Project Vision

Jing WANG

Yang LI

Sorbonne Université

Master 1 in Automation and Robotics in Intelligent Systems

Semester 2022-2023

24 décembre 2022

1 Introduction

The objective of this project is to build a 3D projection coming out from a pattern. As the pattern changes, the 3D projection change as well. To achieve this goal, we start with the camera calibration to get the intrinc matrix K of the camera. Secondly, we calculate homography to shift from one view to another view of the same scene. In other words, we choose a point in one view to find their corresponding point in another view. After that, the projection matrix P allow us to switch the 2D points to 3D points so that we can create a 3D projection (cube). Finally, we import a video and track the specified points in each frame and we will see that the cube change with the pattern in different angles.

2 Calibrate camera

The camera device we have been using in this TP is : iphone XR (12 megapixel wide camera). Because changes in focal length cause changes in the intrinc matrix, it is necessary to fix the focal length when taking pictures. In this TP we have always fixed the focal length to 0.6. For this image of a tessellated grid commonly used for camera correction, this image was taken from different angles using a camera with the parameters set. The length of the grid in the image was measured : 2.3cm.

Use the cameraCalibrator in Matlab to import all the photos taken.

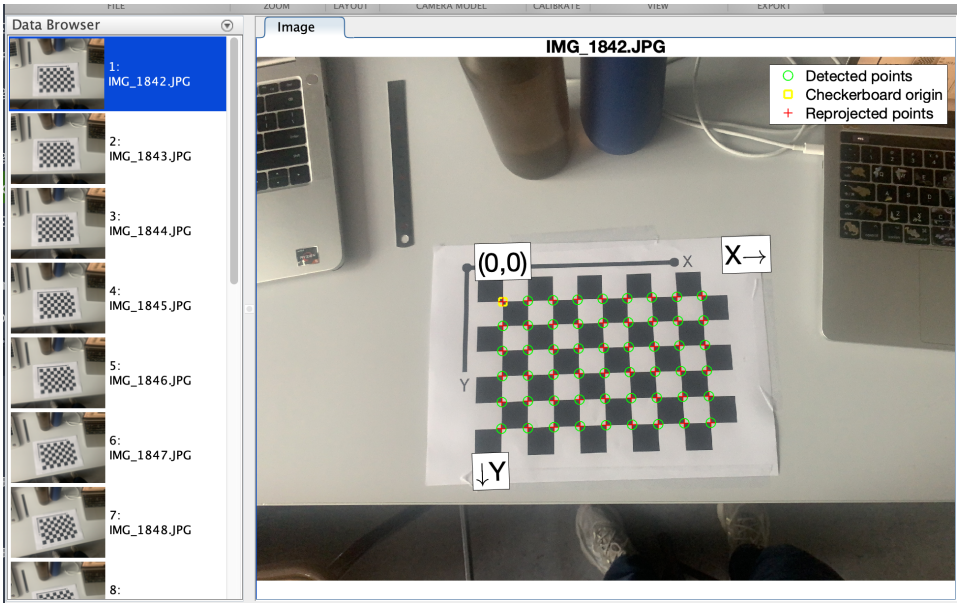


FIGURE 1 – Reprojection errors

The process of importing photos will reveal that some photos do not meet the requirements, the problem with these photos may be :
- Shooting multiple angles too high or too low to get the full image
- The image is not in the center
- The camera is tilted too far from the picture
- The light is not bright enough

Keep the value of Reprojetion Errors to within 0.5 by removing out-of-spec photos (figure 2).

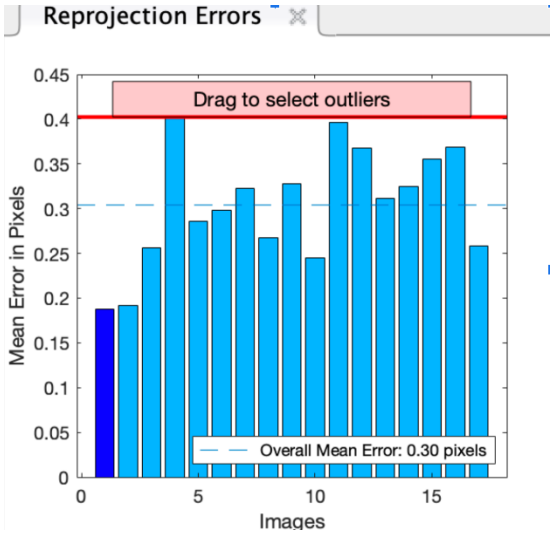
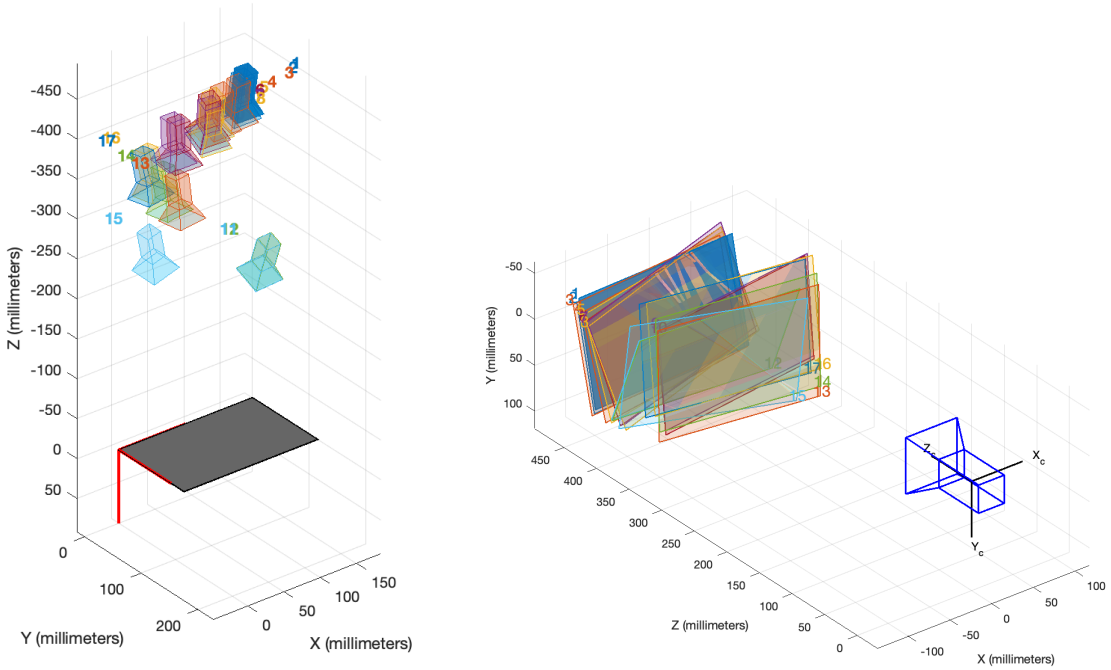


FIGURE 2 – Reprojection errors



(a) (b)

FIGURE 3 – Pattern-centric (a) and Camera-centric(b)

The form of K is

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The Intrinsic matrix of the camera can be obtained.(figure 4). The result need to be transferred.

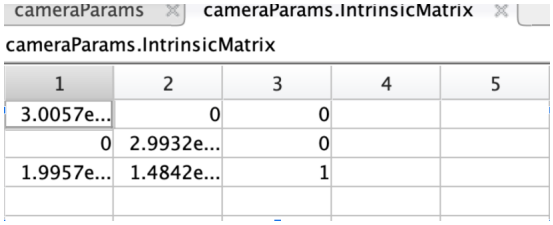


FIGURE 4 – Camera intrinsic matrix K^T

To see the effect of removing lens distortion, select "Show Undistorted" on the "Calibration" tab. If the calibration is accurate, the distorted lines in the image preview will become straight 5.

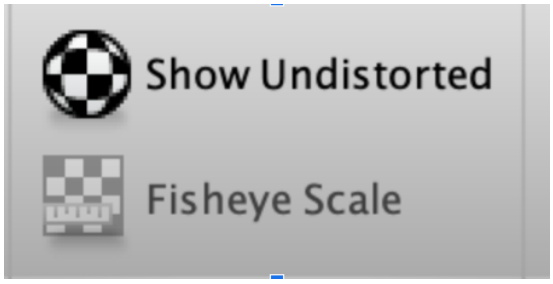


FIGURE 5 – Removing lens distortion

3 Homography

3.1 Theory

We can shift from one view to another view of the same scene by multiplying the homography matrix. With the points in one view, we try to find the corresponding point in another view. The homography transformation be expressed as follows :

$$\hat{P} = HP = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} P \tag{2}$$

Homography has 8 degree of freedom even though it contains 9 elements so the number of unknowns that need to be solved for is 8.
We need to find the corresponding set of points in two pictures with different perspectives. We assume P = $[x_s,y_s,1]^T$ and $\hat{P} = [x_d,y_d,1]^T$ in homogeneous coordinates. P and $\hat{P}$ are the same point

respectively in photo 1 and photo 2. Their coordinates satisfy the following conditions :

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} \tag{3}$$

$$\begin{cases} 0 = h_{11}x_s^i + h_{12}y_s^i + h_{13} - x_d^i(h_{31}x_s^i + h_{33}y_s^i + h_{33}) \\ 0 = h_{21}x_s^i + h_{22}y_s^i + h_{23} - y_d^i(h_{31}x_s^i + h_{33}y_s^i + h_{33}) \end{cases} \tag{4}$$

$$\begin{bmatrix} x_s^i & y_s^i & 1 & 0 & 0 & 0 & -x_d^i x_s^i & -x_d^i y_s^i & -x_d^i \\ 0 & 0 & 0 & x_s^i & y_s^i & 1 & -y_d^i x_s^i & -y_d^i y_s^i & -y_d^i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Each set of points gives us two equations, now there are 8 unknowns so we need at least 4 points to solve the system. Then we have chosen 4 sets of points and created a matrix of 8*9 as follows :

$$\begin{bmatrix} x_s^1 & y_s^1 & 1 & 0 & 0 & 0 & -x_d^1 x_s^1 & -x_d^1 y_s^1 & -x_d^1 \\ 0 & 0 & 0 & x_s^1 & y_s^1 & 1 & -y_d^1 x_s^1 & -y_d^1 y_s^1 & -y_d^1 \\ x_s^2 & y_s^2 & 1 & 0 & 0 & 0 & -x_d^2 x_s^2 & -x_d^2 y_s^2 & -x_d^2 \\ 0 & 0 & 0 & x_s^2 & y_s^2 & 1 & -y_d^2 x_s^2 & -y_d^2 y_s^2 & -y_d^2 \\ x_s^3 & y_s^3 & 1 & 0 & 0 & 0 & -x_d^3 x_s^3 & -x_d^3 y_s^3 & -x_d^3 \\ 0 & 0 & 0 & x_s^3 & y_s^3 & 1 & -y_d^3 x_s^3 & -y_d^3 y_s^3 & -y_d^3 \\ x_s^4 & y_s^4 & 1 & 0 & 0 & 0 & -x_d^4 x_s^4 & -x_d^4 y_s^4 & -x_d^4 \\ 0 & 0 & 0 & x_s^4 & y_s^4 & 1 & -y_d^4 x_s^4 & -y_d^4 y_s^4 & -y_d^4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We do SVD decomposition for this matrix of 8*9. H is the last column of V because we suppose that the diagonal matrix is sorted in descending order, so the last column in V corresponds to the smallest singular value. As the singular values determine the weight of the error, so we choose the last column of V as the solution. Then we reshape the matrix into a 3*3 matrix. Finally, we divide the matrix by h_{33} to standardise the matrix.

3.2 Code implementation

-Import two images
-Select 4 points (or more) from each of the two images
-Calculate the matrix H from the coordinates of the points. The theory is shown before.
-Verify. Selecting a point from one picture will automatically display the corresponding point on the other picture.

The red dots in the diagram indicate the 4 points we have chosen for calculating H.

Once we have completed the calculation we select the green point and the program will automatically display the corresponding point on the second diagram.

(a) (b)

FIGURE 6 – Original image (a) and second image(b)

We can see that our program is working well, the green dots on the two graphs correspond to each other.

3.3 Evaluation

To verify that our results were correct, we looked for code written by others on the internet (website is in the reference) and took the same points to calculate the matrix H. The code we found is in the figure 7. The two results are in the figure 8. We can see that both methods give the same result so our code is correct.

```
function H = CalcH(P1, P2)
x = P1(:, 1);
y = P1(:, 2);
X = P2(:, 1);
Y = P2(:, 2);
A = zeros(length(x(:))*2,9);
for i = 1:length(x(:)),
    a = [x(i),y(i),1];
    b = [0 0 0];
    c = [X(i);Y(i)];
    d = -c*a;
    A((i-1)*2+1:(i-1)*2+2,1:9) = [[a b;b a] d];
end
[U S V] = svd(A);
h = V(:,9);
H = reshape(h,3,3)';
H = H./H(3,3);
end
```

(a)

```
H = func_homography(P1,P2)

H1= CalcH(P1, P2)
```

(b)

FIGURE 7 – Code to calculate H (third-party method)

H1 H

3x3 double

	1	2	3
1	0.0281	-0.0162	1.7509e+03
2	-0.1822	0.4675	650.0282
3	-1.3636e-04	-2.9813e-06	1

(a)

H1

3x3 double

	1	2	3
1	0.0281	-0.0162	1.7509e+03
2	-0.1822	0.4675	650.0282
3	-1.3636e-04	-2.9813e-06	1

(b)

FIGURE 8 – The two results using our code(a) and third-party method(b)

4 Projection matrix P

4.1 Theory

Our objective is to project 3D image from 2D coordinates. To achieve this, we need to find the coordinates of the corresponding 3D point through the 2D point. The pin hole camera model will be used :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K(R|T) \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

with $[x,y]^T$ coordinates of the projection point in pixels(point 2D), [X,Y,Z] coordinates of 3D points, K camera intrinsic matrix (TP1) and (R|T)= rotation matrix and translation vector.Rotation matrix R stores camera's 3D Orientation and translation vector signifies camera's position in 3D space. Firstly, to calculate (R|T), we assume Z=0 for a point in the pattern plane. In this case, the third column of the R matrix is negligible.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{6}$$

which indicates that

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{7}$$

This form of matrix multiplication is similar to the previous homography. $\hat{P}$ = HP = K(R|T)P. We can deduce that H = aK(R|T). Here a is a factor to guarantee det(R) = 1. So a(R|T) = K^{-1}H.

$$a \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{8}$$

which equals to

$$a[R_1 \quad R_2 \quad T] = K^{-1}[H_1 \quad H_2 \quad H_3] \tag{9}$$

Here, we denote the i-th column of H as H_i (for example, $h_1 = [h_{11},h_{21},h_{31}]^T$) and the same for (R|T).

$$\begin{cases} aR_1 = K^{-1}H_1 = R_1' \\ aR_2 = K^{-1}H_2 = R_2' \\ aT = K^{-1}H_3 = T' \end{cases} \tag{10}$$

According to the properties of rotation matrix, each column of the rotation matrix is Orthogonal to each other. R3' = $_1$' × R_2' = $a^2K^{-1}H_1 \times K^{-1}H_2$. Now have the expression of R' :

$$[R_1' \quad R_2' \quad R_3'] = [aR_1 \quad aR_2 \quad a^2R_3] \longrightarrow detR' = a^4 det(R) \tag{11}$$

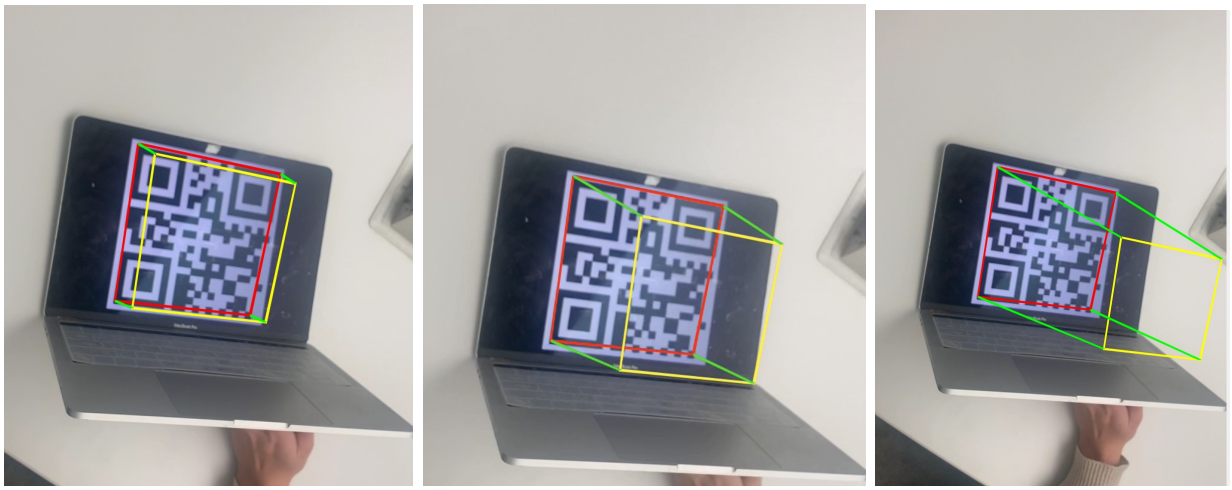
Knowing that det(R) = 1. So detR' = a^4. With this relation, we can calculate a = $\pm \sqrt[4]{det[R1' \quad R2' \quad R3']}$. The plus and minus signs indicate whether the camera is above or below the z=0 plane. For the two solutions, two cameras are symmetrical with respect to the z=0 plane. Here, we use the plus sign. Then, we can deduce the (R|T) matrix. To conclude,

$$(R|T) = [\frac{R1'}{a} \quad \frac{R2'}{a} \quad \frac{R3'}{a^2} \quad \frac{T'}{a}] \tag{12}$$

with the expression of R1' R2' R3' T' in the equation 4.1 and a = $\sqrt[4]{det[R1' \quad R2' \quad R3']}$.

4.2 Code implementation

Once we got the matrix K(R|T), we can project the 2D points into 3D. By changing the value of z, we can find the corresponding points and form a 3D cube. The projection matrix P = K(R|T). We set z to 50, 200 and 500. The result is in the figure 9. We can see that the height of the cube increases as z increases and our code works well. For the subsequent parts, we use z = 200.



(a) (b) (c)

FIGURE 9 – z = 50, z = 200, z = 500

5 Tracking points

In this section, we use Matlab's pointtracker, a function that helps us to automatically track a specified point.

First we will import a video (videoTest.mov) and then select four points in the first frame of this video. In our test we are using a QR code image, so we select the four vertices of this QR code.

The pointtracker will then be used to automatically track these four points. For each frame in the video, we use the previously calculated matrix H and projection matrix P to calculate the projection of the specified point in 3D and use the insertShape function to connect the specified and projected points for observation. Each of the processed frames is then stored in a new video file (videoProjection.mp4).

We note that this process is slow for matlab operations to process the video, with a 3-second video taking 2 minutes to process.

6 Conclusion

In this TP we first measured the intrinsic matrix of the camera using camera calibration in Matlab, then wrote the equation to calculate the matrix H and the equation to calculate the matrix

KRT. Using all the data and equations obtained above, we were able to project the points from 2D to 3D on a video.

7 Reference

Third-party method to calculate Homography :

$https://blog.csdn.net/u011624019/article/details/79560194$

Matlab official site :

$[camera-calibration]$

$(https://fr.mathworks.com/help/vision/camera-calibration.html)$

$[Vision.PointTracker]$

$(https://fr.mathworks.com/help/vision/ref/vision.pointtracker-system-object.html)$