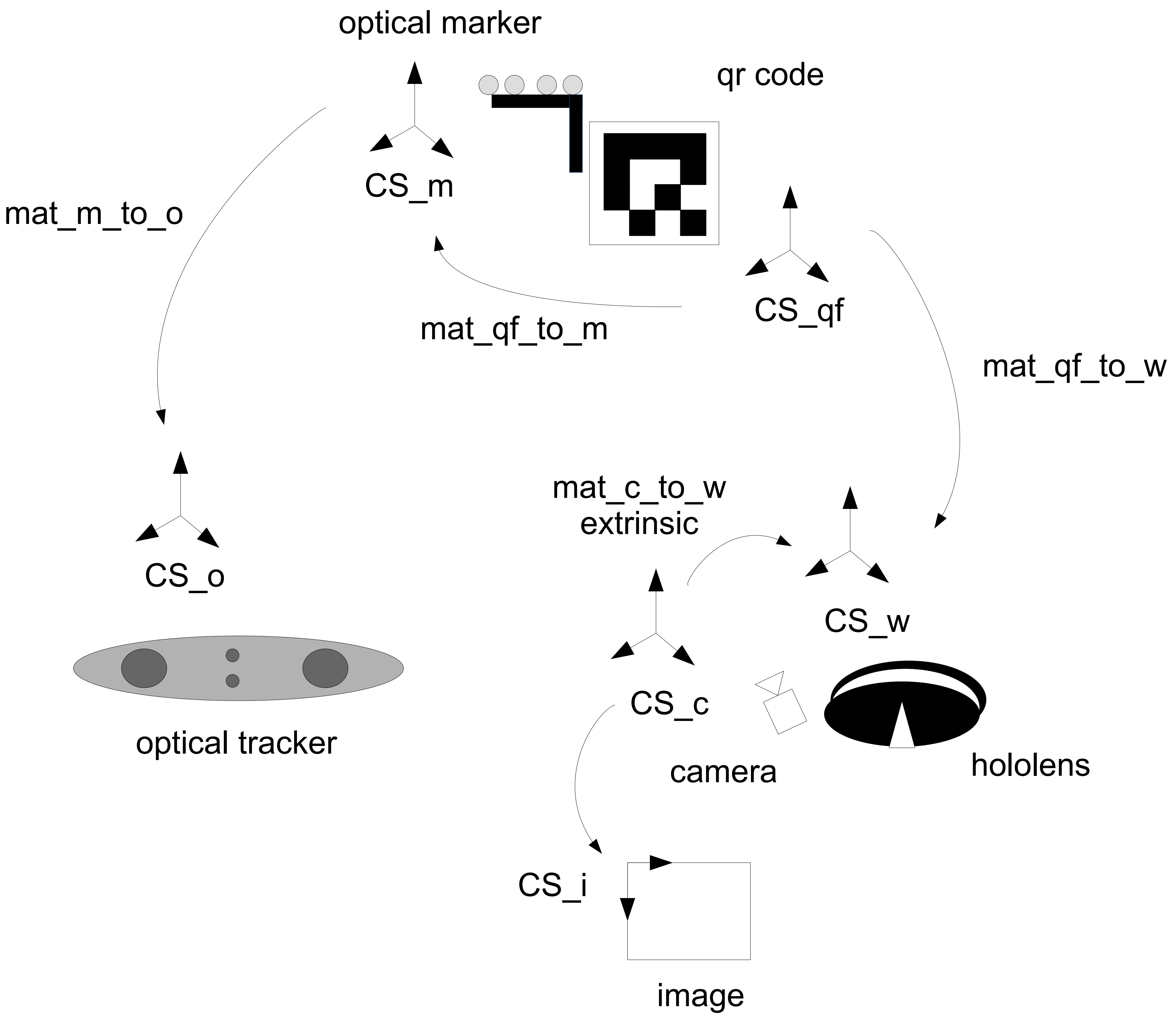


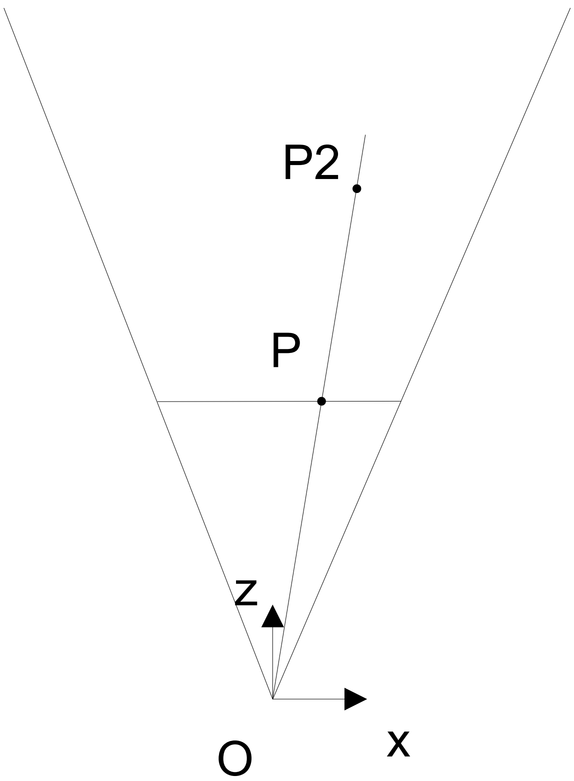
CS_x -> coordinate system of x
mat_a_to_b -> 4x4 transformation matrix to go from CS_a to CS_b

m is optical marker (centered in one of the spheres, look at the specs)
o is optical tracker (somewhere in the middle of the tracker)
qf is front qr code (in the middle of the qr code)
w is the world CS of the hololens (somewhere in the middle of the glasses)

c is one camera looking at z direction (could be following the context: pv (rgb camera), vl_front_left (grayscale front left camera), vl_front_right (grayscale front right camera), ahat (articulated hand tracking depth camera), lt (long throw depth camera)

CS_i for the 2D image coordinate system of the camera. No transformation matrix but the look-up-table (LUT) from 2D pixel to 3D position in CS_c (z=1) should be used.
get_lut_projection_pixel() function to get the 3D position in CS_c (z=1) from the 2D pixel in CS_i.
get_lut_pixel_image() function to get the 2D pixel in CS_i from the 3D position in CS_c.

projection vs unprojection



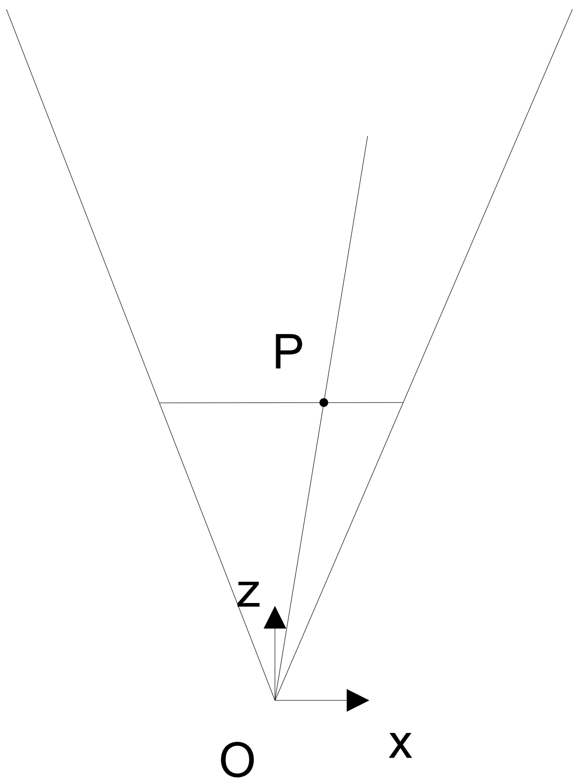
P = [P_x,P_y,1] 3D point in CS_c (O is center)
P2 = [P_x, P_y, P_z] possible real 3D position

$P2 = O + t.\overrightarrow{OP}$

Unprojection
P2_x = 0 + t.(P_x - 0) = t.P_x
P2_y = 0 + t.(P_y - 0) = t.P_y
P2_z = 0 + t.(P_z - 0) = t.P_z = t

Projection
P_x = P2_x/t
P_y = P2_y/t
P_z = P2_z/t = 1 → t = P2_z
→ P_x = P2_x/P2_z
→ P_y = P2_y/P2_z

depth sensor



P = [x,y,1]
d depth (in mm)

P2 = [d.x, d.y, d]
or ?
P2 = $[d.x/\|\overrightarrow{OP}\|,\ d.y/\|\overrightarrow{OP}\|,\ d/\|\overrightarrow{OP}\|]$