METR4202 -- Robotics
Tutorial 2 – Week 3: Homogeneous Coordinates

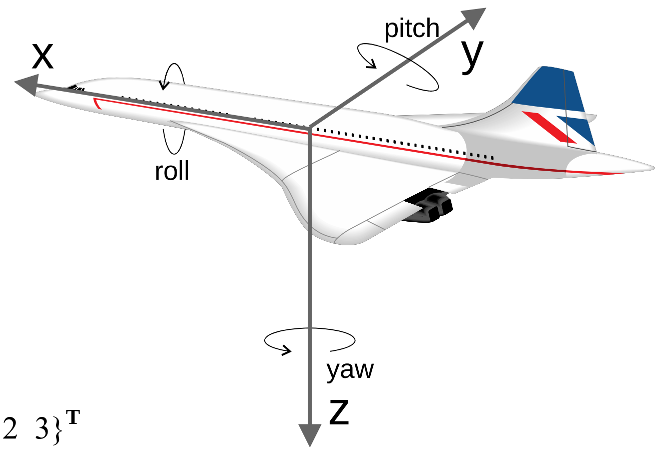
SOLUTIONS & COMMENTARY

Questions

1. Calculate the homogeneous transformation matrix ${}^{A}_{B}T$ given the translations (${}^{A}\boldsymbol{P}_{B}$) and the roll-pitch-yaw rotations (as α-β-γ) applied in the order yaw, pitch, roll. **[20 points]**

   a. α=10°, β=20°, γ=30°, ${}^{A}\boldsymbol{P}_{B}=\{1\ \ 2\ \ 3\}^{\mathbf{T}}$

First let's assume a convention, such as the "Engineering" convention in which Yaw is about the Z-axis (pitch is about the Y-axis and roll is about the X-axis) [[Lecture 2, Slide 24]].



Recall that :

$$ {}^{A}_{B}\mathbf{T} = \begin{bmatrix} & {}^{A}_{B}\mathbf{R} & & {}^{A}\mathbf{P}_{B} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Thus, we need ${}^{A}\boldsymbol{R}_{B}$ and ${}^{A}\boldsymbol{P}_{B}$. ${}^{A}\boldsymbol{P}_{B}$ is given as $\{1\ \ 2\ \ 3\}^{\mathbf{T}}$

${}^{A}\boldsymbol{R}_{B}$ is The problem request the rotation in yaw-pitch-roll, which using the aforementioned convention would be Z-Y-X. We can compute these from the Euler Angle representation [[Lecture 2, Slide 26]] or using the robotics toolbox with:

```
R = rpy2r(ROLL, PITCH, YAW, 'zyx')  % see doc rpy2r
```

This gives:

```
Rab = rpy2r(10, 20, 30, 'deg', 'zyx');
Rab =
    0.9254    0.0180    0.3785
    0.1632    0.8826   -0.4410
   -0.3420    0.4698    0.8138
```

Thus:

$$ {}^{A}_{B}\mathbf{T} = \begin{bmatrix} 0.9254 & 0.01803 & 0.3785 & 1.0 \\ 0.1632 & 0.8826 & -0.441 & 2.0 \\ -0.342 & 0.4698 & 0.8138 & 3.0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix} $$

The bit to note, which might be confusing, is that ${}^{A}\boldsymbol{P}_{B}$ is given in the {A} coordinate frame. If (hypothetically) it was given in the {B} coordinate frame, then we would have to rotate (i.e., ${}^{A}\boldsymbol{P}_{B}={}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{P}_{B}$) to make the frames (or basis) align.

b. $\alpha=10°$, $\beta=30°$, $\gamma=30°$, ${}^{A}\boldsymbol{P}_{B}=\{3\ \ 0\ \ 0\}^{\mathbf{T}}$

This is similar to (a). Using the same procedure as before we get:

```
Rab=rpy2r(10, 30, 30, 'deg', 'zyx')
Rab =
    0.8529    0.0958    0.5133
    0.1504    0.8963   -0.4172
   -0.5000    0.4330    0.7500
```

$$
{}^{A}_{B}\mathbf{T} = \begin{bmatrix} 0.8529 & 0.09582 & 0.5133 & 3.0 \\ 0.1504 & 0.8963 & -0.4172 & 0 \\ -0.5 & 0.433 & 0.75 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}
$$

2. Compare the output of: $\alpha=90°$, $\beta=180°$, $\gamma=-90°$, ${}^{A}\boldsymbol{P}_{B}=\{0\ \ 0\ \ 1\}^{\mathbf{T}}$ and $\alpha=90°$, $\beta=180°$, $\gamma=270°$, ${}^{A}\boldsymbol{P}_{B}=\{0\ \ 0\ \ 1\}^{\mathbf{T}}$ **[10 Points]**

The difference between these is 360° in γ.
There can be numerical issues though at such angles. Note the ±0 terms when computed in MATLAB:

```
>> R1=rpy2r(90, 180, 270, 'deg')
R1 =
    0.0000   -1.0000    0.0000
   -0.0000    0.0000    1.0000
   -1.0000   -0.0000   -0.0000

>> R2=rpy2r(90, 180, -90, 'deg')
R2 =
   -0.0000   -1.0000    0.0000
   -0.0000    0.0000    1.0000
   -1.0000    0.0000   -0.0000
```

3. Given the following 3x3 rotation matrices: **[40 points]**

$$R_1 = \begin{bmatrix} 0.7500 & -0.4330 & -0.5000 \\ 0.2165 & 0.8750 & -0.4330 \\ 0.6250 & 0.2165 & 0.7500 \end{bmatrix}, \quad R_2 = \begin{bmatrix} 0.6399 & -0.2351 & -0.6159 \\ 0.2860 & 0.5854 & -0.4970 \\ 0.3221 & 0.2488 & 0.7132 \end{bmatrix},$$

$$R_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0.8660 & 0.5000 & 0 \\ -0.500 & 0.8660 & 0 \end{bmatrix}, \quad R_4 = \begin{bmatrix} 0.0238 & 0.1524 & 0.9880 \\ -0.3030 & -0.9407 & 0.1524 \\ 0.9527 & -0.3030 & 0.0238 \end{bmatrix}$$

a. Are these (within practical numerical limits) valid rotation matrices? Why?

A rotation matrix is orthonormal. Thus, let's check if these matrices satisfy these properties. Chiefly:

- Orthogonally: `inv(R) = transpose(R)`
- Normal: `det(R)=1 , norm(R)=1`

Running some checks in Matlab we get:

	mean(inv(R1)-R1')	**det(R)**	**norm(R)**
R1	4.0792e-007	1	1
R2	0.1819	0.5042	1.1286
R3	8.4679e-006	1	1
R4	-1.3397e-006	1	1

Thus we can conclude (with practical numerical limits) that **R1, R3, and R4 are rotation matrices** and that **R2 is not a rotation matrix**.

b. If yes, determine the Roll, Pitch, and Yaw that define each matrix. Do you believe their values?

A rotation matrix is redundant and thus is over constrained for the 3 Euler angles. The values can be found by solving at least 3 of the 9 nonlinear simultaneous equations (see also Lecture 2, Slide 26).

In the robotics toolbox this is implemented (in part) as part of the **`tr2rpy`** function. Forming these rotation matrices and running this gives:

```
>> tr2rpy(TR1, 'deg')
ans =
   29.9993  -30.0002   29.9993

>> tr2rpy(TR3, 'deg')
ans =
         0   90.0000   59.9993
>> tr2rpy(TR4, 'deg')
ans =
  -81.1239   81.1266  -81.1239
```

Rounding and using some "intuition" we get the roll-pitch-yaw as:

- R1: Roll: 30 °, Pitch: -30 °, Yaw: 30 °

  We might trust this.

- R3: Roll: 0 °, Pitch: 90 °, Yaw: 60 °

  We might **not** trust or "believe" this answer. At pitch angles of 90 degrees, **there is a singularity** in the Roll-Pitch-Yaw Euler angles set (see also Lecture 3 – Slide 34 and or the documentation for tr2rpy [**`doc tr2rpy`**]). (In this case the "Roll" value is undetermined, but it is set to zero by the toolbox).

- R4: Roll: -81 °, Pitch: 81 °, Yaw: -81 °

  We might **not** trust this answer either. Intuition suggests that this "rather unusual" value. A little bit of sleuthing with the value in radians, rather suggests that this was a matrix computed without the appropriate degree conversion or setting (i.e., the numbers were in degrees but processed as radians). Thus giving: **Roll: 30 °, Pitch: -30 °, Yaw: 30 °.**

  That is the user may have typed **`rpy2tr([30 -30 30]),`**
  when they should have typed **`rpy2tr([30 30 30], 'deg')`**

While it was not asked for directly, it is possible to enforce orthogonally constraints back on R2 and to estimate the roll-pitch-yaw values.

There are several means for normalizing the matrix. One convenient approach is to use the singular value decomposition (SVD).

The SVD of R2 is:

```
[UR2,SR2,VR2] = svd(R2)
UR2 =
   -0.7282    0.0725   -0.6815
   -0.5131    0.6014    0.6123
    0.4543    0.7956   -0.4008
SR2 =
    1.1286         0         0
         0    0.7480         0
         0         0    0.5973
VR2 =
   -0.4133    0.6346   -0.6530
   -0.0143    0.7125    0.7015
    0.9105    0.2993   -0.2854
```

We can “normalize” this by setting the diagonal terms to 1.
Thus:
SR2NORM =

1	0	0
0	1	0
0	0	1

Multiplying this back together with the previous left-singular (or unitary or U) and right-singular (or unitary conjugate or V) matrices.

Thus giving:
>> R2NORM=UR2*SR2NORM*VR2'
R2NORM =

0.7920	-0.4159	-0.4469
0.1939	0.8654	-0.4620
0.5789	0.2793	0.7661

Solving for the angles via tr2rpy (tr2rpy(TR2N, 'deg')) gives:
Roll=31.0906 ° , Pitch=-26.5419 ° , Yaw=27.7067 °

Rounding/Engineering intuition suggests that this might actually be:
Roll=30° , Pitch=-30° , Yaw=30°