CS 6334.001 Virtual Reality Homework 3

Professor Yu Xiang

October 14, 2021

Problem 1

(2 points)

Gyro integration.

Suppose at time t, the angular velocity of a rigid body sensed from a gyro is $\hat{\omega}_t$, which is a 3 dimensional vector of angular velocity in x, y, z direction in the local coordinates of the rigid body. Given the estimated orientation of the rigid body at time $t - 1$ with respect to a fixed world frame as a quaternion $\hat{q}_{t-1}$ and the time interval between two time steps as Δt, estimate the orientation $\hat{q}_t$ by gyro integration using $\hat{\omega}_t$, $\hat{q}_{t-1}$ and Δt.

(Hint 1) Let $q(\mathbf{v}, \theta)$ be a quaternion obtained by the axis-angle representation of 3D rotation, i.e., rotating the vector $\mathbf{v}$ by the angle θ. Figure out what are $\mathbf{v}$ and θ in this problem.

(Hint 2) Given two quaternions q_1 and q_2, the combined quaternion of rotation by q_1 followed by q_2 is $q = q_2 q_1$ with quaternion multiplication.

Problem 2

(2 points)

Tilt correction with accelerometers.

Suppose at time t, the estimated orientation of a rigid body with respect to a fixed world frame by gyro integration is a quaternion $\hat{q}_t$. Let $\hat{a}_t$ be the sensed linear acceleration from an accelerometer on the rigid body. We assume that there is no external acceleration. Therefore, the sensed linear acceleration corresponds to gravity. Express the angle ϕ between the sensed acceleration and the gravity vector $\mathbf{y} = (0, 1, 0)^T$ in the world frame using $\hat{q}_t$ and $\hat{a}_t$. See Figure 1.

(Hint 1) The linear acceleration from the accelerometer is in the local coordinates of the rigid body.

(Hint 2) To rotate a vector $\mathbf{v}$ by a quaternion q, we have $\mathbf{v}' = q\mathbf{v}q^{-1}$ with quaternion multiplication and quaternion inverse. We assume all quaternions are with unit length.

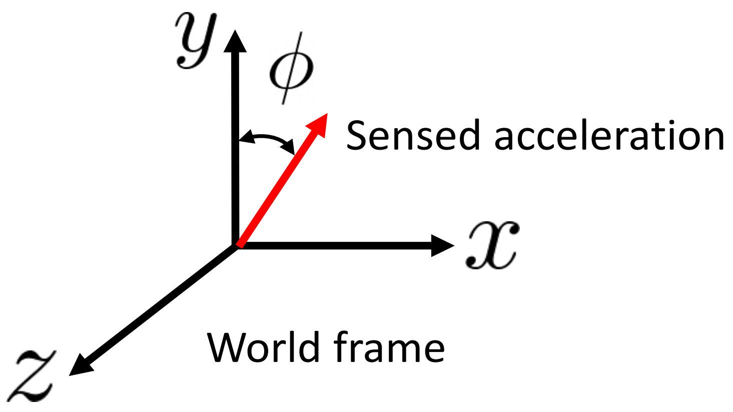

Figure 1: Illustration of the sensed acceleration from an accelerometer in the world frame.

Problem 3

(2 points)

EPnP.

Let $\mathbf{c}_1^c, \mathbf{c}_2^c, \mathbf{c}_3^c, \mathbf{c}_4^c$ be four control points in the camera frame. Then for the ith 3D point $\mathbf{p}_i^c$, $i = 1, 2, \ldots, n$ in the camera frame, we have

$$\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c, \tag{3.1}$$

where $0 \le \alpha_{ij} \le 1$ and $\sum_{j=1}^{4} \alpha_{ij} = 1$. According to the camera projection geometry, we have

$$w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix}, \tag{3.2}$$

where (u_i, v_i) are the pixel coordinates of the 3D point $\mathbf{p}_i^c$, w_i is an unknown variable, (f_u, f_v) are the focus lengths, (u_c, v_c) is the principal point, and $\mathbf{c}_j^c = (x_j^c, y_j^c, z_j^c)^T$ for $j = 1, \ldots, 4$.

(1) Show that

$$\begin{aligned} \sum_{j=1}^{4} \left[\alpha_{ij} f_u x_j^c + \alpha_{ij}(u_c - u_i) z_j^c \right] &= 0 \\ \sum_{j=1}^{4} \left[\alpha_{ij} f_v y_j^c + \alpha_{ij}(v_c - v_i) z_j^c \right] &= 0 \end{aligned} \tag{3.3}$$

(2) Let

$$\mathbf{x} = \begin{bmatrix} x_1^c \\ y_1^c \\ z_1^c \\ x_2^c \\ y_2^c \\ z_2^c \\ x_3^c \\ y_3^c \\ z_3^c \\ x_4^c \\ y_4^c \\ z_4^c \end{bmatrix} \tag{3.4}$$

be a 12×1 vector of all the control points. We can write down Eq. (3.3) as

$$M\mathbf{x} = 0, \tag{3.5}$$

where M is a $2n \times 12$ matrix, where n is the number of 3D points. Write down the $(2i-1)$th row and $(2i)$th row of M that correspond to the ith 3D point $\mathbf{p}_i^c$, $i = 1, 2, \ldots, n$.

Problem 4

(4 points)

Download the homework3_programming.zip file from eLearning, Assignments, Homework 3. Implement using PnP methods from OpenCV to compute the camera pose in the get_observation() function in table_scene_pnp.py. You need to install cv2 and scipy in python.

In this example, we have a flat plate on a table as shown in Figure 2. We can use Harris corner detection to detect the four corners of the plate on the image. Then we can apply PnP methods to compute the camera pose with respect to the plate. Here, the 3D coordinates of the four corners are (-0.05, -0.05, 0), (0.05, -0.05, 0), (0.05, 0.05, 0) and (-0.05, 0.05, 0).

After your implementation, run the table_scene_pnp.py in Python. The script should print the required information according to the instructions in the code. Submit your script to eLearning, and TA will run your script to verify it.

Here are some useful resources:

- Python basics `https://pythonbasics.org/`
- PyBullet `https://pybullet.org/wordpress/`
- Numpy `https://numpy.org/doc/stable/user/basics.html`
- PnP in OpenCV `https://www.pythonpool.com/opencv-solvepnp/`

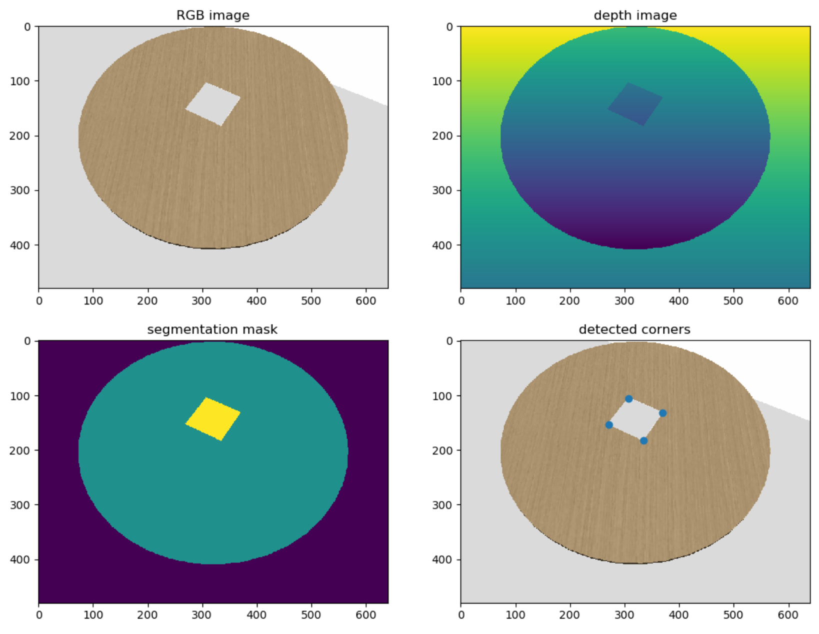

Figure 2: Visualization of the images and the corner detection in the programming code.