HW1

Problem 1

(3 points)

Suppose a pinhole camera has a camera intrinsic matrix K. Let the camera extrinsics be a 3D rotation R, and a 3D translation $\mathbf{t}$. Given a pixel $(x, y)^T$ in an image, assume the depth of the pixel is d, where depth is the distance between the 3D point of pixel and the camera center. Compute the coordinates of the 3D point in the **world coordinate system**.

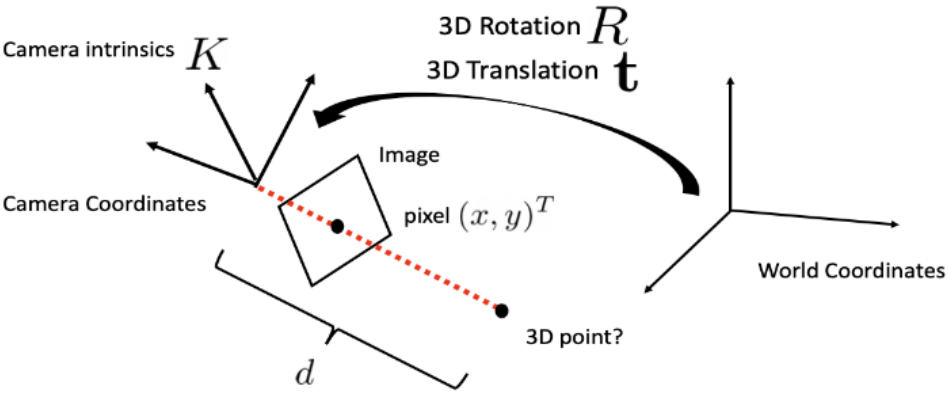

Figure 1: Backprojection of a pixel.

Camera Intrinsics:

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad K^{-1} = \begin{bmatrix} \frac{1}{\alpha_x} & 0 & \frac{-x_0}{\alpha_x} \\ 0 & \frac{1}{\alpha_y} & \frac{-y_0}{\alpha_y} \\ 0 & 0 & 1 \end{bmatrix}$$

Normalized Image Coordinates: N → multiply pixel coordinates: P with K^{-1}:

$$N = PK^{-1}$$

$$\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\alpha_x} & 0 & \frac{-x_0}{\alpha_x} \\ 0 & \frac{1}{\alpha_y} & \frac{-y_0}{\alpha_y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{x_p - x_0}{\alpha_x} \\ \frac{y_p - y_0}{\alpha_y} \\ 1 \end{bmatrix}$$

3D Camera Coordinates : C → Depth (d) multiplied with the Normalized image coordinates (N)

$$C = dN$$

$$C = d\begin{bmatrix}\frac{x_p - x_0}{\alpha_x}\\ \frac{y_p - y_0}{\alpha_y}\\ 1\end{bmatrix} = \begin{bmatrix}d\frac{x_p - x_0}{\alpha_x}\\ d\frac{y_p - y_0}{\alpha_y}\\ d\end{bmatrix}$$

Reverting Transformation of R and t from Camera Coordinates: C back to World Coordinates: W

if $WR + t = C \therefore W = CR^{-1} - t$

$$\begin{bmatrix}x_w\\ y_w\\ w_w\end{bmatrix} = R^{-1}\begin{bmatrix}d\frac{x_p - x_0}{\alpha_x}\\ d\frac{y_p - y_0}{\alpha_y}\\ d\end{bmatrix} - t$$

Problem 2

```
#TODO: implement this function: for each 2D coordinate x, its
def backproject(depth, intrinsic_matrix):
    # Get the height (H) and width (W) of the depth map
    H, W = depth.shape

    # Initialize a point cloud array with zeros, having the s
    pcloud = np.zeros((H, W, 3))

    # Loop through each pixel in the depth map
    for i in range(H):
        for j in range(W):
            # Create a homogeneous coordinate for the current
            x = np.array([j, i, 1])

            # Get the depth value at the current pixel
            d = depth[i, j]

            # Compute the 3D point by multiplying the depth v
            X = d * np.linalg.inv(intrinsic_matrix) @ x

            # Store the computed 3D point in the point cloud
            pcloud[i, j] = X
```

```
    return pcloud
```