# Computer Vision : Augmented Reality

SIVANESAN Shivamshan KINGSTON Kishanthan

December 24, 2022

# Contents

# List of Figures

# 1 Introduction

Augmented Reality is one of the domains that is constantly evolving with muliple possible applications such as for Architecture, Educations and Industrial Manufacturing and many more. AR combines the real world content and the ones computed by a computer but also allows us to interact with it. It make you see something which does not exist in your real environnement. One of the primary example of such technology is Pokemon Go.These technologies allow in a simple way to understand augmented reality.

In our case we are trying to project a 2D image in 3d using the basics of computer vision such as intrinsic, homography and projection matrices but at the same time we also have to take into account the perpective in which the image is taken, then noises that could possible appear when taking a photo etc... In order to do so, we will use the Computer Vision Toolbox, a module that has necessary tools to finalize the project.

# 2 Theory

The pinhole camera model describes how the 3D world is projected onto a 2D image, projected by a camera. This model gives us the algebrique model of the 3D world onto a 2D image by taking into account the focal distance, the pixels and the camera rotaion and translation.

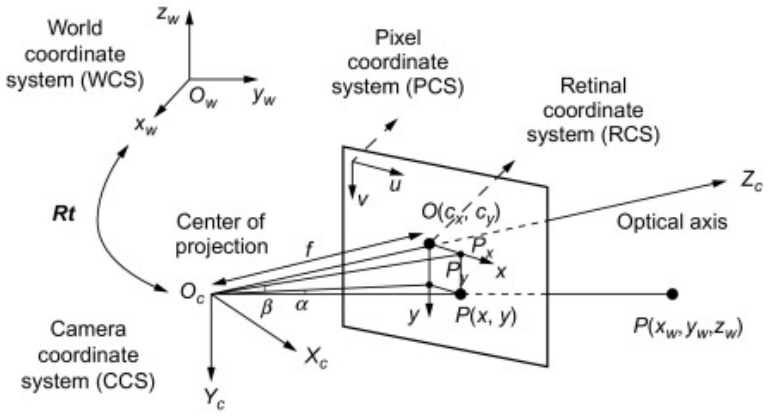


Figure 1: Pinhole camera model

Thus we have the following equation which will allow us to project the 3D cloud in the 2D image :

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = P * \begin{bmatrix} X_d \\ Y_d \\ Z_d \\ 1 \end{bmatrix}$$

with

$$P = K * (R|T) \tag{1}$$

This equation will be used to render a 2D image in 3D thus the effect of the augmented reality.

## 2.1 Intrinsic Matrix K

The camera calibration is the first step into the 3D projection process. We can define two parameters in camera calibration which are intrinsic and extrinsic parameters. We will only use the intrinsic parameters which will help

us to define the rotational, translation and projection matrices. The extrinsic matrix defines the transformation matrix from the world coordinate to the camera coordinates. Whereas the intrinsic matrix is linked to the focal of the camera. It helps to transform the camera coordinate to the pixel coordinate.

The camera calibration calculates the camera matrix by using both of these parameters but we will focus only on the intrinsic parameters which is a 3x3 upper triangle matrix defined as :

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$(f_x, f_y)$ : The focal length in pixels
$(c_x, c_y)$ : Optical center in pixels
$s$ : The skew coefficient

## 2.2 Homography Matrix H

The homography matrix is the relationship between the real world space and the image coordinates by mapping the corresponding points into a single image since they share the same central projection. This matrix is estimated using the direct linear transformation(DLT) based on the similarity between two images. In order to calculate the homography matrix, we will use an image of a pattern with four corners in which we know their real world coordinates of the pattern and to find the pixel coordinates which corresponds to the real world coordinates.

The homography matrix H is a 3x3 matrix, represented as :

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

$(x_d, y_d)$: The real world coordinates,
$(x_p, y_p)$ : The coordinates of pattern in pixel

As we can see there are 9 unknowns corresponding to the 9 degrees of freedom. After doing some manipulations, we obtain the following 8x9 matrix called A containing 4 pairs of linearly independant equation for each

pair of coordinates from the world and images pixels, the second matrix is a 1x9 matrix containing all variables of the homography matrix. Both of these matrices will be used to calculate the values of the variables of the homography matrix. One of the recurrent problem is that the extraction of images points cannot be done without imprecision which could effect the values of the homgraphy matrix.

$$\underbrace{\begin{bmatrix} x_d^1 & y_d^1 & 1 & 0 & 0 & 0 & -x_p^1 x_d^1 & -x_p^1 y_d^1 & -x_d^1 \\ 0 & 0 & 0 & x_d^1 & y_d^1 & 1 & -y_p^1 x_d^1 & -y_p^1 y_d^1 & -y_d^1 \\ x_d^2 & y_d^2 & 1 & 0 & 0 & 0 & -x_p^2 x_d^2 & -x_p^2 y_d^2 & -x_d^2 \\ 0 & 0 & 0 & x_d^2 & y_d^2 & 1 & -y_p^2 x_d^2 & -y_p^2 y_d^2 & -y_d^2 \\ x_d^3 & y_d^3 & 1 & 0 & 0 & 0 & -x_p^3 x_d^3 & -x_p^3 y_d^3 & -x_d^3 \\ 0 & 0 & 0 & x_d^3 & y_d^3 & 1 & -y_p^3 x_d^3 & -y_p^3 y_d^3 & -y_d^3 \\ x_d^4 & y_d^4 & 1 & 0 & 0 & 0 & -x_p^4 x_d^4 & -x_p^4 y_d^4 & -x_d^4 \\ 0 & 0 & 0 & x_d^4 & y_d^4 & 1 & -y_p^4 x_d^4 & -y_p^4 y_d^4 & -y_d^4 \\ x_d^5 & y_d^5 & 1 & 0 & 0 & 0 & -x_p^5 x_d^5 & -x_p^5 y_d^5 & -x_d^5 \\ 0 & 0 & 0 & x_d^5 & y_d^5 & 1 & -y_p^5 x_d^5 & -y_p^5 y_d^5 & -y_d^5 \\ x_d^6 & y_d^6 & 1 & 0 & 0 & 0 & -x_p^6 x_d^6 & -x_p^6 y_d^6 & -x_d^6 \\ 0 & 0 & 0 & x_d^6 & y_d^6 & 1 & -y_p^6 x_d^6 & -y_p^6 y_d^6 & -y_d^6 \\ x_d^7 & y_d^7 & 1 & 0 & 0 & 0 & -x_p^7 x_d^7 & -x_p^7 y_d^7 & -x_d^7 \\ 0 & 0 & 0 & x_d^7 & y_d^7 & 1 & -y_p^7 x_d^7 & -y_p^7 y_d^7 & -y_d^7 \\ x_d^8 & y_d^8 & 1 & 0 & 0 & 0 & -x_p^8 x_d^8 & -x_p^8 y_d^8 & -x_d^8 \\ 0 & 0 & 0 & x_d^8 & y_d^8 & 1 & -y_p^8 x_d^8 & -y_p^8 y_d^8 & -y_d^8 \end{bmatrix}}_{A} * \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{H} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

Since we know the values of the A matrix and to find the values of the homography matrix H all we have to do is to resolve the equation : $A * H = 0$ which means finding the values of H which satisfies the equation. One way of finding these values is to calculate the matrix A then using the Singular Value Decomposition to find the singular value. The value of H is usually the last column vector of V. Once we have the homography matrix, we will normalize the matrix by dividing every single value of the homography matrix by the last value of the homography matrix $H_{33}$.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} / H_{33}$$

The DLT gives us the H matrix without separting the intrinsic matrix K, the roational matrix R and the translation vector T. This brings us to the projection matrix which will allow us to decompose H into H into $H = K * (R|T) = \alpha * K * (R|T)$

## 2.3 Projection Matrix P

The projection matrix allows us to project our 2d coordinate points in 3d represented in homogenous coordinates. Thus using all the information from above, it's possible to get the pinhole model as follows :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K * (R|T) * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

$$M' = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \; M = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$M' = P * M = K(RT) * M$ , with M' projected coordinates of four square's corners, M real coordinates of the square corners, P the projection matrix which is 3x4 matrix, K intrinsic matrix and RT rotation matrix and translation vector. We use the homography and the intrinsic matrix to find our rotation (R) and translation (T) matrices.

So if we rewrite the equation n°5, we have the following set:

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

For a point in the pattern plane, Z = 0 :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & 0 & t_x \\ r_{21} & r_{22} & 0 & t_y \\ r_{31} & r_{32} & 0 & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & s & c_x \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix}}_{=H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{7}$$

As we can see that the only known matrices are the intrinsic matrix K and the homography matrix H, the matrix containing the rotational matrix and the translation vector are unknown but we can see that the homography and the camera pose contain the same information hence we can pass the information between them easily since we know that : $H = K * (R|T)$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{8}$$

Finally we obtain the following equation that gives the values of the rotational and translation matrix:

$$\underbrace{\begin{bmatrix} \widetilde{r_{11}} & \widetilde{r_{12}} & \widetilde{r_{13}} & \widetilde{t_x} \\ \widetilde{r_{21}} & \widetilde{r_{22}} & \widetilde{r_{23}} & \widetilde{t_y} \\ \widetilde{r_{31}} & \widetilde{r_{32}} & \widetilde{r_{33}} & \widetilde{t_z} \end{bmatrix}}_{(R|T) approximative} = \begin{bmatrix} f_x & s & c_x \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}^{-1} * \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{9}$$

Essentially, we need to do is to find the $\widetilde{R^1}$ and $\widetilde{R^2}$ and $\widetilde{T}$. And $\widetilde{R^3}$ can be found by doing the cross product of $\widetilde{R^1}$ and $\widetilde{R^2}$ since $\widetilde{R^3}$ is collinear to $\widetilde{R^1}$ and $\widetilde{R^2}$.

The values of the rotational matrix and the translation vector that we get through the calculations are approximative due to noise in the data of image point coordinates, so the real formula $H = \alpha * K * (R|T)$ where $\alpha$ is a scale factor that we need to find in order to get real rotation matrix and translation vector.

Once the value of scale factor $\alpha$ found, all we have to do is to multiply the $\widetilde{R^1}$ and $\widetilde{R^2}$ and $\tilde{T}$ by the inverse of the scale factor and redo a cross product between the new $R^1$ and $R^2$ to get $R^3$ this give us the rotational matrix R and the translation vector T :

$$R^1 = \widetilde{R^1} * 1/\alpha$$

$$R^2 = \widetilde{R^2} * 1/\alpha$$

$$T = \widetilde{T} * 1/\alpha$$

$$R^3 = R^1 \times R^2 \tag{10}$$

Finally, we can calculate the projection matrix P which gives us the following equation :

$$P = K * (R|T) \tag{11}$$

Thus, we have all the necessary matrices needed to calculate the projection points in the 3D world and to project these points onto our pattern.

# 3 Experience and Results

In this section, we are putting into motion the theory by using Matlab and the Computer Vision Toolbox to retrieve the intrinsic matrix of the camera by doing camera calibration, detecting the pose of pattern and calculating the projection matrix etc.. In order to project the 2D image in the 3D world, we start by detecting the pose of each pattern by hand of a simple square image and calculating each matrices(K,H,P) after that, we will focus on estimating the poses of the square points automatically with tools from the computer Vision Toolbox.

## 3.1 Intrinsic Matrix

To start with, most cameras uses an automatic calibration system in order to take photos, thus ignoring the details of an image. In this section we will focus on ajusting our camera to use manual focus instead of an automatic one. To do so, we used an application called **YAMERA**, which only required us to ajust the focus of the camera maunally until the image that appeared on the screen had all the details and wasn't blurred. To extract the camera's parameters, the standard process consits of taking an image of an object whose structure is perfectly-known called a calibration chart(mire de calibrage). In our case, the image is an image of a checkered board in which the camera calibration should identify the corner between the black and white boxes.

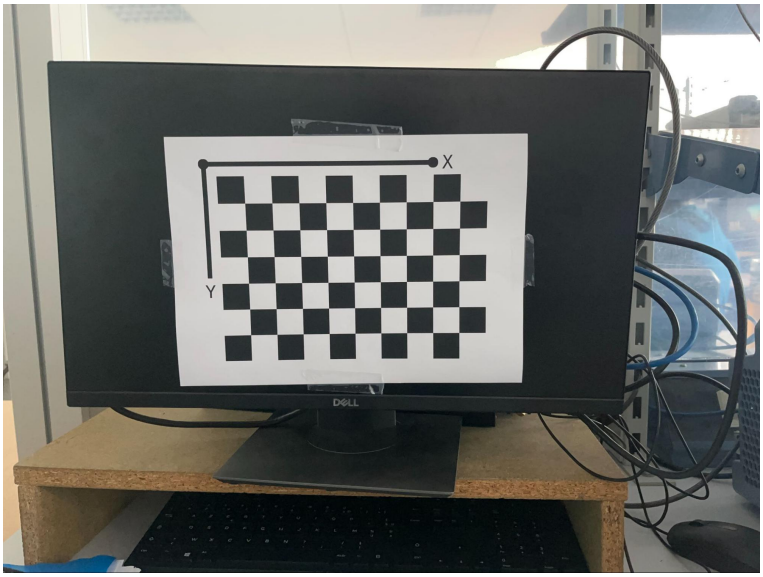


Figure 2: Pattern used for camera calibration

The manual focus of the camera was necessary since we needed it to acquire the intrinsic matrix K of the camera. In order to do a precise camera calibration, we needed to have :

- At least 30 images taken from different orientations

- The calibration pattern shouldn't be over 45° relatively to the camera plane
- Avoid putting images with certain blur in them.

Once the images taken with the pattern on it by using the manual focus, we enter the following command on MATLAB's command prompt : cameraCalibrator for camera calibration. Then we add all the taken images and finally we use the Calibrate button the Calibration tab using the default settings and using the standard camera model to start the calibration process.

Using the default settings, the algorithm gives us the calibration accurracy. This gives us 2 different methods which can be used to evaluate and understand the accurracy using the default settings before adjusting them. The methods we used to evaluate the results were:

- The reprojection error which gives us an overall mean error which needed to be smaller than 0.5. We used this to find the images which had an impact on this mean error thus the precision of the calibration.
- The Pattern centric and camera centric plots, if any error occured such as an image appearing behind the camera

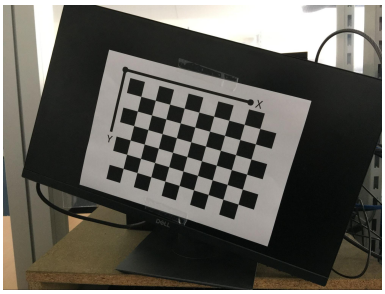

Figure 3: Exemple of error caused by quality of the image

For the first process using the default settings, we got an overall mean error of **0.75** pixels. In order to reduce the reprojection error, we delete the image that has a high reprojection error one at a time and recalibrating again. After deleting an image we check the undistorted image to check if the pattern is entierly taken into account. The recurrent problems, that we observed is that some of the images given for calibration process were blurry

or perhaps had an angle greater than 45°. Another method that we used is to add more images taken in different orientations to improve the mean error.

We continue to do so until we obtain a mean error smaller is than 0.5 but at the same time verifying that the undistorted version of the image takes into account all the pattern and straightens the distorted lines. We finally got an overall mean error of**0.23** pixels with 20 images.

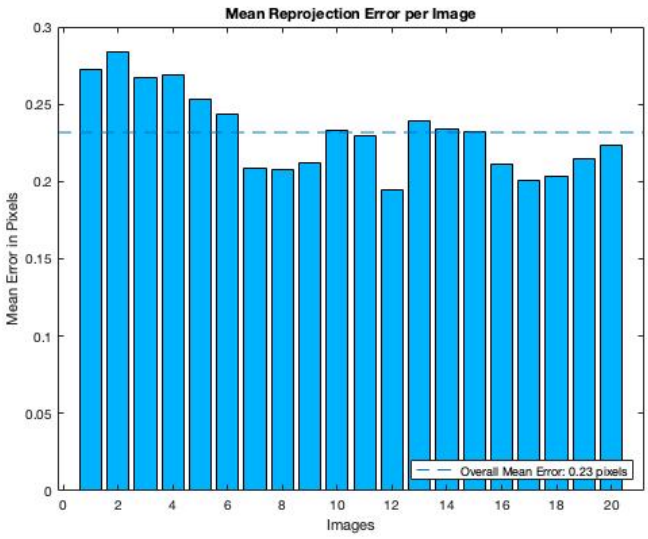


Figure 4: Mean projection error per image

Following the camera calibration process that gave us an overall projection mean error of 0.23 pixels. We can retrieve the intrinsic matrix K of the camera through the variable called cameraParameters. This is the following matrix obtained through this process whose values are in pixels :

$$K = \begin{bmatrix} 3.0730e+3 & 0 & 2.0332e+3 \\ 0 & 3.0654e+3 & 1.5007e+3 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

This matrix ressembles to the one defined in the Theortical section of the intrinsic matrix. As we can see, that the matrix is upper triangle matrix and could be identified to each variable of the intrinsic matrix.

## 3.2 Homography Matrix

We have created a function in matlab which allows us to directly calculate our homography matrix using the coordinates of our four different corners of our square using the world coordinates and their image points. The function Singular Value Decomposition (SVD) method is then used to find the H (homography matrix) which usually correspond to the last column of the V matrix that gives us the homography matrix.

To begin with, the image we used to calculate the homography matrix is represented in the following figure :

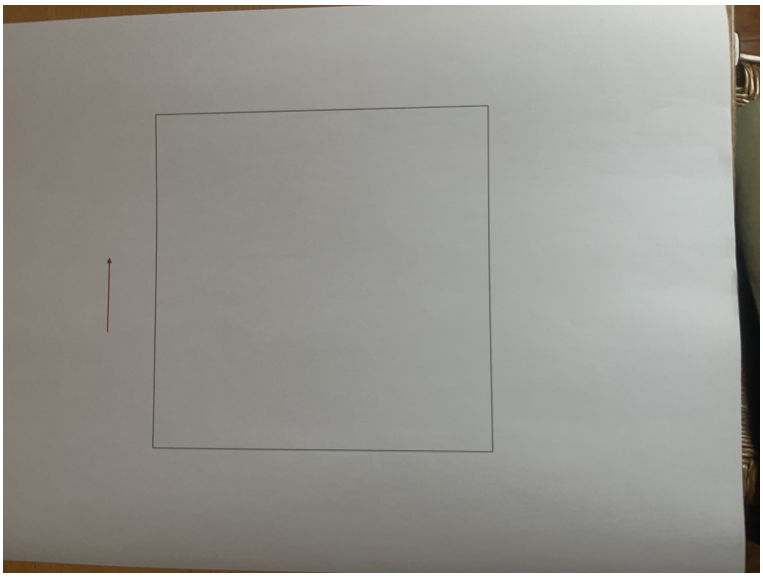

Figure 5: Simple pattern used to calculate the homography

Using this image where we know the world coordinates of each points in the square, we can retrieve the 4 image coordinate points(x,y) of the square using **ginput** which return a 4x2 matrix containing the x and y values of each point of the square.

Following this we can calclate the A matrix (4) using a for loop and then we can use the SVD function to calculate and retrieve the last column of V matrix and transform that matrix into a 3x3 matrix by iterating through the last column and adding it to the H matrix. We also normalize the H matrix that we get by dividing the matrix by the value of H(3,3).

This is the following matrix H that we get for the pattern that we used:

$$H = \begin{bmatrix} 287.7108 & 6.3324e-12 & 84.0000 \\ -0.6376 & 286.0000 & 183.0000 \\ -0.0035 & 2.2786e-14 & 1 \end{bmatrix} \tag{13}$$

To see if the matrix H is correct, we use the equation (2) to verify to see if we get the x and y coordinates of the image points of the square:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} 287.7108 & 6.3324e-12 & 84.0000 \\ -0.6376 & 286.0000 & 183.0000 \\ -0.0035 & 2.2786e-14 & 1 \end{bmatrix} * \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{14}$$

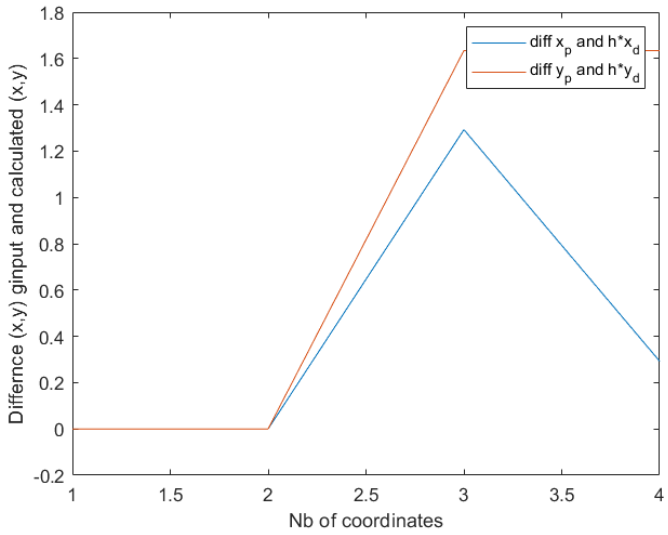


Figure 6: Difference between the (x,y) image points and the ones calulated through homography

Thus we obtain the following figure (6) above which shows us the difference between the image points that we get using ginput function and the image points that we calculated using the $H * \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$ As we can see the difference

isn't entirely equal to zero for the last two coordinates but at least nears the zeros $10^{-12}$ present in the vertical axis. We can also see that there are certain variations on level of differences between the image points found through ginput and the calculated ones. This difference could also appear when we try to detect the image points automatically but are negliable. The following image is when we stack the image points that we get from ginput and the image points calculated through the equation.

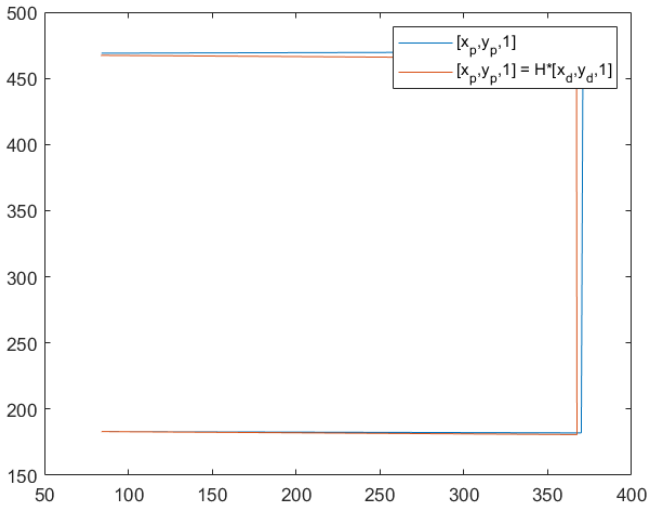


Figure 7: Stacking the values of (x,y) image points and (x,y) image points calculated through the homography

## 3.3 Projection Matrix

For the projection matrix all we need is to get the H matrix and intrinsic matrix K that we have calculated before. For the projection matrix, we have created a function that takes both H and K matrices and returns the P the projection matrix.

Just as we mentionned in the Theory section for the projection matrix, we use the equation(4) to calculate the $(R|T)$ matrix and then we use this equation : $H = \alpha * K * (R|T)$, we begin by finding the scale factor $\alpha$ by using

the function **nthroot**. This function takes the determinant of our matrix RT and another input which correponds to size of our matrix.

$$\alpha = nthroot(det((R|T)), 4) \tag{15}$$

Once we find the scale factor $\alpha$ we use the equation (9) to get the real values of RT matrix which finally enables us to calculate the projection matrix P by using the equation (10).

The following matrix correspond to matrix $(R|T)$:

$$(R|T) = \begin{bmatrix} 0.9970 & -5.9105e-15 & -1.5724e-13 & -6.7944 \\ -7.2308e-14 & 1.0030 & -3.5952e-14 & -4.6050 \\ 1.5677e-13 & 3.6060e-14 & 1 & 10.7127 \end{bmatrix} \tag{16}$$

This is the projection matrix that we get for the pattern that we used :

$$P = \begin{bmatrix} 3.0638e+3 & 5.5146e-11 & 2.0330e+3 & 899.8708 \\ 1.3619e-11 & 3.0746e+3 & 1.5007e+3 & 1.9604e+3 \\ 1.5677e-13 & 3.6060e-14 & 1 & 10.7127 \end{bmatrix} \tag{17}$$

As we can see that the following 3x3 matrix of P ressembles the intrinsic matrix K with some exceptions there aren't any huge changes on the diagonal values or to last column of this matrix, the only differences are at these positions P(2,1),P(3,1),P(3,2) and P(1,2) are not equal to zero but approaches it.

$$\begin{bmatrix} 3.0638e+3 & 5.5146e-11 & 2.0330e+3 \\ 1.3619e-11 & 3.0746e+3 & 1.5007e+3 \\ 1.5677e-13 & 3.6060e-14 & 1 \end{bmatrix} \approx K \tag{18}$$

Once we have the projection matrix P, we can project our 2D image to create a cube. We know the real coordinates of our square where Z = 0 to create a cube we simply create another set of the same coordinates where Z = 1 in this set. To project our points,we simply multiply the two set of our real world coordinates by the matrix P to get the 8 image points of our cube to stack it into our image.

$$P_i' = P * C_i \tag{19}$$

$P_i'$ corresponds to the projected point and the index i corresponds to each point of our cube and finally the $C_i$ corresponds to the real world coordinates where there a 4 sets which has Z = 0 and another 4 set Z = 1.

Normalization on the projected point $P_i'$ by dividing it by the third value of the projected point:

$$P_i' = P_i'/P_i'(3) \tag{20}$$

The entire process of projecting points is done using another function called Point Projection which takes the projection matrix P as an input and returns all the projected points(x,y).

The following figure is what we get for the projected points and to get the image points of our square, we continue to use ginput:

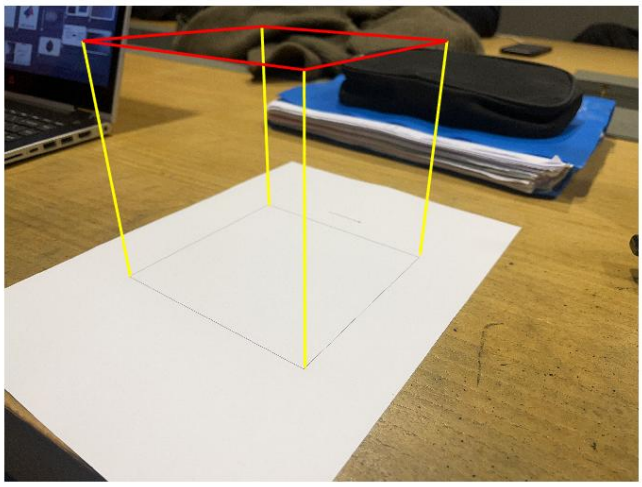

Figure 8: Projection of points on the pattern

## 3.4 Pose estimation

This brings us to the final problem of this project that consists of detecting these image points automatically not by using ginput. To do so, we have multiple differents tools that can be used, furthermore we will start using this pattern in order to detect these image points:

As you can see there 4 circles each with different sizes the reason behind this, will be explained down below. There are multiple tools that the Computer Vision Toolbox gives us to our disposal such as :

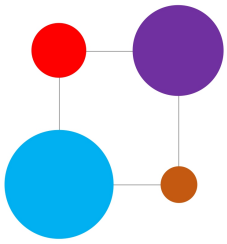

Figure 9: Final pattern used for automatic pattern detection

- vision.PointTracker
- Binarizing the images using threshold in order to track the largest contours on images using imbinarize,harrisDetection for the tracking the contours.
- imfindcircles to detect coloured disks.

We used the third method for the automatic pose estimation. To do this, we have created function called Detection which takes the image as the input and return the x and y image coordinates. As for the imfindcircles this function returns the (x,y) coordinates of the center of the circle that it has detected. In order to detect each of these circles properly, we have differents sizes for each point of the square in order to get the image (x,y) coordinates of the square, if they happen to be same size then, this will only take into account the first circle and will return only those coordinates instead of other 3 circles. Each radius of the circle is known.

The setting required to find the circles properly have been tested and changed multiple times such as the ObjectPolarity to dark since the circle colours are darker than the background on the photos but also the higher sensitivity(0.98) to detect the circles.

Once the pose estimation is done, we will use a series of images taken from different orientations and project the 3D cube on to these images and relying the each points between them to create a cube.

The figure below correspond to the projection of points on the pattern that were detected automatically :

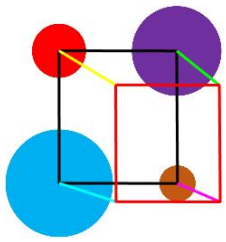

Figure 10: Point projection on the final pattern through automatic detection

# 4 Conclusion

In conclusion, the implementation of the augmented reality by projecting the 2D image in 3D, by taking into account the camera matrix, the homography that maps the corresponding points between the real world space and the image coordinates into a single image, the rotational matrix R and the translation vector which enabled us to calculate the P, the projection matrix. Finally the most complicated part of the project consisted of detecting these points automatically hence multiple trial and errors were necessary to find the best way to detect the edge of squares precisely and to plot the stack the 3D cube onto the taken image.

# 5 Source

# References

[1] What are Intrinsic and Extrinsic Camera Parameters in Computer Vision? — by Aqeel Anwar — Towards Data Science.

[2] https://fr.mathworks.com/help/vision/ug/camera-calibration.html

[3] OpenCV: Basic concepts of the homography explained with code

[4] https://fr.mathworks.com/help/images/ref/imfindcircles.html