1. Table detection

In order to take detect the table, we have to take advantage of its features. The table edges are long straight lines. It is also blue and stands out from its environment. So our approach consists of finding the lines present in the input image and then filter them according to different criteria. After that, the lines are grouped depending on which side of the table they belong to. From this group of lines, the four sides of the table is calculated

1.1 Lines detection

Using Hough peaks and Hough lines, the algorithm start by detecting the lines that are present in the input image. We consider as a valid lines the ones that are on the edge of the table. The lines are therefore filtered on the following criteria:

Orientation

Given the point of view of the camera, we don't expect to have an edge of the table showing vertically on the image. Hough lines algorithm uses a polar coordinates system, therefore we filter the lines directly from θ of the polar coordinates.

Color separation

A valid line will have from one side the blue table color and then from the other side any other color (floor green, wall white…). To decide which lines are valid, two sample pixels are taken from each side of the line. The midpoint of the line along with the orthogonal line going through the midpoint are calculated. The sample pixels are equally distant points from the centroid on the orthogonal lines. The HSV values of these points are considered as it is a more robust way to deduce the color.

Given two points $p_1 = (h_1, s_1, v_1)$ and $p_2 = (h_2, s_2, v_2)$, a color is considered blue if

$$blue_1 = (h_1 > BLUEMIN) \wedge (h_1 < BLUEMAX) \wedge (h_1 > VLIGHTMIN)$$

where the thresholds (BLUEMIN, BLUEMAX and VLIGHTMIN) are defined from the hue values of the blue color. A minimum value of brightness is also imposed as blacks may be considered as blue.

A line is considered valid if

$$line = blue_1 \veebar blue_2$$

1.2 Lines grouping

After detecting the lines that are on the edge of the table, we need to extract the four lines that outline the shape of the table. We start by grouping the lines into four groups. Using the k-means clustering algorithm, the lines are separated into four groups. The algorithm uses squared euclidean distance.

Given that each line is represented by $ax+b$, we calculated the weighted average of the lines. We gave more weight to the longest lines as they are more accurate.

The fact the we get many lines for each of the edges of the table is due to the radial distortion induced by the wide lens camera.

Calculating the corner points is then straight forward, as they are the intersection of the edge lines.

2. Camera Calibration

2.1 Internal Parameters K

Knowing that the image of the absolute conic ω depends only on the internal parameters K of the camera matrix P, we can find K by calculating the image of the absolute conic ω.

In addition to that, we know that two vanishing points v_1 and v_2 that correspond to perpendicular directions satisfy the following property:

$v_1^T \omega v_2 = 0$

To generate enough constraints on ω, we need to find multiple vanishing points corresponding to perpendicular directions. After detecting lines using Hough Lines, we choose the perpendicular lines and calculate their intersection with the vanishing line.

The vanishing line is found by: intersecting parallel lines to find vanishing points and the fitting a line going through these vanishing points found before.

Using the Choleski factorization, we compute the matrix K from $\omega = (KK^T)^-1$

2.2 External Parameters

Using the homography that maps the table to real world coordinates and the internal parameters K we find the camera rotation and translation.

3D Reconstruction