3D位姿变换案例

名称及缩写：

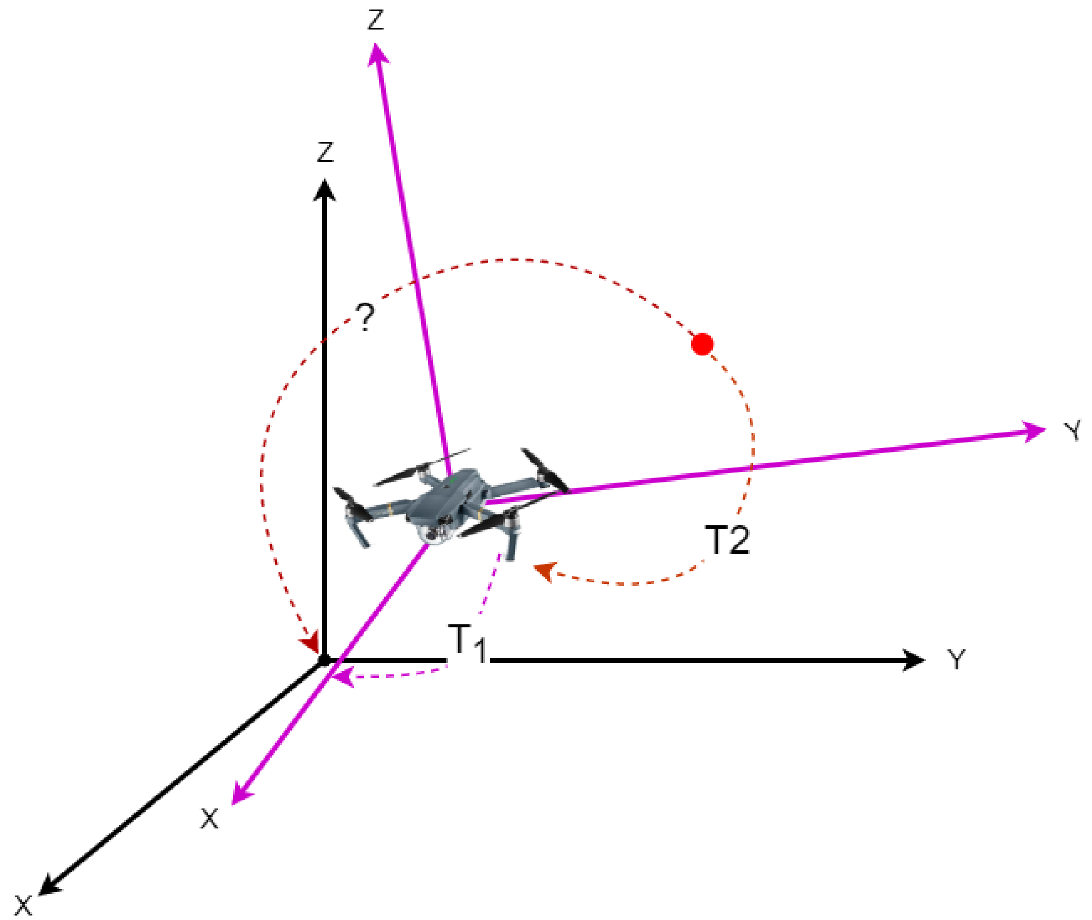
基地：Base （B）

侦察机/四轴飞行器：Quadcopter （Q）

火炮坦克：CannonTank（C）

敌军目标：Enemy（E）

案例一：侦察机汇报基地



题目：

如上图，已知

- 一个侦察机相对于总部的位置 $P_1 = {}^BP_Q = [1.5, 2.8, 3.2]$
- 侦察机其姿态 $RPY = [15, 30, 10]$ 单位为角度

求其位姿 $T_1 = {}^B_QT$？

- 若已知目标相对于侦察机的位置为 $P_2 = {}^QP_E = [-0.2, 3.4, 2.1]$，

求敌军目标相对于基地的坐标位置 ${}^BP_E$ ？

求解：

根据坐标系位姿的定义，我们可以将 RPY 作为欧拉角的 z, y, x 构建旋转矩阵，然后和位置共同构建 4x4 变换矩阵。

随后可以使用如下方式得到 ${}^{B}P_{E}$：

$$
{}^{B}P_{E} = {}^{B}_{Q}T \cdot {}^{Q}P_{E}
$$

实现：

- 准备依赖: `transformation.py`

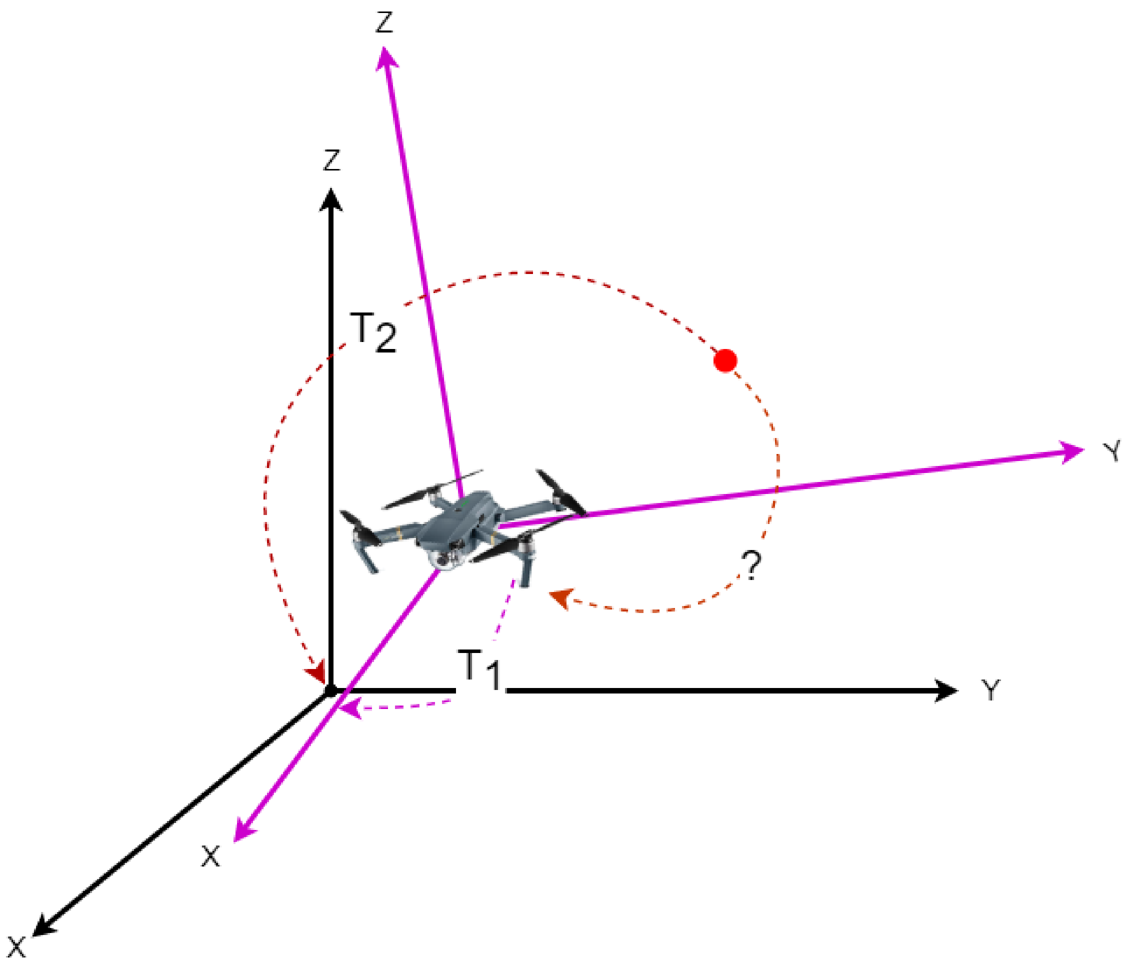
```
"""
坐标转换工具
"""
import numpy as np
from math import cos, sin, atan2

np.set_printoptions(precision=3, suppress=True, formatter={'float': "
{:.3f}".format})


def euler2matrix(theta, format="degree"):
    """
    将欧拉角转成旋转矩阵
    :param theta: 按照 x, y, z顺序的旋转角度
    :param format: 指定格式，默认为角度，否则为弧度
    :return: 一次按照动轴ZYX运动的欧拉角旋转矩阵
            等同于按照定轴XYZ的RPY运动
    """

    if format == "degree":
        theta = np.deg2rad(theta)

    q_x, q_y, q_z = theta

    R_x = np.array([[1, 0, 0],
                    [0, cos(q_x), -sin(q_x)],
                    [0, sin(q_x), cos(q_x)]])

    R_y = np.array([[cos(q_y), 0, sin(q_y)],
                    [0, 1, 0],
                    [-sin(q_y), 0, cos(q_y)]])

    R_z = np.array([[cos(q_z), -sin(q_z), 0],
                    [sin(q_z), cos(q_z), 0],
                    [0, 0, 1]])

    return R_z @ R_y @ R_x


def is_rotation_matrix(R):
    # 求R的转置矩阵 (也是逆矩阵)
    Rt = np.transpose(R)
    # 矩阵点乘自己的逆 = 单位矩阵
    should_be_identity = np.dot(R, Rt)
    I = np.identity(3, dtype=R.dtype)
```

```python
    n = np.linalg.norm(I - should_be_identity)
    return n < 1e-6


def matrix2euler(R):
    """
    旋转矩阵转欧拉角
    :param R: 旋转矩阵，必须是正交矩阵（其逆等于其转置）
    :return: 欧拉角 x, y, z
    """
    assert (is_rotation_matrix(R))

    sy = np.sqrt(R[0, 0] * R[0, 0] + R[1, 0] * R[1, 0])
    # 判断是否是奇异矩阵
    singular = sy < 1e-6

    print("singular: ", singular)

    if not singular:
        z = atan2(R[1, 0], R[0, 0])
        y = atan2(-R[2, 0], sy)
        x = atan2(R[2, 1], R[2, 2])
    else:
        z = 0
        y = atan2(-R[2, 0], sy)
        x = -atan2(R[1, 2], R[1, 1])

    return np.array([x, y, z])


def merge_pose(R, t):
    mat = np.eye(4, dtype=R.dtype)
    mat[:3, :3] = R
    mat[:3, 3] = t
    return mat


def split_pose(T):
    return T[:3, :3], T[:3, 3]


if __name__ == '__main__':
    matrix = euler2matrix([-120, 20, 90])
    print(matrix)

    euler = matrix2euler(matrix)
    print(np.rad2deg(euler))

    pose = merge_pose(matrix, np.array([1, 2, 3]))
    print(pose)

    print(split_pose(pose))
```

- 编写实现 `Exercise-1.py`

```python
import numpy as np

from transformation import euler2matrix, merge_pose

# 侦察机的位置姿态
p1 = np.array([1.5, 2.8, 3.2])
rpy = np.array([15, 30, 10])

# 目标在侦察机坐标系的位置
p2 = np.array([-0.2, 3.4, 2.1])

if __name__ == '__main__':
    R1 = euler2matrix(rpy)

    T1 = merge_pose(R1, p1)
    print(T1)

    p2 = np.append(p2, 1.)

    result = T1 @ p2

    print(result[:3])
```

案例二：基地通知侦察机



题目：

如图，已知

- 侦察机相对于总部的位置 $P_1 = {}^BP_Q = [1.5, 2.8, 3.2]$
- 侦察机姿态 $RPY = [15, 30, 10]$ 单位为角度，
- 目标相对于基地B的位置为 $P_2 = {}^BP_E = [2.286, 5.721, 5.819]$

求敌军目标E相对于侦察机Q的坐标位置 ${}^QP_E$ 。

求解：

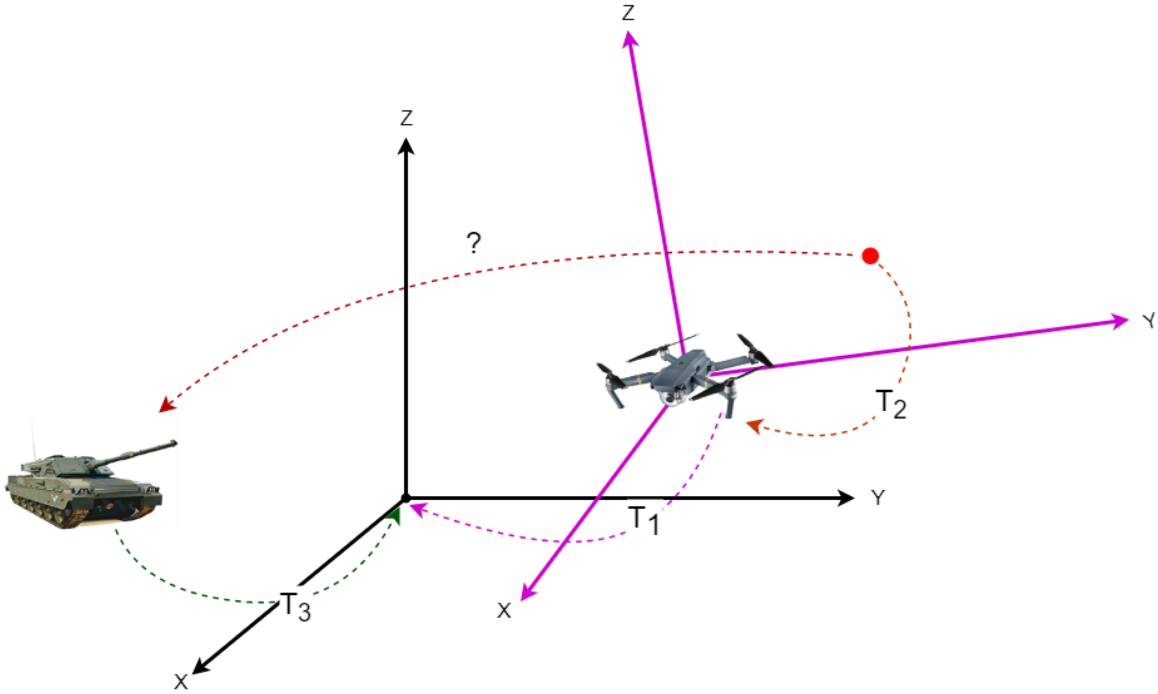
利用P1和RPY构建无人机位姿 ${}^B_QT$，已知如下等式： ${}^BP_E = {}^B_QT \cdot {}^QP_E$ ，使左右两边同时左乘 ${}^B_QT^{-1} = {}^Q_BT$ 可得：

$$ {}^B_QT^{-1} \cdot {}^BP_E = {}^QP_E $$

实现： `Exercise-2.py`

```
import numpy as np

from transformation import euler2matrix, merge_pose

# 侦察机的位置姿态
p1 = np.array([1.5, 2.8, 3.2])
rpy = np.array([15, 30, 10])

# 目标在基地坐标系的位置
p2 = np.array([2.286, 5.721, 5.819])

if __name__ == '__main__':
    R1 = euler2matrix(rpy)

    T1 = merge_pose(R1, p1)
    print(T1)

    p2 = np.append(p2, 1.)

    result = np.linalg.inv(T1) @ p2

    print(result[:3])
```

案例三：侦察机通知坦克



如图，已知：

- 侦察机Q相对于基地B的位姿为： $T_1 = {}^B_Q T$ 其中(p1=[1.5, 2.8, 3.2], rpy1=[15, 30, 10]）
- 敌军E相对于侦察机Q的位置为： $P_2 = {}^Q P_E$ 其中(p2=[-0.2, 3.4, 2.1])
- 坦克C相对于基地B的位姿为： $T_3 = {}^B_C T$ 其中(p3=[2.0, -4.0, 0.5], rpy3=[0, 15, 45])

求敌军目标E在坦克C当前状态下的坐标位置 ${}^C P_E$.