CMPE 161 Assignment 4 Report

Vikrant More (Student ID: 1501413)
Bharath Nagesh (Student ID: 1503735)

1. Differences between the 3 modalities for calculating R_0^t :

1.1. Rodriguez Formula :

The Rodriguez formula calculations for R_0^t requires us to first obtain the Omega(ω) vector[units : rad/s] from the system. Once this is obtained we need to multiply the vector with time to get the Omega(Ω) vector [units : rad]. Using the Ω vector we form the skew symmetric matrix $[\Omega]_X$.

1.1.1. Full - Rodriguez Formula :

Using the Ω vector we first calculate the magnitude of the vector i.e. $\|\Omega\|$. This gives us the angle (θ). Then we calculate the normalized Ω vector (n). Using the 'n' vector we form the skew symmetric matrix $[n]_X$. We then use the formula

$$R(n,\theta) = I - [n]_X\ (\sin(\theta)) + [n]_X^2\,(1 - \cos(\theta))$$

1.1.2. Rodriguez Formula small angle approximation :

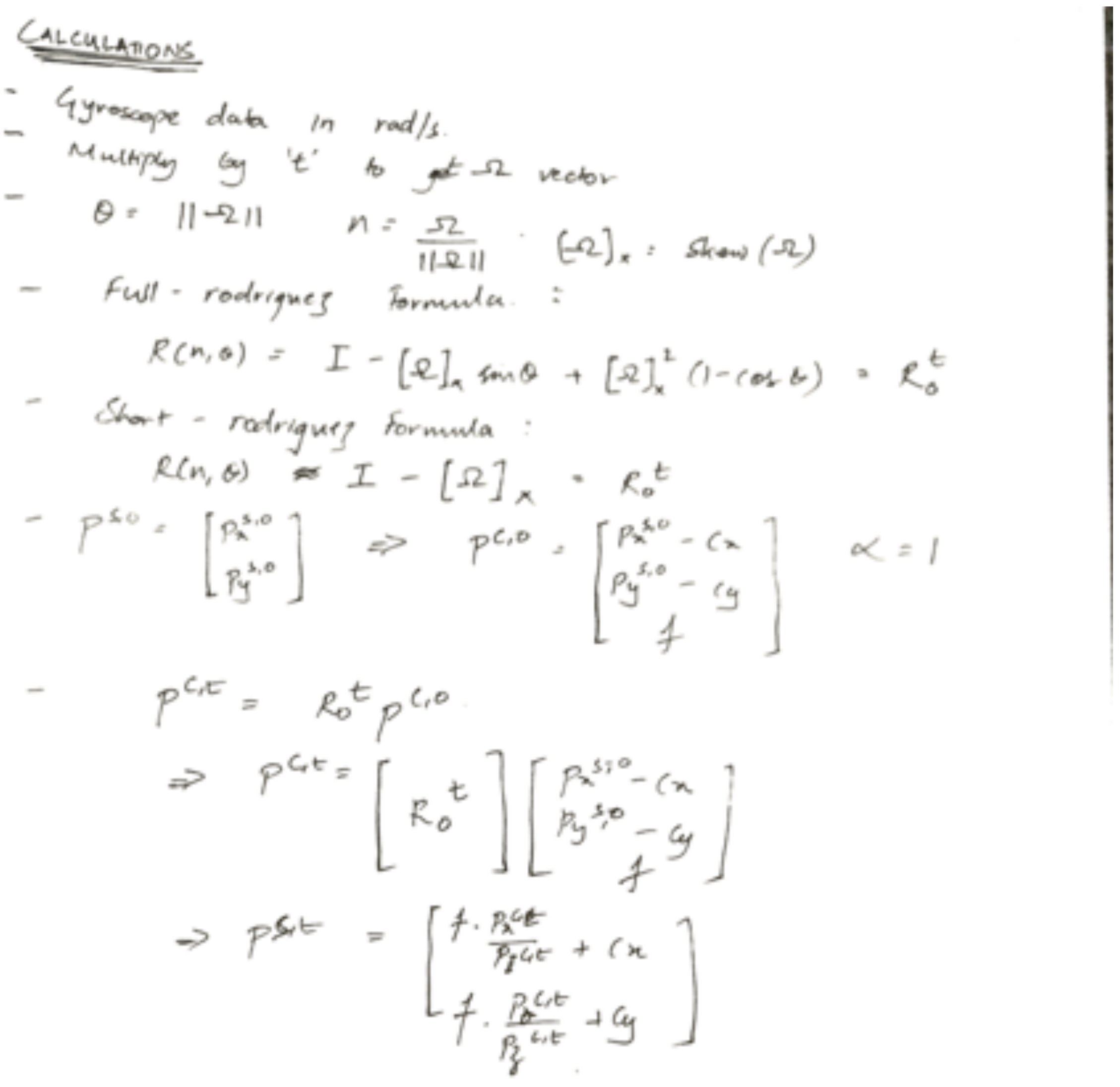
When we apply small angle approximation. We form the skew-symmetric matrix with the Ω vector $[\Omega]_X$. We then use the formula.

$$R(n,\theta) = I - [\Omega]_X$$

1.2. Rotation Matrix obtained from CMDeviceMotion :

In this modality we obtain the Rotation Matrix directly by using data from the class CMDeviceMotion.

2. Formulae used for Computing $P^{C,t}$ and $p^{S,t}$:



CALCULATIONS

- Gyroscope data in rad/s.
- Multiply by 't' to get Ω vector
- $\theta = \|\Omega\|$ $\quad n = \frac{\Omega}{\|\Omega\|}$ $\quad [\Omega]_x = \text{Skew}(\Omega)$
- Full - rodriguez formula :

$$R(n,\theta) = I - [\Omega]_x \sin\theta + [\Omega]_x^2 (1-\cos\theta) = R_0^t$$

- Short - rodriguez formula :

$$R(n,\theta) \approx I - [\Omega]_x = R_0^t$$

- $$p^{S,0} = \begin{bmatrix} p_x^{S,0} \\ p_y^{S,0} \end{bmatrix} \Rightarrow p^{C,0} = \begin{bmatrix} p_x^{S,0} - c_x \\ p_y^{S,0} - c_y \\ f \end{bmatrix} \qquad \alpha = 1$$

- $$p^{C,t} = R_0^t \, p^{C,0}$$

$$\Rightarrow p^{C,t} = \begin{bmatrix} R_0^t \end{bmatrix} \begin{bmatrix} p_x^{S,0} - c_x \\ p_y^{S,0} - c_y \\ f \end{bmatrix}$$

$$\Rightarrow p^{S,t} = \begin{bmatrix} f \cdot \frac{p_x^{C,t}}{p_z^{C,t}} + c_x \\ f \cdot \frac{p_y^{C,t}}{p_z^{C,t}} + c_y \end{bmatrix}$$

3. Description and usage of the system

I started off by creating a class called Rodriguez, where i wrote all basic functions required for computing the rotation matrices. I created functions for computing:

- Norm of the vector
- Skew symmetric matrix of a vector
- Rotation matrix according to full Rodriguez formula
- Rotation matrix using small angle approximation of Rodriguez formula

The touch point was detected in the touches began function. To plot this point on the screen, I multiplied with the mapping constant to account for the change between the resolutions of the screen and that of the context. Then the point is converted to the gyro frame by using:

```
pS.x = touchLocation.x*(_contextSize.y/_viewSize.x);
pS.y = (_contextSize.x) - (touchLocation.y*(_contextSize.x/_viewSize.y));
```

To access gyroscope data, I used the push method. The first rotation matrix was set to identity,as that is the point from where we are supposed to start tracking. The initial time was set at this moment. Then in the next iteration, the new rotation matrix was obtained from the Rodriguez formula. Then on, for every iteration the rotation matrix was updated by multiplying the old rotation matrix with the new one obtained from the Rodriguez formula for that iteration. In this way the position of the point is updated with the set of rotation starting from the original point. The camera coordinates are then rotated to align with the context reference frame by rotating around the Y axis (X and Z coordinated are negative of their previous value). Now the coordinates are converted back to the pixel coordinates and a circle is drawn around it.

To access device motion data, I again used the push method. The initial rotation matrix was captured in the attitude object. In the next iteration the new rotation is calculated using the the method multiplyByInverseOfAttitude to generate the subsequent rotation matrix. This rotation is then applied to the camera coordinate of the point obtained from the touch interface. The rotated points then aligned with content reference frame by rotating around the Y Axis (X and Z coordinated are negative of their previous value). These coordinates are then converted to pixel coordinates and then a circle is drawn around that point.
To track the point, a single circle is drawn instead of a trail of circles(although this can be selected in UI). When device motion data is being used to track the point, the gyro push method is shut down and vice versa.

3.1 Using the Application

The app has been coded up to work in the following way:

To start the sensors the start button needs to be pressed. This is the first step to be done when the app starts. Then if one wishes to use the Rodriguez formula implementation to track the object,

the Gyro segment is to be selected. Here if the full Rodriguez formula is to be used, the switch must be kept off. If the small angle approximation is desired to be used, then the switch must be turned on. This can be done even when the gyro segment has been selected. To use device motion filtered values, the device motion segment must be selected.
Once the segment is selected, one can touch the screen to select the point to be tracked. The code has been written so that instead of a trail of circles, a single circle is drawn to track the point(this can be changed in the UI).
The stop button shuts all sensor methods and removes the circle from the screen. No tracking is being done when the stop button is pressed.

4. Individual contribution breakdown

4.1. Vikrant More : Wrote the code for the app. Wrote the description and usage of the app in the report.

4.2. Bharath Nagesh : Worked on the app interface, and wrote the modalities description in the report.