SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

MU4RBI09
Computer Vision Course Project

Made by : Zahra BENSLIMANE & M'hamed BENABID.

Group : App 1 ISI

Table of Contents

MU4RBI09 : Computer Vision Course Project
Lab session 01 :
-2D Homography
-What is the homography matrix?
-Calculating homography using the Direct Linear Transform (DLT) algorithm on point correspondences.

Lab session 02 : Camera Calibration
-Estimating the intrinsic Camera parameters with MATLAB's cameraCalibrator App
-What are camera intrinsics?
-Estimating the extrinsic camera parameters
-What are camera extrinsics?

Lab session 03 : Object Recognition and Tracking for Augmented Reality
-Calculating the projection matrices.
-The encountered difficulties.
-Conclusion.

Lab session 01 : 2D Homography

Goal : Estimate the homography matrix from corresponding image points (2D-2D planar Homography).

What is the homography matrix?

The planar homography relates the transformation between two planes (up to a scale factor). In other words, the homography relates points in the first view to points in the second view.

Once camera rotation and translation have been extracted from an estimated homography matrix, this information may be used for navigation, or to insert models of 3D objects into an image or video, so that they are rendered with the correct perspective and appear to have been part of the original scene. [1]- Wikipédia

Calculating homography using the Direct Linear Transform (DLT) algorithm on point correspondences.

In the following section, we will Implement a function that takes a set of points m^i_{image1} and m^i_{image2} $\in R^2$ as inputs and returns the 3 by 3 estimated homography matrix H such that :

$M^{\tilde{i}}_{\text{image1}} = H.M^{\tilde{i}}_{\text{image2}}$ .... (1)

where $M^{\tilde{i}}_{\text{image1}}$ and $M^{\tilde{i}}_{\text{image2}}$ are the homogeneous vectors representing the corresponding points.

From (1), we get :

$$\begin{bmatrix} u_{\text{image1}} \\ v_{\text{image1}} \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} . \begin{bmatrix} u_{\text{image2}} \\ v_{\text{image2}} \\ 1 \end{bmatrix}$$

we can write $\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$ as $\begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix}$ where h_i^T is the i^{th} line of the Homography matrix.

since there is an equality between the two vectors, clearly $M^{\tilde{i}}_{\text{image1}} \times H.M^{\tilde{i}}_{\text{image2}} = 0$

thus,

$$\begin{bmatrix} u_{\text{image1}} \\ v_{\text{image1}} \\ 1 \end{bmatrix} \times \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} . \tilde{M}^i_{\text{image2}} = 0 \Longrightarrow \begin{bmatrix} u_{\text{image1}} \\ v_{\text{image1}} \\ 1 \end{bmatrix} \times \begin{bmatrix} h_1^T . \tilde{M}^i_{\text{image2}} \\ h_2^T . \tilde{M}^i_{\text{image2}} \\ h_3^T . \tilde{M}^i_{\text{image2}} \end{bmatrix} = 0 \Rightarrow \begin{bmatrix} v_{\text{image1}} . h_3^T . \tilde{M}^i_{\text{image2}} - h_2^T . \tilde{M}^i_{\text{image2}} \\ h_1^T . \tilde{M}^i_{\text{image2}} - u_{\text{image1}} . \ h_3^T . \tilde{M}^i_{\text{image2}} \\ u_{\text{image1}} . h_2^T . \tilde{M}^i_{\text{image2}} - v_{\text{image1}} . h_1^T . \tilde{M}^i_{\text{image2}} \end{bmatrix} = 0$$

By rearranging the equation, we get :

$$\begin{bmatrix} O_3^T & -M^i_{\text{image2}} & v_{\text{image1}} . \tilde{M}^i_{\text{image2}} \\ M^i_{\text{image2}} & O_3^T & -u_{\text{image1}} . \tilde{M}^i_{\text{image2}} \\ u_{\text{image1}} . \tilde{M}^i_{\text{image2}} & -v_{\text{image1}} . \tilde{M}^i_{\text{image2}} & O_3^T \end{bmatrix} . \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0$$

with $O_3^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$

Each pair of points generates two linearly independent equations of the form.

$$\begin{bmatrix} O_3^T & -M^i_{\text{image2}} & v^i_{\text{image1}} . \tilde{M}^i_{\text{image2}} \\ M^i_{\text{image2}} & O_3^T & -u^i_{\text{image1}} . \tilde{M}^i_{\text{image2}} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0$$

With n pairs we get 2n equations on 9 unknowns ! So we only need 4 set of pairs

It is then possible to stack them into a matrix A to compute:

$$A. H = 0$$

Such as:

$$A. H = \begin{bmatrix} O_3^T & -M^1_{\text{image2}} & v^1_{\text{image1}} . \tilde{M}^1_{\text{image2}} \\ M^1_{\text{image2}} & O_3^T & -u^1_{\text{image1}} . \tilde{M}^1_{\text{image2}} \\ \dots & \dots & \dots \\ O_3^T & -M^n_{\text{image2}} & v^n_{\text{image1}} . \tilde{M}^n_{\text{image2}} \\ M^n_{\text{image2}} & O_3^T & -u^n_{\text{image1}} . \tilde{M}^n_{\text{image2}} \end{bmatrix} . \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0$$

Then we applied the matlab built in function SVD to compute the singular value decomposition P=USV^T and select the last singular vector of V as the solution to H.

Lab session 02 : Camera Calibration

Goal : Calibrating our phone cameras by estimating the intrinsic and extrinsic parameters from 1 or multiple images.

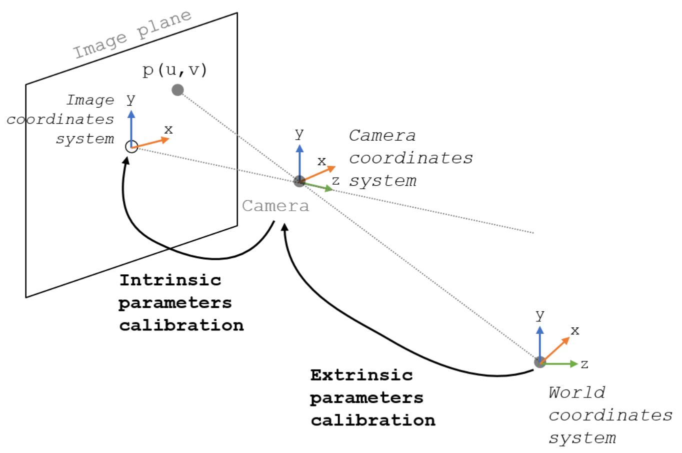


Image plane
p(u,v)
Image coordinates system
y
x
Camera coordinates system
y
x
z
Camera
Intrinsic parameters calibration
Extrinsic parameters calibration
World coordinates system
y
x
z

Estimating the instrinc Camera parameters with MATLAB's cameraCalibrator App

What are camera intrinsics?

https://dsp.stackexchange.com/questions/2736/step-by-step-camera-pose-estimation-for-visual-tracking-and-planar-markers/2737#2737

It allows mapping between pixel coordinates and camera coordinates in the image frame

For this matter, we use the Single **Camera Calibrator App : cameraCalibrator** found in the Computer Vision Toolbox.

The key steps and requirement to use this app are copied and summarize below [2]- Matlab Help

1. Prepare images, camera, and calibration pattern.
2. Add images and select standard or fisheye camera models.
3. Calibrate the camera.

The **Camera Calibrator** app uses a checkerboard pattern. We use a checkerboard that contains an even number of squares along one edge and an odd number of squares along the other edge.This criteria enables the app to determine the orientation of the pattern and the origin. The calibrator assigns the longer side to be the *x*-direction.

We use at least 10 to 20 images of the calibration pattern to estimate the parameters, we made sure that we :

- Do not modify the images, (for example, do not crop them).
- Do not use autofocus or change the zoom settings between images.
- Capture the images of a checkerboard pattern at different orientations relative to the camera

Measure one side of the checkerboard square and give it as input to the cameraCalibrator while uploading the images.

Estimating the extrinsic camera parameters

What are camera extrinsics?

It describes the orientation and location of the camera. This refers to the rotation and translation of the camera with respect to some world coordinate system.

The Zhang analytical pose estimator:

The homography can be estimated using the DLT algorithm . As the object is planar, the transformation between points expressed in the object frame and projected points into the image plane expressed in the normalized camera frame is a homography. Only because the object is planar, the camera pose can be retrieved from the homography, assuming the camera intrinsic parameters are known.

let $\{\mathrm{Mi}\}$ be a set of world coordinates for forming a plane for each image and $\{\mathrm{mi}\}$ their corresponding projected image coordinates, we can write $\{\mathrm{mi}\} = H.\{\mathrm{Mi}\}$ where $\mathrm{Mi} = \begin{bmatrix} \mathrm{Xi} \\ \mathrm{Yi} \\ \mathrm{Zi} \end{bmatrix}$ and H the homography between the points.

in a general case, if P is the projection matrix that maps a 3D point in space to a 2D point in the image plane, then : $P.\widetilde{\mathrm{Mi}} = K\,(R\,,\,T\,)\begin{bmatrix} \mathrm{Xi} \\ \mathrm{Yi} \\ \mathrm{Zi} \\ 1 \end{bmatrix} \equiv \widetilde{\mathrm{mi}}$

to simplify the equation, we take $\mathrm{Zi} = 0$ and take out r3 since it has no contribution,

$$P.\widetilde{\mathrm{M}}\mathrm{i} = K\,(\mathrm{r1}, \mathrm{r2}, \mathrm{r3}\,,\ T\,) \begin{bmatrix} \mathrm{Xi} \\ \mathrm{Yi} \\ 0 \\ 1 \end{bmatrix} \equiv \widetilde{\mathrm{m}}\mathrm{i} \qquad \Longrightarrow \quad \widetilde{\mathrm{M}}\mathrm{i} = K\,(\mathrm{r1}, \mathrm{r2}\,,\ T\,) \begin{bmatrix} \mathrm{Xi} \\ \mathrm{Yi} \\ 1 \end{bmatrix} \equiv H.\widetilde{\mathrm{m}}\mathrm{i}$$

so, $K\,(\mathrm{r1}, \mathrm{r2}\,,\ T\,) \equiv H \Longrightarrow \lambda\,.\ (\mathrm{r1}, \mathrm{r2}\,,\ T\,) = K^{-1}\,.\,H$

knowing that the rotation Matrix is an orthogonal matrix, the cross product of the 2 first column vectors gives us the last one : $\lambda\,.\,\mathrm{r1} \times \lambda\,.\,\mathrm{r2} = \lambda^{2}\,.\,\mathrm{r3}$

$\det(\lambda\,.\,\mathrm{r1},\ \lambda\,.\,\mathrm{r2}\,,\ \lambda^{2}\,.\,\mathrm{r3}) = \lambda^{4}\,.\,\det(\mathrm{r1}, \mathrm{r2}, \mathrm{r3})$, however $\det(\mathrm{r1}, \mathrm{r2}, \mathrm{r3})$ = 1 ,

hence :

$$\det(\lambda\,.\,\mathrm{r1},\ \lambda\,.\,\mathrm{r2}\,,\ \lambda^{2}\,.\,\mathrm{r3}) = \lambda^{4} \Longrightarrow \lambda = \sqrt[4]{\det(\lambda\,.\,\mathrm{r1},\ \lambda\,.\,\mathrm{r2}\,,\ \lambda^{2}\,.\,\mathrm{r3})} \ \text{ or } \ \lambda = -\sqrt[4]{\det(\lambda\,.\,\mathrm{r1},\ \lambda\,.\,\mathrm{r2}\,,\ \lambda^{2}\,.\,\mathrm{r3})}$$

Lab session 03 :

Object Recognition and Tracking for Augmented Reality

Goal : **integrate virtual 3D objects** on the image.

1 - Record video of any sort of marker* while rotating around it.

3 - Estimate the pose of the camera video frame

4 - Map the world coordinates into the image plane.

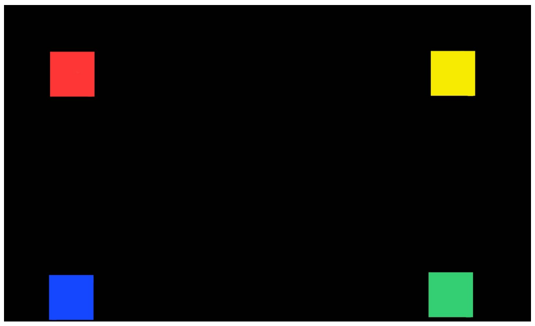
We call the distinctive picture that can be recognised by the device, the marker. A marker can be anything, as long as it has enough unique visual points. Images with lots of corners and edges work especially well.

To simplify the task at hand, we used an RGB marker in the form of a Polygon.

1. Calculating the projection matrices :

<u>The marker :</u>

In this stage, the choice of which object to use to map the world coordinates onto the image plane is very important. We chose to draw 4 rectangles, each on a different corner and each colored differently, onto a black surface to solve the predicament.



<u>The world points :</u>

The center of the red square = [0 0]

The center of the yellow square = [187 1]

The center of the green square = [186 108]

The center of the blue square = [0 109]

Point correspondences : Image points - World points:

The decision to color each square differently and to place them on a black surface, was made to facilitate the use of color filters. On each image, four different color filters, red, blue, yellow and green, will be added to isolate each corresponding square.

Computing the centroids of each color square :

Each mask gives us a binary image containing the pixels corresponding to a certain color. We go through the binary images and compute the mean value of their x and y coordinates. We then take these centroids as our projected image points.

Computing the trajectory :

By going through the video frame by frame and computing the centroids, we are able to have a precise position of each square; this will be used to position a 3D object onto these sets of points to give the illusion of a rotating object.

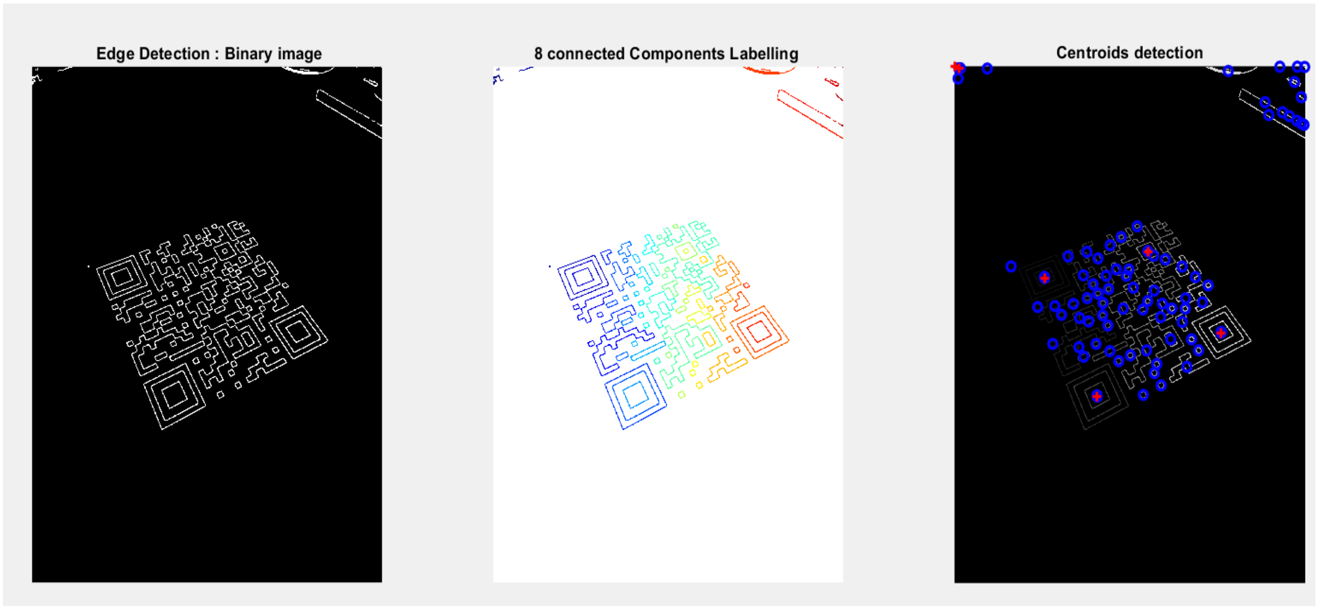
2. The encountered Difficulties :

Before considering the solution detailed above to find the point correspondences on the image plane, we started with the following idea:

- Converting the original image coded by RGB planes to an image coded in grayscale.
- Computing the gaussian gradient derivative masks Gx Gy along the x and y axis.
- Computing the derivative of the grayscale image by convoluting it with Gx and Gy.
- Computing the magnitude of the gradient at each pixel.
- Binarizing the image.
- Apply morphological operations(Opening & closing) to remove the salt&pepper noise from the image
- Applying the 8 connected Components Labeling.
- Plot the centroids of each blob.

The position detection patterns of a QR code are the 3 square eyes (square inside square inside square), so to extract only the centroid on these eyes + the alignment pattern we only took into consideration the centroids who belong to two or more blobs (up to an epsilon Error tolerated)

- We loop through each video frame and stock the 4 points to form a 4 trajectories.
- In each frame iteration, we make sure to stock each newly calculated point in the corresponding trajectory. We were able to do this by calculating the quadratic distance between the current point and the last calculated point in the trajectory vectors, and taking the trajectory corresponding to the minimum distance.


Edge Detection : Binary image
8 connected Components Labelling
Centroids detection

Left Figure : Binary image resulting from the magnitude of the gradient

Middle Figure : The resulting image of the 8 connected Components Labeling

Right Figure : The centroids of each detected region in blue & the center of the interesting points in red (point 0 0 to be ignored)

Unfortunately, due to a limited time, trying to solve the problem using this method was found to be too difficult, so we implemented the method above.

Conclusion :

Cameras project 3D scenes onto 2D images. The pinhole camera model shows the relationship between the world coordinates and image coordinates using the intrinsic and extrinsic matrices that capture different characteristics of the camera.

If we want to find the relationship between the image coordinates and world coordinates of points that reside on the same plane (coplanar points), we can use homography estimation. The homography matrix can be estimated using two corresponding sets of points, one in world coordinates and the other in image coordinates.