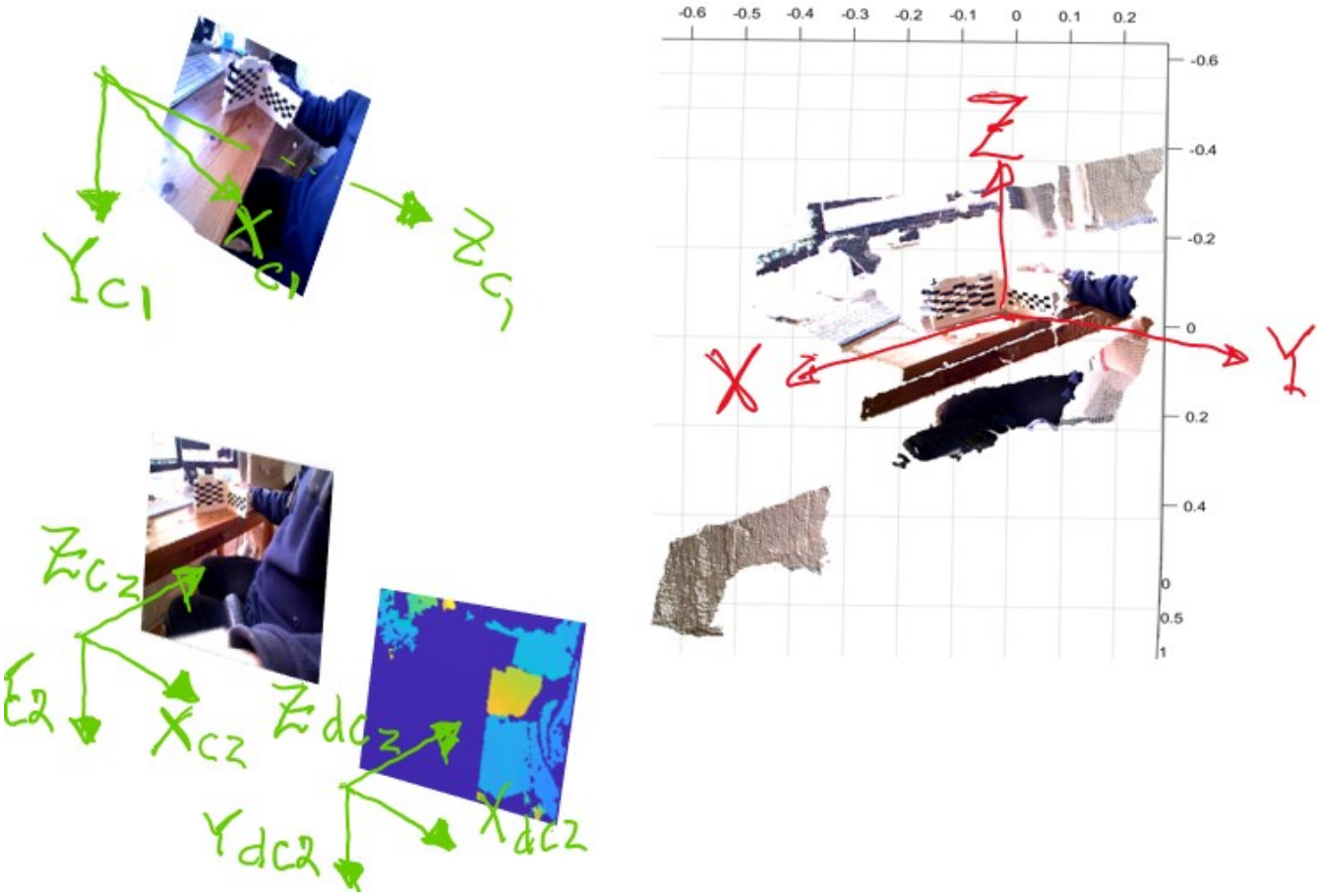
Class 9: From camera calibration to 3D registration

Objectives: Use camera model and rigid transformations to relate multiple images of the same static scene.

Two depth cameras capture both rgb and depth images of a scene as shown in the figure below:

The scene contains a special 3D object, the Lego 3D grid, of which we know its structure and dimensions. This object is described ahead.

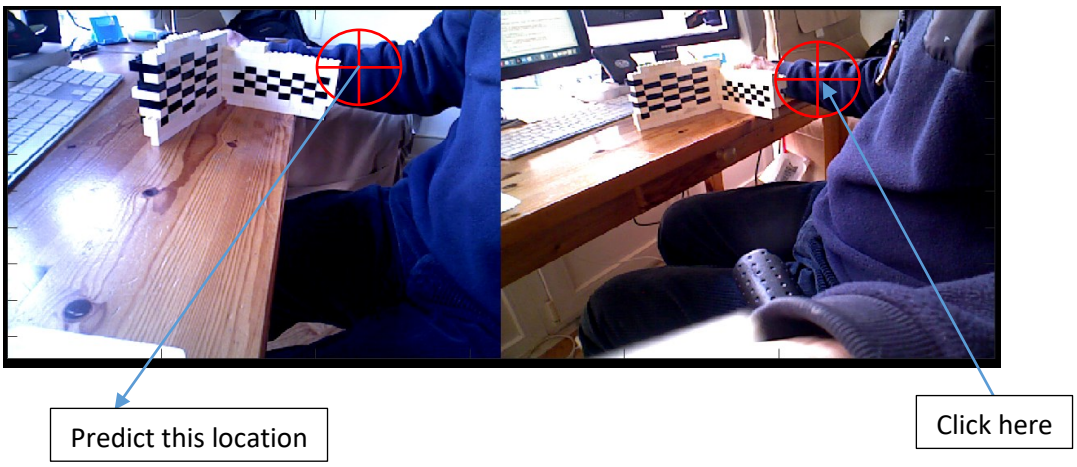
To better understand the data, refer to the figure below where we show both 3D point clouds (computed from one of the cameras) and some relevant coordinate frames. To avoid visual clutter we omit the depth image from camera 1 but both camera 1 and 2 have rgb and depth images.



Objectives for the class

With the data described before, write a small piece of code (matlab or python+opencv) that reads the coordinates of a point in the RGB image of camera 2 and predicts the position of that point in the RGB image of camera 1.

Schematically in the next figure we describe, pictorially, what is the objective of the code.



This task can be accomplished by many ways, that is, you have more data than you need to accomplish the goal. The dataset allows you to exercise the methods and models you have learned from class 1. Think about the problem, define all coordinate systems and transformations required and implement the estimation procedures, most notably DLT's rather than using matlab's or opencv's. Of course, after you do this, reimplement with the libraries and compare.

You may need to review:

Camera model (class 1 and 2) – how to compute depth from depth images and camera intrinsics

Camera calibration - How to compute camera matrix, intrinsic and extrinsic parameters. You may use the Lego object to do that.

Rigid transformations – Compute rigid transformations and its inverses between 3D reference frames.

On a second version of the problem, you may use 2 point clouds to compute the rigid transformations between the two cameras. This is related to the contents of theory class 6 (3d point cloud registration – the Procrustes Problem). In this case, because you have depth cameras, you must click on points in both rgb images, obtain their 3D coordinates (in the respective camera frames) and from these 3D pairs of points compute the rotation and translation between the two cameras.

Data available:

Files CalibrationScene_0000.mat-CalibrationScene_0014.mat hold a struct with the information of each image in the following fields:

pontos – rgb coordinates of grid points in Lego (concat of uvright and uvleft)
uvright – rgb coordinates of the right-side grid of Lego
uvleft - rgb coordinates of the left-side grid of Lego
XYZLeft, XYZRight – the 3D position of Lego points in Lego's frame
Image – the rgb image
Xyz – point cloud obtained from depth image.

```
struct with fields:

    pontos: [45×2 double]
   uvright: [21×2 double]
    uvleft: [24×2 double]
   XYZLeft: [24×3 double]
  XYZRight: [21×3 double]
     image: [480×640×3 uint8]
       xyz: [307200×3 double]
```

Calib_asus.mat – Calibration parameters of the depth camera

You only need this data to do the job. However we add extra code to help you doing it in case you did not do previous homeworks/PB classes.

Get_rgbd.m – function to register rgb image to depth image.

PIV_pcfromdepth.m – function to compute point cloud from depth image.

Qrcamera.m – QR decomposition to compute K, R and T from camera matrix (check camera calibration slides)