FSG 2023 - Quiz 5

Javier Montané Ortuño

Since distortion is not considered, we apply the pinhole camera model. First, we build the 2D homogeneous coordinates of the object's center. Then, we use the intrinsic parameters, specifically the camera matrix, to obtain 3D homogeneous coordinates in the camera frame. After scaling, we get the Euclidean 3D coordinates. Lastly, we employ the extrinsic parameters, involving rotation and translation, to transform back and determine the 3D coordinates in the vehicle coordinate frame.

Data given

We know the **intrinsic parameters** of the camera:

$$\begin{aligned} \text{Focal Lengths:} &\quad fx = 241, fy = 238 \text{ (in pixels per unit length)} \\ \text{Principal Point:} &\quad cx = 636, cy = 548 \text{ (in pixels)} \end{aligned}$$

As well as the **extrinsic parameters**:

$$\begin{aligned} \text{Traslation of camera:} &\quad x = 500, y = 160, z = 1140 \text{ (in milimeters)} \\ \text{Rotation of camera (Euler Angles):} &\quad \text{roll} = 100^\circ \quad \text{pitch} = 0^\circ \quad \text{yaw} = 90^\circ \end{aligned}$$

And finally the distance between the object and the camera frame ($d = 2.7$m) and the pixel coordinates of the object center:

$$(u, v) = (795, 467) \quad \text{(in pixels)}$$

Steps

Here are the steps for the 2D to 3D transformation using the camera matrix:

1. **Correct for Principal Point Offset**:

$$\begin{aligned} \text{Corrected}_u &= u - cx \\ \text{Corrected}_v &= v - cy \end{aligned}$$

2. **Normalize Coordinates**:

$$\begin{aligned} \text{Normalized}_x &= \frac{\text{Corrected}_u}{fx} \\ \text{Normalized}_y &= \frac{\text{Corrected}_v}{fy} \end{aligned}$$

3. **Construct 2D Homogeneous Coordinates**:

$$\text{Homogeneous\_2D} = [\text{Normalized}_x, \text{Normalized}_y, 1]$$

4. **Construct the Camera Matrix**:

$$K = \begin{bmatrix} fx & 0 & -cx \\ 0 & fy & -cy \\ 0 & 0 & 1 \end{bmatrix}$$

5. **Perform 2D to 3D Transformation**: To convert to a 3D homogeneous coordinate in the camera frame:

$$\text{Homogeneous\_3D\_in\_camera\_frame} = K^{-1} \times \text{Homogeneous\_2D}$$

6. **Apply Distance Scaling**: Scale the 3D point based on the object's distance d:

$$\text{Euclid\_3D\_Cam} = d * \text{Homogeneous\_3D\_in\_camera\_frame}$$

7. **Convert Euler Angles to Rotation Matrix and multiply it**: You can find the matrix in *Wikipedia*, I am not writing that in LaTeX. Also, since we want the inverse and it is orthonormal, we can simply take the transpose.

$$\text{Inv\_Rotated} = R * \text{Euclid\_3D\_Cam}$$

8. **Take into account the traslation as well**: Simply, add the traslation vector to the inverse rotated one and those are the 3D coordinates of the object in the vehicle coordinate frame. And the coordinates in meters are:

$$x = 6.728, y = 4.057, z = 7.695$$

Code implementation.

```
import numpy as np
import math
#-----------------------------------------
#DECLARING DATA VARIABLES:

# Intrinsic Parameters
fx = 241  # Focal length in the x direction
fy = 238  # Focal length in the y direction
cx = 636  # Principal point's x coordinate
cy = 548  # Principal point's y coordinate

# Object's Distance (in m) from the Camera Frame
d = 2.7

# Pixel coordinate's of the object's center
u=795
v=467

# Euler Angles in radians
roll = math.radians(100)
pitch = math.radians(0)
yaw = math.radians(90)

#Define the rotation matrice and get the inverse
R = np.array([[math.cos(yaw)*math.cos(pitch), -math.sin(yaw)*math.cos(roll) + ma
               [math.sin(yaw)*math.cos(pitch), math.cos(yaw)*math.cos(roll) + mat
               [-math.sin(pitch), math.cos(pitch)*math.sin(roll), math.cos(pitch)
R=R.T #It is orthonormal, the traspose is the inverse as well

#Traslation vector in meters:
tras=np.array([0.5,0.16,1.140])
#-----------------------------------------
# CALCULATIONS:

# Normalized 2D Homogeneous Coordinates
Norm_x = (u - cx)/fx
Norm_y = (u - cy)/fy

# Construct 2D Homogeneous Coordinate
Homo_2D_Cam = np.array([Norm_x, Norm_y, 1])

# Construct the Camera Matrix
K = np.array([[fx, 0, -cx],
```

```
                [0, fy, -cy],
                [0, 0, 1]])

# Perform 2D to 3D Transformation in camera frame
Homo_3D_Cam = np.linalg.inv(K).dot(Homo_2D_Cam)

# Apply Distance Scaling
Euclid_3D_Cam = Homo_3D_Cam * d

#Apply inverse rotations:
Inv_Rot = np.dot(R, Euclid_3D_Cam)

#Apply the traslation
Euclid_3D_Veh=Inv_Rot+tras

#Print the result
x, y, z = Euclid_3D_Veh
print(f"3D Point in Vehicle Frame (meters): x={x}, y={y}, z={z}")
```