So basically we are given points in the real world frame and the camera parameters and we want to project the real coordinate point from the real world to image points (image frame)

Steps

1. Initialize the cube coordinates
2a. Read intrinsic parameters file K
2b. Read distortion parameters file D (optional depending if we are doing on distorted or non distorted image)
3. Now loop through all the image files in images or images_undistorted (depending whether we want distorted images or not and if we have read D)
4. For the corresponding image read the pose file ->
From the pose the first 3 points are rotation w and the next 3 are translational points t
K = w / theta where theta is the **norm of w. Theta = ||W||**
And then we can get our homogenous transform matrix

Cause we have R from the image and just append t

2.1.2 Axis-angle representation for rotations

In this exercise, the rotation R from the world frame to the camera frame is given using the axis-angle representation for rotations. Specifically, a 3D rotation is parameterized by a 3D vector $\omega = (\omega_x, \omega_y, \omega_z)^T$, where $\mathbf{k} = \frac{\omega}{\|\omega\|}$ is a unit vector indicating the axis of rotation, and $\|\omega\| = \theta$ is the magnitude of the rotation about the axis. Rodrigues' rotation formula allows to convert this representation to a rotation matrix:

$$R = I + (\sin\theta)[\mathbf{k}]_\times + (1 - \cos\theta)\,[\mathbf{k}]_\times^2$$

where $[\mathbf{k}]_\times = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$ is the *cross-product matrix* for the vector $\mathbf{k}$.

5. Now we have the real world points and we have calculated our transform matrix. All we need to do is just applying the formula below.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R \mid T \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad \text{where} \quad K = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

points_chessboard.py:

1. RGB2GRAY
2. We use Rodrigues' rotation formula to convert the rotation vector into a rotation matrix.
3. We apply the distortion formula, taking into account the radial distortion parameters D and the distance of the points from the optical center.
4. First we normalize, then we apply the projection transformation by multiplying the normalized points with the intrinsic matrix K. Finally, function that we created scales the projected points by dividing by the third coordinate to obtain the 2D image coordinates.
5. Last function performs undistortion on the distorted image using the distortion coefficients D.

Ressources : https://www.youtube.com/watch?v=uHApDqH-8UE

https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

https://www.mathworks.com/help/vision/ug/camera-calibration.html

https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec

https://ftp.cs.toronto.edu/pub/psala/VM/camera-parameters.pdf

some videos in this channel : First Principles of Computer Vision

https://www.youtube.com/playlist?list=PL2zRqk16wsdoCCLpou-dGo7QQNks1Ppzo

https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-for-camera-calibration-in-computer-vision/

Udacity Computer Vision Exercises