Train Control Assignment 02

เริ่มสั่งงาน : 3 ตุลาคม 2562 กำหนดส่ง : ก่อนวันที่ 11 ตุลาคม 2562

Git-repo : https://github.com/zeabusTeam/train_control

เนื้อหา

1. Basic Matrix Operator & Expression (Lecture & Assignment)
2. Frame description plane dimension
3. Thruster Mapper
4. งานเรื่อง ROS NODE of Thruster Mapper

1. Basic Matrix Operator & Expression

ref: http://tewlek.com/anet_matrix.html

ref: https://math.stackexchange.com/questions/1335693/invertible-matrix-of-non-square-matrix

เรากล่าวว่า A เป็นเมทริกซ์ ที่มีมิติ m×n $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = [a_{ij}]$

ในส่วนนี้ขอให้อ่านเนื้อหาเพิ่มเติมทั่วไปเกี่ยวเก็บ matrix ตาม ref แรกที่ให้ไปเพื่อเรียนรู้เกี่ยวกับการอ้างอิงถึงสมาชิก สมบัติการคูณ การหาร เนื้อหาที่จะกล่าวต่อไปนี้ จะขอกล่าวเพียงนำไปสู่การหา invertible of non square / square matrix.

1.1. Transpose expression

การทรานสโฟส เป็นการนำแถวที่ n มาเป็นหลักที่ n แทนของทรานสโพสนั้น ๆ A^t เป็นทรายสโฟสของ A

$$A = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \\ c & c_2 \end{bmatrix}, A^t = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \\ c & c_2 \end{bmatrix}^t = \begin{bmatrix} a_1 & b_2 & c_1 \\ a_2 & b_2 & c_2 \end{bmatrix}$$

1.2. determinant expression (det A, |A|)

ตามหลักแล้วในการ determinant ควรกล่าวหลัง minor และ cofactor แต่ในที่นี้จะขอกล่าวก่อน เพราะมองว่าเป็นพื้นฐานส่วนหนึ่งของการทำ minor และ cofactor

การทำ determinant ทำได้สำหรับ square matrix โดยมีพื้นฐานอยู่ที่ matrix dimension = 2×2 โดยหา determinant ได้ดังต่อไปนี้

$$\det\left(\begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}\right) = a_1 b_2 - a_2 b_1$$

สำหรับกรณีที่ dimension มากกว่า 2 ให้เลือกแถวใด หรือหลักใด มาเพียงหนึ่ง แล้วใช้ cofactor มาช่วยในการหา ดังวิธีต่อไปนี้

สมมุติเลือกแถวที่ i square matrix n $\det A = \sum_{j=1}^{n} a_{ij} C_{ij}(A)$

สมมุติเลือกหลักที่ j square matrix n $\det A = \sum_{i=1}^{n} a_{ij} C_{ij}(A)$

1.3. minor expression ($M_{ij}(A)$)

การทำ minor เป็นการทำที่ทำได้เฉพาะกับเมตทริกซ์จตุรัสเท่านั้น โดยให้ตัดแถวที่ i หลักที่ j ออกแล้วทำการหา determinant เมตทริกซ์ที่เหลือ ก็จะสามารถหาค่าของ minor ณ ตำแหน่งนั้น ๆ ได้

1.4. cofactor expression ($C_{ij}(A)$)

มีสมการการทำงานดังนี้

$$C_{ij}(A) = (-1)^{i+j} M_{ij}(A)$$

1.5. adjacent expression (adj(A))

ใช้ได้สำหรับ square matrix เท่านั้น

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, adj(A) = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

1.6. invertible square matrix

$$A^{-1} = \frac{1}{\det(A)} adj(A)$$

1.7. invertible non-square matrix

เมื่อเมทริกซ์ที่ต้องการหา inverse ไม่ใช่เมตรทริกซ์จตุรัส เราจะแบ่งในวิธีการหาออกเป็น 2 แบบ คือ ด้านซ้าย กับด้านขวา โดยข้อมูลอ้างอิงตาม ref 02

$$A_{right}^{-1} = A^T(AA^T)^{-1} \ , \qquad A_{left}^{-1} = (A^TA)^{-1}A^T$$

1.8. Assignment for matrix part

ขอให้ฝึกทำความเข้าใจในเรื่องการหา invertible of matrix ซึ่งจะใช้ความรู้ทุกหัวข้อที่เกี่ยวข้อง ตั้งแต่ determinant cofactor minor and adjacent ทั้งนี้ถ้ามีวิธีอื่นในการหาสามารถใช้วิธีนั้นได้ ตามสะดวก โดยมีแบบฝึกหัดทั้งหมด 6 ข้อ สามารถใช้เครื่องคิดเลข หรือคอมพิวเตอร์ตรวจสอบได้ แต่ขอให้เขียนถึงขั้นตอนการทำ expression ต่างๆ โดยมุ่งเน้นให้เห็นถึงความเข้าใจการทำงานของมัน

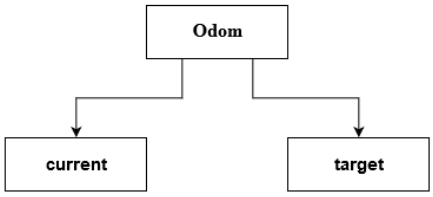
2. Frame description plane dimension

ref : http://wiki.ros.org/tf/Tutorials/Introduction%20to%20tf

ref : https://www.ros.org/reps/rep-0105.html

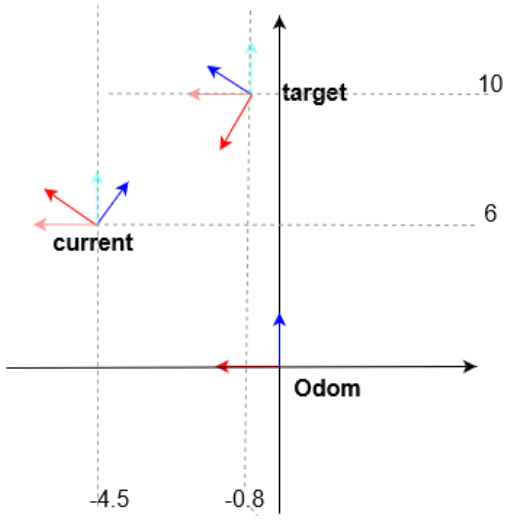
ขอให้อ่านเนื้อหาข้างตนแล้วทำความเข้าใจคร่าว ๆ ถ้ามีปัญหายังไม่เข้าใจแนวคิดขอให้หาเวลาว่างมาพบ เพื่อสอบถามเรื่องที่ไม่เข้าใจ เพราะส่วนนี้จะเป็นแก่นสำคัญในการทำงาน Control ที่จะใช้การคำนวณผ่าน transformation frame to frame

ลองพิจารณาตัวอย่างต่อไปนี้



จะเห็นว่าเรามี 3 เฟรม คือเฟรม current, odom, target ในการทำงานเราให้ odom เป็นจุดเริ่มต้น การหา tf จาก odom → current คือการหาว่าเราอยู่ตำแหน่งใด ส่วน odom → target คือการหาว่าเป้าหมายเราเป็นตำแหน่งใดลักษณะอย่างไร ดังนั้น current→target คือการหาว่า เราห่างจาก target เท่าไร หรือก็คือ error ที่เราได้นั้นเอง

ต่อไปเราจะพูดอย่างเจาะจงในเรื่องของกรณี 2 มิติ คือ ระนาบ xy เพื่อทำความเข้าใจถึงหลักการทำงานของมัน โดยมีโครงสร้างการกำหนด parent frame กับ child frame ตามตัวอย่างข้างบน



จากภาพดังกล่าว เป็นเปรียบเสมือนแผนที่ โดยกำหนดให้เส้นสีน้ำเงินคือ แกน x และเส้นสีแดง คือแกน y จากแผนภาพเราสามารถสรุปได้ดังนี้ โดยกำหนดให้ การแสดงผลคือ (x,y) , (yaw) โดย xy คือระยะ x , y หน่วย meter ส่วน yaw คือมุม หน่วย radian นั้นเอง

Child / Parent	Odom	Current	Target
Odom	(0 , 0) , (0)	(6 , -4.5) , (-0.7)	(10 , -0.8) , (0.8)
Current	(-6 , +4.5) , (0.7)	(0 , 0) , (0)	(4 , 3.7) , (1.5)
Target	(-10 , 0.8) , (-0.8)	(-4 , -3.7) , (-1.5)	(0 , 0) , (0)

ขอให้ทำความเข้าใจ ข้อมูลที่ได้จากตารางให้ได้ว่ามาได้อย่างไร

3. Thruster Mapper

สมการหลักที่ใช้ในการคำนวณเกี่ยวกับการแตกแรงเข้าสู่ทรัสเตอร์แต่ละตัวคือ

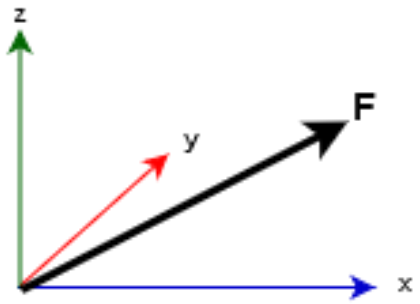
$$\tau_{robot} = MF_{thruster}$$

- เมื่อ τ_{robot} คือ แรงเชิงเส้น เชิงมุมใน frame ของ robot
- เมื่อ M คือ เมทริกซ์ที่จะ map แรง thruster แต่ละตัวเข้าสู่ τ_{robot} ขอเรียกว่า เมทริกซ์เชื่อมโยง
- $F_{thruster}$ คือ เทกทริกซ์แรงของทรัสเตอร์

ก่อนที่จะเข้าสู่การสร้างเมทริกซ์เชื่อมโยง จะขอทบทวนเนื้อหาฟิสิกส์เบื้องต้นเกี่ยวกับ แรงเชิงเส้น และแรงโมเมนต์

3.1. Force Linear

ในกรณีนี้จะขอพูดถึง 3 มิติเป็นหลัก เพื่อให้เห็นภาพครอบคลุมมากที่สุด เวลาคำนวณ เราจะแยกแรงออกเป็นทั้งหมด 3 แกน หรือตามจำนวนมิติด้วยกัน ดังนี้



จากภาพดังกล่าวสมมุติให้แรง F มีมุม α , β ,γ ทำกับแกน x , y , z ตามลำดับ ดังนั้นจะสามารถแตกเข้าแกน x , y ,z ได้ด้วย $F\cos\alpha$, $F\cos\beta$, $F\cos\gamma$ ตามลำดับนั้นเอง เราจะสามารถเขียนในรูปของสมการหลักในบทนี้ได้ว่า

$$\begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} = \begin{bmatrix}cos\alpha\\ cos\beta\\ cos\gamma\end{bmatrix}|F|$$

3.2. Moment of Force

แรงนั้นจะทำให้วัตถุหมุนหรือใหม่ก็ขึ้นอยู่กับ โมเมนต์ที่เกิดขึ้นจากแรงนั้น ๆ นั้นเอง ดังสมการต่อไปนี้

$$\boldsymbol{\tau} = \boldsymbol{r} \times \boldsymbol{F}$$

- เมื่อ r คือระยะทางจากจุดหมุน ไปจุดที่แรงกระทำ $<r_x, r_y, r_z>$ เมื่อทั้งหมดเป็นระยะห่างในแกน x, y ,z
- เมื่อ F คือเวกเตอร์ของทรัสเตอร์
- τ คือโมเมนต์ ของ แรงรอบแกน x y z

การทำงานของ × คือ cross product โดยเราจะมอง vector F เป็น unit vector กับ ขนาด ทำให้ได้

$$\tau = (r \times \hat{u}_F)|F| \, , \begin{bmatrix} \tau_\alpha \\ \tau_\beta \\ \tau_\gamma \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} |F|$$

3.3. เมทริกซ์เชื่อมโยง

โดยเมทริกซ์เชื่อมโยงนั้นจะเป็นการรวมส่วนที่อยู่ข้างหน้า |F| เข้ามาอยู่ด้วยกันนั้นเอง เช่น

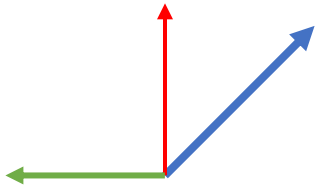
$$\tau_{robot} = \begin{bmatrix} cos\alpha \\ cos\beta \\ cos\gamma \\ d_x \\ d_y \\ d_z \end{bmatrix} |F|$$

4. งานเรื่อง ROS NODE of Thruster Mapper

จงเขียน node ROS ที่รับ msg force_plane ที่ระบุว่าต้องการแรงเชิงเส้นในแกน x, y เท่าไร แล้วโมเมนต์รอบแกน y เท่าไร โดยจะส่งออกเป็นว่า ทรัสเตอร์แต่ละตัวจะต้องออกแรงเท่าใด เมื่อกำหนดให้มีข้อมูลดังนี้

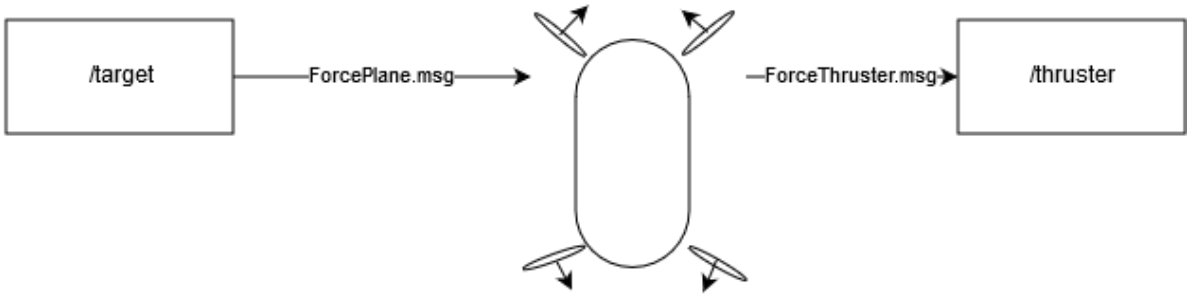
หมายเลขทรัสเตอร์	ระยะห่างจากจุดหมุน x	ระยะห่างจากจุดหมุน y	มุมที่ทำกับแกน +x
1	0.25 m	0.2 m	-45 องศา
2	0.25 m	-0.2 m	45 องศา
3	-0.4 m	0.2 m	-135 องศา
4	-0.4 m	-0.2 m	135 องศา

เมื่อการวัดมุมวัดดังนี้



- เส้นสีแดง คือแกน x หน้าหุ่น
- เส้นสีเขียว คือแกน y ด้านซ้ายหุ่น
- เส้นน้ำเงินคือแรงทรัสเตอร์ F+ โดยวัดมุมจากสีแดงไปน้ำเงิน

โดยภาพรวมของโปรแกรมมีดังนี้



ให้เขียน node ที่ subscribe topic /target โดยใช้ message คือ ForcePlane แล้วส่งออกว่าทรัสเตอร์แต่ละตัวต้องออกเท่าไรผ่าน topiv /thruster โดย message คือ ForceThruster ทั้งนี้ตัว message ได้สร้างไว้เรียบร้อยแล้วที่ package train_control

การทำงานของให้ทำการ pull branch master ของ repository โดยถ้าต้องการเขียนภาษา C/C++ ให้เขียน CMakeLists.txt ใน directory HW02 ได้เลย ทั้งนี้นั้นไฟล์ source code ที่ใช้รันเกี่ยวข้องทั้งหมดจะอยู่ใน HW02 เท่านั้น โดยใช้คำสั่ง

git pull origin master

ใน directory ดังกล่าว ได้หมอบตัวอย่างโค้ดที่ใช้จริง แต่เป็นการคุม 6 degree of freedom ถ้ามีเรื่องสงสัย หรืออยากรู้ของให้มานัดพบ ถามต่อหน้า

ในการส่งงานให้ส่งด้วยวิธีเดิม

ประวัติการปรับงานครั้งที่ **2**

Version	Detail	Date	Name
1.0	งานในส่วนของ thruster mapper ครั้งแรกบนระนาบ plane	2019 Oct, 03	K.Supasan