1 Homogeneous coordinates

To simplify the derivation of the geometric camera model, let us first introduce the concept of projective space with homogeneous coordinates. In Euclidean geometry, we define a point in 3- D space with its three coordinates, $X_E = [X, Y, Z]^T$. In projective space, we represent the same point with a 4-D homogeneous notation, $[X_1, X_2, X_3, X_4]^T$, so that the following relation holds: $X = [\frac{X_1}{X_4}, \frac{X_2}{X_4}, \frac{X_3}{X_4}, 1]^T$. In general, the following relation holds for the point expressed in Euclidean and homogeneous coordinates:

$$X_E = [X, Y, Z]^T \rightarrow X = \lambda \left[\frac{X_1}{X_4}, \frac{X_2}{X_4}, \frac{X_3}{X_4}, 1 \right]^T,$$

where λ is an arbitrary non-zero scale factor. In the projective space, all points that differ only in factor λ are considered equivalent.

2 Homography

Homographic transformation or homography is commonly used in projective geometry, as it represents a transformation between two planes in projective space. In the context of augmented reality, you can think of the homography as the transformation that projects a square marker board from the world coordinate system into a polygon in the image plane on the camera sensor (i.e., pixel coordinates).

Let $\{x_k^{(w)}\}_{k=1:4}$ denote the corner points of a square marker board in the world coordinate system, and let $\{x_k^{(c)}\}_{k=1:4}$ denote the corresponding projections of the corner points in the image coordinates. More specifically, $x_k^{(w)} = [x_k^{(w)}, y_k^{(w)}, 1]^T$, and $x_k^{(c)} = [x_k^{(c)}, y_k^{(c)}, 1]^T$. Then, we can project a point from world coordinates, $x_k^{(w)}$, to the point in the image plane, $x_k^{(c)}$, via homography H_w^c using the following equation:

$$x_k^{(c)} = \lambda H_w^c x_k^{(w)}, \tag{1}$$

where the homography H_w^c is a 3×3 matrix:

$$H_w^c = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{32} & h_{32} & h_{33} \end{bmatrix} \tag{2}$$

The relation given in Equation 1 is written using homogeneous coordinates, and is therefore defined only up to a scale, which we explicitly denoted by the scale factor λ. In practice, this means that the matrix $\boldsymbol{H}_w^c$ has only 8 degrees of freedom instead of 9, as we fix the value of the last element, $h_{33} = 1$.

$$\boldsymbol{x}_k^{(c)} \times \boldsymbol{H}_w^c \boldsymbol{x}_k^{(w)} = \mathbf{0}_{3\times 1}, \tag{3}$$

where $0_{3\times 1}$ denotes a column vector of zeros. Let us define the homography with its rows, $\boldsymbol{h}_i : \boldsymbol{H}_w^c = [\boldsymbol{h}_1^T, \boldsymbol{h}_2^T, \boldsymbol{h}_3^T]^T$. Substituting in Equation 3, we get:

$$\begin{bmatrix} \mathbf{0}^T & -\boldsymbol{x}_k^{(w)T} & y_k^{(c)}\boldsymbol{x}_k^{(w)T} \\ \boldsymbol{x}_k^{(w)T} & \mathbf{0}^T & -x_k^{(c)}\boldsymbol{x}_k^{(w)T} \\ -y_k^{(c)}\boldsymbol{x}_k^{(w)T} & x_k^{(c)}\boldsymbol{x}_k^{(w)T} & \mathbf{0}^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}_1^T \\ \boldsymbol{h}_2^T \\ \boldsymbol{h}_3^T \end{bmatrix} \tag{4}$$

The above system of equations contains only two linearly independent equations. Therefore, we can rearrange it into the following form:

$$\begin{bmatrix} x_k{}^{(w)} & y_k{}^{(w)} & 1 & 0 & 0 & 0 & -x_k{}^{(c)}x_k{}^{(w)} & -x_k{}^{(c)}y_k{}^{(w)} & -x_k{}^{(c)} \\ 0 & 0 & 0 & x_k{}^{(w)} & y_k{}^{(w)} & 1 & -y_k{}^{(c)}x_k{}^{(w)} & -y_k{}^{(c)}y_k{}^{(w)} & -y_k{}^{(c)} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1^T \\ \boldsymbol{h}_2^T \\ \boldsymbol{h}_3^T \end{bmatrix} = \mathbf{0}_{2\times 1}. \tag{5}$$

As we can see, each pair of corresponding points gives us two linearly-independent equations; given four pairs of corresponding points (where no subset of three points is collinear), we get eight linearly-independent equations. As mentioned before, a homography has eight degrees of freedom, and is therefore fully determined by at least four correspondences. We can write the problem of solving these eight equations in matrix form, by combining all equations in a matrix $\boldsymbol{A}$, and writing

$$\boldsymbol{A}\boldsymbol{h} = \mathbf{0}_{8\times 1} \tag{6}$$

$$\begin{bmatrix} x_1{}^{(w)} & y_1{}^{(w)} & 1 & 0 & 0 & 0 & -x_1{}^{(c)}x_1{}^{(w)} & -x_1{}^{(c)}y_1{}^{(w)} & -x_1{}^{(c)} \\ 0 & 0 & 0 & x_1{}^{(w)} & y_1{}^{(w)} & 1 & -y_1{}^{(c)}x_1{}^{(w)} & -y_1{}^{(c)}y_1{}^{(w)} & -y_1{}^{(c)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_4{}^{(w)} & y_4{}^{(w)} & 1 & 0 & 0 & 0 & -x_4{}^{(c)}x_4{}^{(w)} & -x_4{}^{(c)}y_4{}^{(w)} & -x_4{}^{(c)} \\ 0 & 0 & 0 & x_4{}^{(w)} & y_4{}^{(w)} & 1 & -y_4{}^{(c)}x_4{}^{(w)} & -y_4{}^{(c)}y_4{}^{(w)} & -y_4{}^{(c)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

When $\boldsymbol{A}$ happens to be a square matrix, we can find the exact solution to the problem. In cases when the system is over-determined (i.e., more than four correspondences are given), the matrix $\boldsymbol{A}$ is not square, and we can find the least-squares solution. The solution can be

efficiently found via the singular value decomposition (SVD) as the unit eigen-vector that corresponds to the ***smallest*** eigen-value of $\boldsymbol{A}$

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = \boldsymbol{U} \begin{bmatrix} \lambda_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_{99} \end{bmatrix} \begin{bmatrix} v_{11} & \dots & v_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \dots & v_{99} \end{bmatrix}^T \tag{8}$$

The elements of homography, $\boldsymbol{h}$, are obtained from the last column of $\boldsymbol{V}$; because we require that $h_{33} = 1$, we normalize with the value of $\boldsymbol{v_{99}}$:

$$\boldsymbol{h} = \frac{[v_{19}, \dots, v_{99}]^T}{v_{99}}. \tag{9}$$

To summarize, the main steps involved in estimating the homography are:

- Construct matrix $\boldsymbol{A}$ as described by Equation (6)
- Decompose matrix $\boldsymbol{A}$ via SVD: `[ U, S, V ] = svd(A)`.
- Obtain vector $\boldsymbol{h}$ as described in Equation (9)
- Rearrange the elements of h into a 3×3 matrix H using the function reshape, i.e., $\boldsymbol{H}$ = reshape$(\boldsymbol{h}, 3, 3)'$.

3 Geometric camera model for augmented reality

Suppose our camera is observing a point $\boldsymbol{x}^{(w)} = [x^{(w)}, y^{(w)}, z^{(w)}, 1]^T$, expressed in the (homogeneous) world coordinates, as illustrated by Figure 1. The coordinate system of camera $\boldsymbol{C}$ is rotated and translated with respect to the world coordinate system. In augmented reality, we wish to determine how the point $\boldsymbol{x}^{(w)}$ projects into the point $\boldsymbol{x}^{(c)} = [x_{(c)}, y^{(c)}, 1]^T$ lying in the camera's image plane, $\boldsymbol{\Pi}$. This relation is given by the projection matrix $\boldsymbol{P}$:

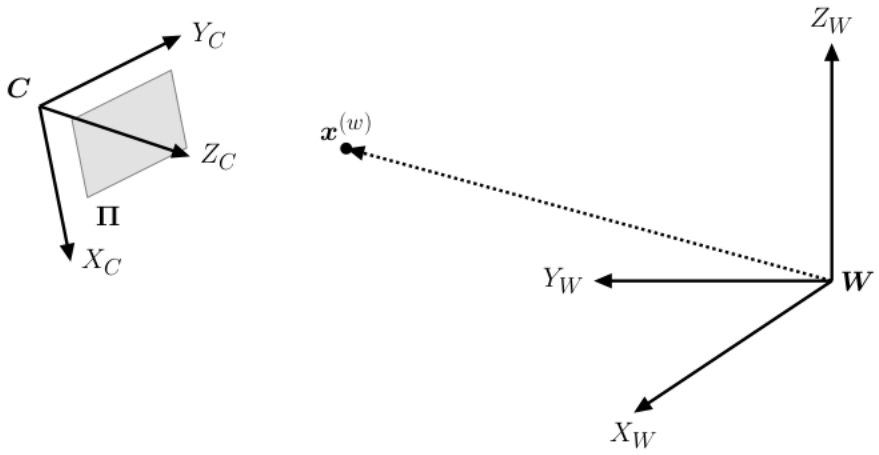

Figure 1: World coordinate system $\boldsymbol{W}$ and camera's coordinate system $\boldsymbol{C}$.

$$\boldsymbol{x}_{(c)} = \boldsymbol{P}\boldsymbol{x}^{(w)}. \tag{10}$$

The projection matrix $\boldsymbol{P}$ can be written as a product of two matrices, $\boldsymbol{K}$ and $[\boldsymbol{R} \mid \boldsymbol{t}]$,

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R} \mid \boldsymbol{t}]. \tag{11}$$

Matrix $\boldsymbol{K}$ is called intrinsic camera matrix. It contains the camera's intrinsic parameters, which are estimated during camera calibration. This matrix represents the transformation of a 3-D point from the camera's coordinate system to the 2-D point in the camera's image plane (pixel coordinates). Matrix $[\boldsymbol{R} \mid \boldsymbol{t}]$ consists of rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$. These are the camera's extrinsic parameters, which describe the transformation of a 3-D point from the world coordinate system to the 3-D point in the camera's coordinate system.

In augmented reality applications, the camera is typically moving, which means that the projection matrix $\boldsymbol{P}$ keeps changing. Intrinsic parameters of the camera typically do not change, hence the matrix $\boldsymbol{K}$ is constant, and can be determined in advance via a camera calibration procedure. On the other hand, the camera's pose — rotation and translation with respect to the world coordinate system – keeps changing. Therefore, the main problem in an augmented reality application is a robust estimation of the camera's pose (rotation $\boldsymbol{R}$ and translation t) with respect to the reference object (marker), which define the origin of the world coordinate system. An example is shown in Figure 2. In this example, the world coordinate system is fixed to the plane of the marker; consequently, the Z coordinate for every point on the marker plane equals to 0 in the world coordinate system.

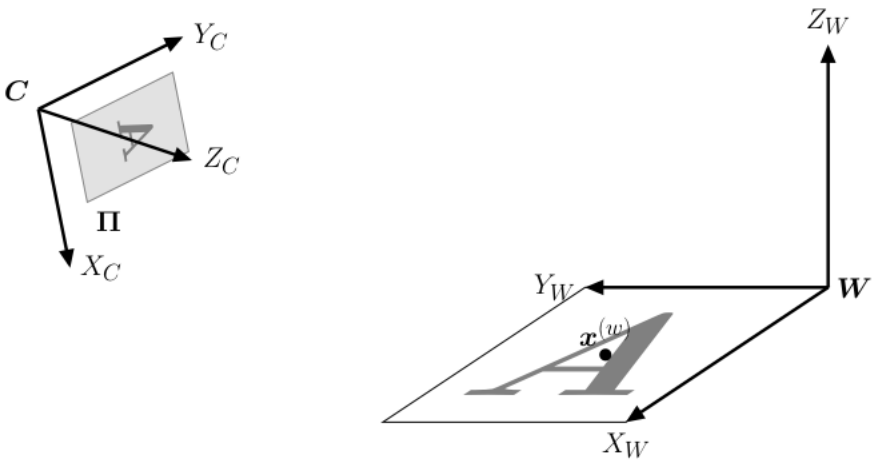

Figure 2: text

3.1 Estimation of the camera's pose with respect to a planar marker

Let us consider the transformation of an arbitrary point from a planar marker, $\boldsymbol{x}^w = [x^{(w)}, y^{(w)}, 0, 1]^T$, to the point in the camera's image plane, $\boldsymbol{x}^{(c)} = [x^{(c)}, y^{(c)}, 1]^T$. For easier analysis, we will write the rotation matrix using its columns, $\boldsymbol{r_i} : \boldsymbol{R} = [\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}]$. Then, we can write $[\boldsymbol{R} \mid \boldsymbol{t}] = [\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}, \boldsymbol{t}]$, and equation becomes:

$$\begin{aligned}
\boldsymbol{x}^{(c)} &= \boldsymbol{P}\boldsymbol{x}^{(w)} \\
\boldsymbol{x}^{(c)} &= \boldsymbol{K}[\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}, \boldsymbol{t}][x^{(w)}, y^{(w)}, 0, 1]^T \\
\boldsymbol{x}^{(c)} &= \boldsymbol{K}[\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{t}][x^{(w)}, y^{(w)}, 1]^T \\
\boldsymbol{x}^{(c)} &= \boldsymbol{H}[x^{(w)}, y^{(w)}, 1]^T,
\end{aligned} \tag{12}$$

where we defined $\boldsymbol{H} = \boldsymbol{K}[\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{t}] = [\boldsymbol{h_1}, \boldsymbol{h_2}, \boldsymbol{h_3}]$. From our problem formulation (Figure 3), we know that $\boldsymbol{H}$ is transforming points between two planes (i.e., the planar marker and the image plane). Therefore, transformation $\boldsymbol{H}$ is a homography, and we can estimate it from correspondences between the known model of our planar marker and the marker's detection in the image. Suppose that the set of four points $\{\boldsymbol{x}_k^{(w)}\}_{k=1:4}$ denotes the marker's corners, detected in the image, while points $\{\boldsymbol{x}_k^{(w)}\}_{k=1:4}$ denote the corresponding corners in the model of the marker (see Figure 3). Using these two sets of corresponding points, we can use the procedure from Section 2 to estimate the unknown homography $\boldsymbol{H}$.

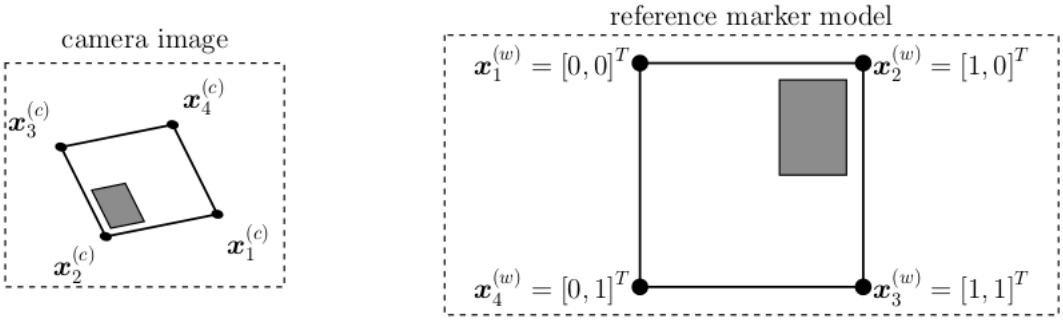

Figure 3: Left image shows an example of marker's corner detections in the image (pixel coordinates), while the right image shows the same corners, expressed in the marker's coordinate system (we chose the marker's sides to be of unit lengths).

Once we have computed the homography $\boldsymbol{H}$, we need to derive the rotation matrix $\boldsymbol{R}$ and translation vector t from it. Let us define a new matrix $\boldsymbol{B} = [\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{t}]$. Considering equation 11, we can write:

$$\lambda \boldsymbol{H} = \boldsymbol{K} \widetilde{\boldsymbol{B}} \tag{13}$$

$$\widetilde{\boldsymbol{B}} = \lambda \boldsymbol{K^{-1}} \boldsymbol{H} \tag{14}$$

$$\boldsymbol{B} = \lambda \widetilde{\boldsymbol{B}} (-1)^{|\widetilde{\boldsymbol{B}}|<0}, \tag{15}$$

where we used scale factor λ to stress that the scale needs to be explicitly computed. The term $(-1)^{|\widetilde{\boldsymbol{B}}|<0}$ means that if the computed $\widetilde{\boldsymbol{B}}$ has negative determinant, we multiply it by -1. The reason for this is that we obtain two possible solutions for the camera pose; one assumes that the object is located behind the camera ($\boldsymbol{B}$ has positive determinant), while the other assumes that the object is located behind the camera ($\boldsymbol{B}$ has negative determinant); we pick a solution with the object located in front of the camera, hence $\boldsymbol{B}$ needs to have a positive determinant.

We write the matrix $\boldsymbol{B}$ using its columns, $\boldsymbol{b}_i$: $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3]$. The first column, $\boldsymbol{b}_1$, is proportional to the first column of the rotation matrix, $\boldsymbol{r}_1$; the second column, $\boldsymbol{b}_2$, is proportional to the second column of the rotation matrix, $\boldsymbol{r}_2$, while the third one, $\boldsymbol{b}_3$, is proportional to the translation vector $\boldsymbol{t}$. We know that the rotation matrix is orthonormal; the last column must be perpendicular to the first two, hence we can compute it as the vector (cross) product of the first two columns. At the same time, all three columns of rotation matrix need to be of unit length. However, in practice, the estimated columns $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are not unit vectors, and we need to estimate their scale – the λ paramter.

We estimate λ as the average length of the first two columns of $\boldsymbol{B}$, using the following equation:

$$\lambda = \left(\frac{||\boldsymbol{K}^{-1}\boldsymbol{h}_1|| + ||\boldsymbol{K}^{-1}\boldsymbol{h}_2||}{2} \right)^{-1} \tag{16}$$

Now we can compute all elements of the rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$:

$$\boldsymbol{r}_1 = \lambda \boldsymbol{b}_1 \, , \boldsymbol{r}_2 = \lambda \boldsymbol{b}_2 \, , \boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2 \, , \boldsymbol{t} = \lambda \boldsymbol{b}_3. \tag{17}$$

In practice, we would typically perform a non-linear optimization of $[\boldsymbol{R} \mid \boldsymbol{t}]$ matrix parameters as the final step.

The procedure for estimating camera's pose and the projection model therefore consists of the following steps:

- Detect four corners of the marker in the input image captured by camera (the true corners for the UPRIGHT orientation of marker).
- Determine the world coordinates of the corners, as shown in Figure, and compute the homography between the detected corner points in the image (pixel coordinates) and the corresponding points in the reference (world) coordinate system.
- Use Equations 14-17 to determine the camera's pose – the rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$.
- Constuct projection matrix, $\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R} \mid \boldsymbol{t}]$.
- You can now project any point from the world coordinate system to the image plane, using Equation 11. For example, consider a corner of a cube that lies aligned on top of the marker. We fixed the origin of the world coordinate system in the top-left corner of the marker, with the Z axis pointing downwards; hence, a corner on the top face of the cube above the origin will be written in homogeneous coordinates as $\boldsymbol{x}^{(w)} = [0, 0, -1, 1]^T$. This point can be easily projected into image plane as $\boldsymbol{x}^{(c)} = \boldsymbol{P}\boldsymbol{x}^{(w)}$; keep in mind that the point $\boldsymbol{x}^{(c)}$ is expressed in homogeneous coordinates, and its last element needs to be 1. If it is not, the vector $\boldsymbol{x}^{(c)}$ needs to be divided by its last element!