3. `find_3d_points.py`: The goal is the to find a 3D point $\mathbf{X} = (X, Y, Z, W)$ minimizes the reconstruction error defined as

$$\epsilon^2 = ||\mathbf{x} - \hat{\mathbf{x}}||^2.$$

We define $\mathbf{x} = (x_1, y_1, x_2, y_2)^T$ as the given coordinates of two 2D points in the two images. The vector $\hat{\mathbf{x}} = (\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2)^T$ represents the projections of of $\mathbf{X}$ onto each of the two images. The 2D projections in homogenous coordinates are given by the respective camera matrices P_i:

$$\tilde{\mathbf{x}}_i = P_i\mathbf{X}.$$

In inhomogenous coordinates,

$$\hat{\mathbf{x}}_i = \tilde{\mathbf{x}}_{i[:2]}/\tilde{\mathbf{x}}_{i3} = \frac{P_{i[:2]}\mathbf{X}}{P_{i3}\mathbf{X}}.$$

Here, $A_{[:2]}$ refers to the first two rows of a matrix A (notation borrowed from Python). For each image i, we are essentially solving for

$$0 = \frac{P_{i[:2]}\mathbf{X}}{P_{i3}\mathbf{X}} - \hat{\mathbf{x}}_i.$$

This is equivalent to solving for

$$P_{i[:2]}\mathbf{X} - \hat{\mathbf{x}}_i P_{i3}\mathbf{X} = (P_{i[:2]} - \hat{\mathbf{x}}_i \otimes P_3)\mathbf{X} \equiv A_i\mathbf{X} = 0.$$

We can stack A_1 and A_2 for each of the two matrices to get

$$A = (A_1^T, A_2^T)^T$$

and solve for

$$A\mathbf{X} = 0.$$

The (approximate) solution to this homogenous equation can be found by SVD. $\mathbf{X}$ will be the right singular vector corresponding to the smallest singular value of A. Note that the error for image i will in fact be

$$\frac{A_i\mathbf{X}}{P_{i3}\mathbf{X}}$$

and not

$$A_i\mathbf{X}.$$

Therefore, the errors of the two images will not be weighted equally. As the equations to be minimized are manifestly non-linear, it would be fairly difficult to exactly minimize the total error, weighted equally.