, h_j = v_{ij}^\intercal \, b \tag{5}$$

where

$$v_{ij} = \begin{bmatrix} H_{1i}H_{1j} \\ H_{1i}H_{2j} + H_{2i}H_{1j} \\ H_{2i}H_{2j} \\ H_{3i}H_{1j} + H_{1i}H_{3j} \\ H_{3i}H_{2j} + H_{2i}H_{3j} \\ H_{3i}H_{3j} \end{bmatrix}$$

and

$$b = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix}^\intercal$$

denoting with H_{xy} and B_{xy} the xy-th element of the corresponding matrices. Finally combining (4) and (5) we obtain:

$$\begin{bmatrix} v_{12}^\intercal \\ (v_{11} - v_{22})^\intercal \end{bmatrix} b = 0$$

and for n planes, by stacking equation of the above form, we get:

$$Vb = 0$$

where V is $2n \times 6$.
To solve the previous system at least $n = 3$ planes are required so that to have $rank(V) = 5$ and be able to find b, determined up to a scale factor. However, from practical measurement, we will obtain a full-column rank V. A solution can still be found using linear least square:

$$\underset{\|b\|=1}{\arg\min} \|Vb\|^2$$

which solution is the right singular vector of V corresponding to the smallest singular value (if the singular value decomposition of V is $V = U\Sigma U^\intercal$ then the solution is $b = s_6$ i.e. the rightmost column of S). The solution thus founded allows us to obtain the entries of the sought B.
Once we have found B we can compare its definition $B = (KK^\intercal)^{-1}$, and its unique Cholesky factorization, defined as $B = LL^\intercal$, to obtain the desired K:

$$K = (L^\intercal)^{-1}$$

2.1.2 Approach description

In practice we have implemented all the steps explained above in the function *estimateIntrinsicStd()*, notice that in the code we have:

- imposed B to be positive definite (to be able to compute its Cholesky factorization);
- normalized the K matrix dividing its elements by $K(3,3)$ (thus imposing $K(3,3) = 1$). This is done because the estimated K is affected by the arbitrary scale factor of P.

2.2 Estimate P

2.2.1 Approach description

Once the matrix K is known, what remain to be found are the rotational matrix R and the translational vector t, to reconstruct the projection matrix P. This can be easily done by recalling (2) and imposing the orthogonality of R:

$$r_1 = \frac{1}{\lambda}K^{-1}h_1, \qquad r_2 = \frac{1}{\lambda}K^{-1}h_2, \qquad t = \frac{1}{\lambda}K^{-1}h_3, \qquad r_3 = r_1 \times r_2$$

where $\lambda = \|K^{-1}h_1\| = \|K^{-1}h_2\|$ and $\times$ is the cross product. Due to noise the obtained R may not be orthogonal. This problem can be tackled by solving an Orthogonal Procrustes problem of the form

$$\underset{W^\intercal W=I}{\arg\min}\|R - W\|_F^2$$

which solution $W^* = UV^\intercal$ is the closest orthogonal matrix to R (in Frobenius norm), where $R = U\Sigma V^\intercal$ is the singular value decomposition of R and I is the identity matrix.

2.2.2 Implementation

This procedure has been implemented in the function *estimateExtrinsicFromHK()* that makes the previous computation given K and the homography H related to a single image.
Due to noise it may happen that $\lambda_1 = \|K^{-1}h_1\|$ and $\lambda_2 = \|K^{-1}h_2\|$ are not perfectly equal, so we've decided to take as value of λ their mean $\frac{\lambda_1+\lambda_2}{2}$.
We've also implemented some checks to be sure that the estimated P is a valid projection matrix by checking that R is actually a rotation and not a reflection and that it respect the Faugeras' theorem[2].

2.3 Estimate homographies

2.3.1 Approach description

If we consider a plane having $z = 0$, the perspective projection (2) and we partition H row-wise as

$$H = \begin{bmatrix} h_1^\intercal \\ h_2^\intercal \\ h_3^\intercal \end{bmatrix}$$

[2]See L2, page 43

we obtain:

$$\omega \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{m} \implies \begin{cases} h_1^\intercal m - u h_3^\intercal m = 0 \\ h_2^\intercal m - v h_3^\intercal m = 0 \end{cases}$$

and by transposing the two equation we can get, in matrix form:

$$\begin{bmatrix} m^\intercal & 0^\intercal & -um^\intercal \\ 0^\intercal & m^\intercal & -vm^\intercal \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0$$

Thus we obtain a system of type $Ah = 0$, with 2 equations for a single correspondence of points. By considering N correspondences, A will be of size $2N \times 9$ and rank 9 due to measurement noise (if $N > 4$).
A least square solution can be found, if $N \geq 8$, by using singular value decomposition in order to obtain the desired homography matrix.

2.3.2 Implementation

The method above is implemented in the function *estimateH*, taking as input a set of real world points belonging to a plane and their respective projections in the image, it compute H.
We've also forced that the estimated homography is positive definite, otherwise, when we compute the extrinsic parameters of the camera, we would obtain a relative pose of the camera that is behind the plane, instead of being in front.
In our case the point correspondences are obtained by considering as plane the checkerboard plane in the image and:

a. detecting the projection (u, v) of the corners of the squares on the board by using the *detectCheckerboardPoints()* function provided in Matlab;

b. finding the coordinates of the corners of the squares assuming that the world reference frame is located in the top-left part of the checkerboard and the size of the squares are known.

3 Radial distortion and its compensation

Due to the fact that real cameras are built with lenses it turns out that for more accurate camera modeling, radial distortion has to be taken into account. While lens distortion is a complex physical phenomenon in general, it is usually modeled by the following 2-parameter model:

$$\begin{cases} \hat{u} = (u - u_0)(1 + k_1 r_d^2 + k_2 r_d^4) + u_0 \\ \hat{v} = (v - v_0)(1 + k_1 r_d^2 + k_2 r_d^4) + v_0 \end{cases} \tag{6}$$

where u and v are the ideal projections (in absence of radial distortion), $\hat{u}$ and $\hat{v}$ are the actual projections, k_1 and k_2 are the parameters of the model and

$$r_d^2 = \left(\frac{u - u_0}{\alpha_u} \right)^2 + \left(\frac{v - v_0}{\alpha_v} \right)^2$$

Once the model is known, the main goal is to find a perspective projection matrix P that is less affected by the radial distortion.
Given the correspondences

$$m_{(i)} = \begin{bmatrix} x_{(i)} \\ y_{(i)} \\ z_{(i)} \\ 1 \end{bmatrix} \quad \longleftrightarrow \quad m'_{(i)} = \begin{bmatrix} u_{(i)} \\ v_{(i)} \\ 1 \end{bmatrix}, \qquad i = 1, \dots n$$

the steps that we have to perform in order to obtain the desired P are:

a. estimate P from the previous correspondences;

b. estimate k_1 and k_2;

c. compensate for radial distortion and get new $m'_{(i)}$, $i = 1, \dots n$;

d. go to step 1 until convergence of P and k_1, k_2

In section 2 on page 1 we've already discussed point a, in the rest of this section, we will discuss about points b and c, chapter 3.1 and 3.2 respectively, and their implementation, chapter 3.3 on the next page.

3.1 Estimating k_1 and k_2

The effects of the radial distortion are modeled by the 2 parameter k_1 and k_2, to estimate them, we can start by rewriting equation (6) as:

$$\begin{cases} \hat{u} - u = (u - u_0) r_d^2 k_1 + (u - u_0) r_d^4 k_2 \\ \hat{v} - v = (v - v_0) r_d^2 k_1 + (v - v_0) r_d^4 k_2 \end{cases} \tag{7}$$

If we assume that the ideal P is known, the ideal undistorted coordinates (u, v) can be computed from the 3D coordinates (x, y, z) by applying the projection. We notice also that, thanks to the fact that r_d can be computed directly from P, the system (7) is linear in the unknowns k_1 and k_2, therefore, if we have many correspondences, we can build an overdetermined system of the form:

$$A \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = b$$

which can be solved using least square, obtaining both $k1$ and $k2$.

3.2 Distortion's compensation

Once k_1 and k_2 have been estimated, we can pass to the step c of the procedure, where we're going to estimate the new $m'_{(i)}$, for $i = 1, \dots, n$, that are not affected by the distortion.
To do so we switch to normalized coordinates defined as:

$$x = \frac{u - u_0}{\alpha_u}, \qquad y = \frac{v - v_0}{\alpha_v} \tag{8}$$

that allow us to rewrite (6) as follow:

$$\begin{cases} \hat{x} = x\left(1 + k_1(x^2 + y^2) + k_2(x^4 + 2x^2y^2 + y^4)\right) \\ \hat{y} = y\left(1 + k_1(x^2 + y^2) + k_2(x^4 + 2x^2y^2 + y^4)\right) \end{cases} \tag{9}$$

From the system of equations just obtained we can recover the undistorted coordinates (x, y) from the distorted ones $(\hat{x}, \hat{y})$ that are actually visible in the image. Of course the system is non-linear in the unknowns x and y, so, to solve it, some iterative method is required. Finally, from (x, y) and equations (8), we can get the $m'_{(i)}$ sought, where $i = 1, \ldots, n$.

3.3 Implementation

We' ve implemented the steps reported at the beginning of this section inside a *while loop* that ends when the difference of the mean reprojection error between all the images (see section 4 on the following page for more detail), in 2 subsequent steps, falls below a certain threshold t (meaning that the improvement on the explanation of the radial distortion are not significant). In our code the threshold value has been set to $t = 1$ empirically.
In addition for:

estimating P We've implemented the function *updateZhangCalibration()* that, given the set of correspondences presented at the beginning of a single iteration, implement the complete Zhang procedure as explained in section 2 on page 1 to obtain the new P matrix for each image. Inside it we call the function cited in that section such as *estimateH()* and *estimateExtrinsicFromHK()*;

estimating $\mathbf{k_1}$ and $\mathbf{k_2}$ We've implemented the function *estimateRadDistParams()*. Given all the projection in all the images, obtained with the actual P matrices estimated in the iteration, and the coordinates of all corners detected in all images, it compute the parameters of the radial distortion;

compensate distortion We've implemented the function *compensateRadialDistorsion()*. Inside, it solves the system (9) and returns the new projections $m'_{(i)}$, $i = 1, \ldots, n$, that will be used as starting point at the beginning of the next iteration to estimate the P matrices.

To solve the non-linear system we've decided to use the built-in Matlab function *fsolve()*, with the default options. Notice that *fsolve()* requires also an initial guess of the solution (x_0, y_0). As initial guess, at each iteration, we've passed the projections that we've obtained in the previous cycle of the loop (that are the $m'_{(i)}$, $i = 1, \ldots, n$, used to estimate the P matrices in the actual iteration) assuming that these are not far from the ideal projections.

Once we have found the refined projections we also perform some checks to notify the user in case they do not belongs to the image plane. This might happen in the case that the estimated distortion parameters are far from the real ones (for example in the first iterations) and the previous estimate of the projected point was near the image border.

4 Reprojection error

4.1 Approach description

In general, to evaluate the validity of the estimated intrinsic and extrinsic parameters from the Zhang's calibration procedure, is useful to compute the geometrical residual for each image:

$$\epsilon(P) = \sum_{i=1}^{n} \left(\frac{p_1^\intercal m_{(i)}}{p_3^\intercal m_{(i)}} - u_{(i)} \right)^2 + \left(\frac{p_2^\intercal m_{(i)}}{p_3^\intercal m_{(i)}} - v_{(i)} \right)^2 \tag{10}$$

where $m_{(i)}$ is the vector containing the homogeneous coordinates of a point in the real world, $(u_{(i)}, v_{(i)})$ are the coordinates of its respective projection in the image plane and n is the number of correspondences that we're considering.

4.2 Implementation

To compute the reprojection error we've created 2 function:

- *computeReprojError()* that implements the formula (10) and compute the reprojection error for a single image. The function is also able to take into account the radial distortion compensation when computing the reprojection error. It does this by taking as input also K, the camera intrinsic parameter matrix, k_1, k_2, the parameters of the radial distortion, and substituting the undistorted projections with the distorted one obtained from (6);
- *meanReprErr()* that simply computes the mean reprojection error in a set of N images. Inside it calls *computeReprojError()* N times.

5 Object superimposition

Once we have available the perspective projection matrix of the camera related to a single image, it becomes relatively easy to superimpose a geometrical object to the calibration plane of the image itself. All we need to do are the following steps:

- define the object respect the world reference frame by means of a set of homogeneous coordinates;
- project in the image each point that defines the object in the real world.

We have implemented in practice the above procedure with a cylinder by:

- creating the set of homogeneous coordinates representing its edges;
- projecting in the image its edges and filling with colors its circular surfaces to distinguish between the lower and the upper one, respectively red and green.

6 Observations and results

a. To implement the procedures reported above, many correspondences between real world points and image points are required. As reported in section 2.3.2 on page 5, to detect the corner of the checkerboard, we have used the Matlab built-in function *detectCheckerboardPoints()*. Unfortunately we ran into problems, while using this function with "Image2" and "Image9", because it fails to detect the checkerboard and returns some image coordinates as [NaN, NaN]. To avoid problems with subsequent calculations we have decided to discard this 2 images from the set of the processed images at the beginning. Notice that other solution exists, such as removing the bad-detected points. Some *detectCheckerboardPoints()*'s results are reported in figure 1 on the next page;

b. On figure 2 on page 11 (a) we can see the projection of checkerboard's corners after the first estimate of the P matrix without taking into account the radial distortion effects. As can be noted the projected points are not perfectly superimposed with the detected points in the image. On (b) we can see the results after the radial distortion compensation, as expected, the projections that take into account the radial distortion are almost perfectly superimposed to the detected points. Furthermore if we compare the labeled points in the images, that represent the projection of a corner in absence of radial distortion, we can see that their coordinates are slightly different: this shows the refinement of P during the compensation procedure for radial distortion;

c. The goodness of the radial distortion compensation procedure can be noticed also in figure 3 on page 12, indeed we have passed from a mean error of approximately 3000 pixel^2 to a mean error of approximately 30 pixel^2 in 56 steps, reducing it by 2 order of magnitude. The error has not gone to zero, some explanation for this behavior may be because we have taken enough steps or because there are other sources of error that are not due to radial distortion. If we consider "Image17" the reprojection error passed from 5010.4413 pixel^2 (without radial distortion) to 33.1665 pixel^2 (with compensation), this means that we were able to explain the effects of the radial distortion quite effectively;

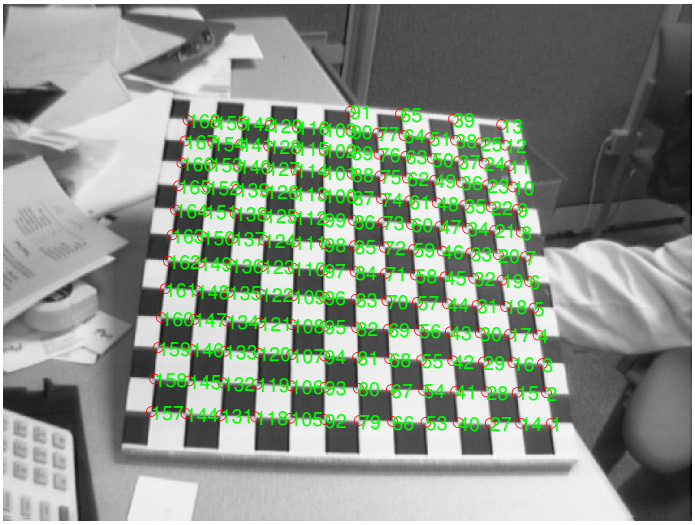
d. Results of the object superimposition on an image can be seen in figure 4 on page 12, where we have also reported the world reference frame for clarity.

e. If we look at the final estimated K matrix we can see that $K(2,2) \neq 0$, in fact its value is -0.8651, meaning that the image plane is slightly skew;

f. in addition to the aforementioned functions we have also implemented other utility functions:

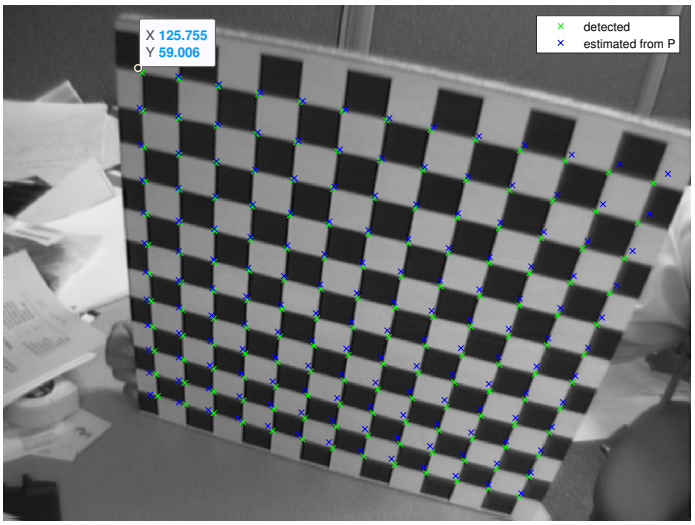
   - *getDetected()* that given a single image returns the image coordinates of the checkerboard's corners detected using *detectCheckerboardPoints()*;
   - *getAllDetected()* that does the same job of *getDetected()*, but returns all the image coordinates of all corners in all the images

(a) *Image*1

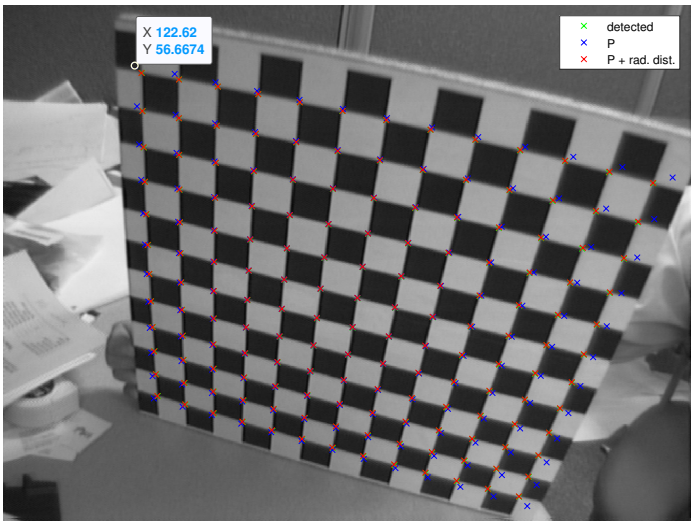


(b) *Image*2

Figura 1: Examples of the numbered square's corners detected using *detectCheckerboardPoints()*



(a) *Image*17, *after first estimate of* P *without radial distortion compensation*



(b) *Image*17, *after compensating for radial distortion*

Figura 2: Detected and projected checkerboard's corner of an image before and after radial distortion compensation

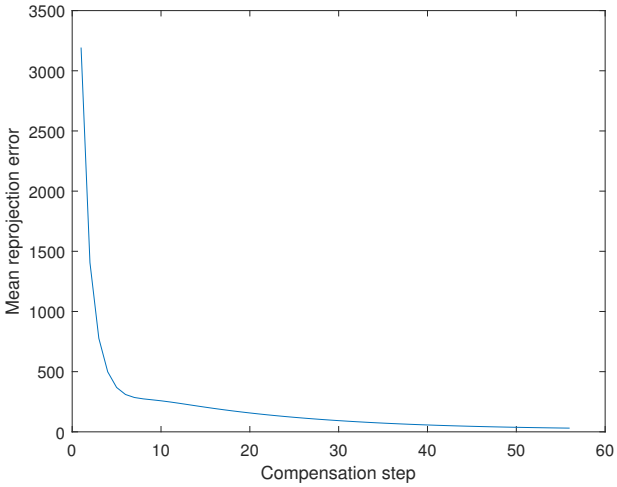


Figura 3: Plot of the mean reprojection error, with radial distorsion, between all images respect the compensation step

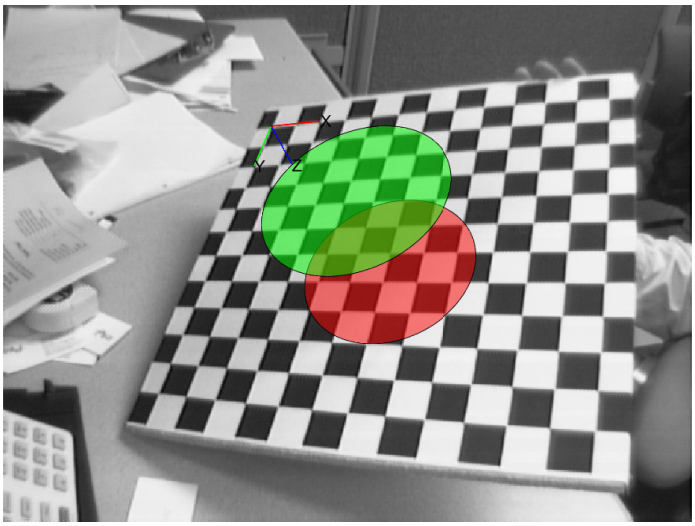


Figura 4: Example of cylinder superimposition on "Image4"

- *getProjected()* that returns the image coordinates of the checkerboard's corners projected in an image. Notice that the returned points does not take into account radial distortion.
- *getAllProjected()* that does the same job of *getProjected()*, but returns all the image coordinates of all corners in all the images

Riferimenti bibliografici

[1] Wilhelm Burger. *Zhang's Camera Calibration Algorithm: In-Depth Tutorial and Implementation*. University of Applied Sciences Upper Austria, School of Informatics, Communications e Media, Dept. of Digital Media, 2016. URL: https://www.researchgate.net/publication/303233579_Zhang's_Camera_Calibration_Algorithm_In-Depth_Tutorial_and_Implementation.

[2] Lorenzo Pantieri e Tommaso Gordini. *L'arte di scrivere con LaTeX*. Accessed: 2021-06-04. 2017.

[3] J. M. Smith e A. B. Jones. *Chemistry*. 7th. Publisher, 2012.

[4] *LaTeX Templates*. https://www.latextemplates.com/. Accessed: 2021-06-04. 2021.

[5] Zhengyou Zhang. *A Flexible New Technique for Camera Calibration*. 1998. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf.