课程5 机器人的空间描述和变换

课程概述

在这个教程中，我们将学习机器人的空间描述和变换。

关于课程

按照规划，本课程需要花费大约2小时来完成，目标受众为四足机器人开发爱好者。

note

本课程是基于Mangdang的Mini Pupper系列产品搭建的。

本课程涉及的源代码托管在Github，你可以点击此处访问。

学习目标

在完成这个教程后，你将能够：

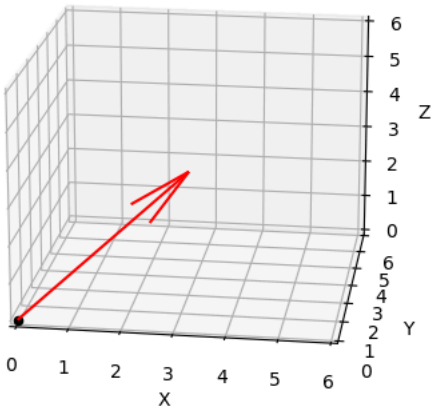
1. 通过刚体与刚体的平动、转动基础知识的学习，熟悉位姿的描述
2. 通过Python编程实践，可视化学习向量在坐标系中的变换，了解空间变换

课程细节

在这个课程中，我们将学习机器人运动学的基础-空间的描述与变换。

空间的描述与变换将定性定量地解释四足机器狗上零件的基本运动，四足机器狗mini pupper上有12个舵机，每个舵机都有自己的空间位置和运动，为了表达舵机等零件的本身的位置和姿态，我们需要定义坐标系并给出表示的规则，这些位置和姿态的描述将作为表达线速度、角速度、力和力矩的基础。

为了描述空间中物体的位置和姿态，一般可以在物体上设置一个坐标系，称为物体坐标系，在参考坐标系中描述通过位置和姿态来描述这个物体坐标系。任一的坐标系也可以作为另一个坐标系的参考坐标系，这就涉及到了坐标系和坐标系之间的位置和姿态的变换，称之为位姿变换。因此，要了解mini pupper的运动，需要研究同一物体在不同坐标系中的位置和姿态的描述方法，以及量化计算位置和姿态的数学方法。



图片1： 相对于坐标系的矢量

准备材料

在开始这个教程之前，你需要准备以下材料：

- 一台安装了Python的PC电脑
- 安装了绘图所需的依赖

引言

在这个教程中，我们将学习如何输入一指定向量和对应的绕行角度，使之在参考坐标系内绕x、y、z轴旋转指定角度，并依次按红、橙、绿、蓝绘制出原向量和绕x、y、z轴旋转后的向量

整体步骤

这个教程将分为几个任务：

1. 学习刚体变换的理论基本知识
2. 学习matplotlib及numpy基本的使用方法
3. 编写Python程序 vector_rotation.py
4. 运行程序，观察效果

请按照教程中的步骤完成每个任务。

任务1：学习基本知识

1.什么是刚体?

为了简化机器人模型，我们将四足机器狗mini pupper上的各关节零件视作刚体（rigid body），刚体是在运动中和受到力的作用后，形状和大小不变，而且内部各点的相对位置不变的物体。

2.怎样描述刚体的位姿?

位置描述

为了描述mini pupper上的零件在空间中的**位置position**，在数学中，我们通常用三个正交的带有箭头的单位矢量来描述一个三维坐标系，在这里称为世界坐标系，世界坐标系A中的点${}^{A}P$可以用一个3×1的位置矢量来表达，这个矢量可被认为是空间中的一个位置。对于任何一个正交坐标系N，都可以有${}^{N}P$这样一个位置矢量来描述点P相对于坐标系N的位置关系。

$$ {}^{A}P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} $$

${}^{A}P$：P点相对于坐标系A的位置矢量
p_x： P向量相对于坐标系A的x轴方向的分量

姿态描述

mini pupper上的零件刚体除了需要表示位置，通常还需要描述它的**姿态attitude**，为了描述这个姿态，我们将在物体上固定一个坐标系并且给出这个坐标系相对参考坐标系的描述。
也就是说，我们用一个固定在物体上的坐标系来描述这个物体的姿态，而这个坐标系的描述可以利用相对参考系的三个主轴单位矢量来描述。
用$\hat{X_B}$ 、$\hat{Y_B}$ 、$\hat{Z_B}$ 来表示坐标系B主轴方向的单位矢量，如果用坐标系A来作为参考系，则写作${}^{A}\hat{X}_B$ 、${}^{A}\hat{Y}_B$ 、${}^{A}\hat{Z}_B$ ，按前述顺序组成3×1的矩阵，这个矩阵被称为**旋转矩阵**。
旋转矩阵${}^{A}_{B}R$是坐标系B相对于参考坐标系A的表达:

$$ {}^{A}_{B}R = ({}^{A}\hat{X}_B \quad {}^{A}\hat{Y}_B \quad {}^{A}\hat{Z}_B) = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{bmatrix} \qquad (\text{旋转矩阵}) $$

${}^{A}_{B}R$：坐标系B相对于参考坐标系A的旋转矩阵
$\hat{X}_B$：坐标系B的x方向的单位矢量
$\hat{X}_A$：坐标系A的x方向的单位矢量

位姿描述

在四足机器狗mini pupper的运动学中，位置和姿态经常成对出现，称之为**位姿pose**，位姿可以用两个坐标系的相对关系来描述。
结合位置描述与姿态描述，得出一个位姿$\{B\}$可以等价地用一个位置矢量${}^{A}P_{BORG}$和一个旋转矩阵${}^{A}_{B}R$来描述：

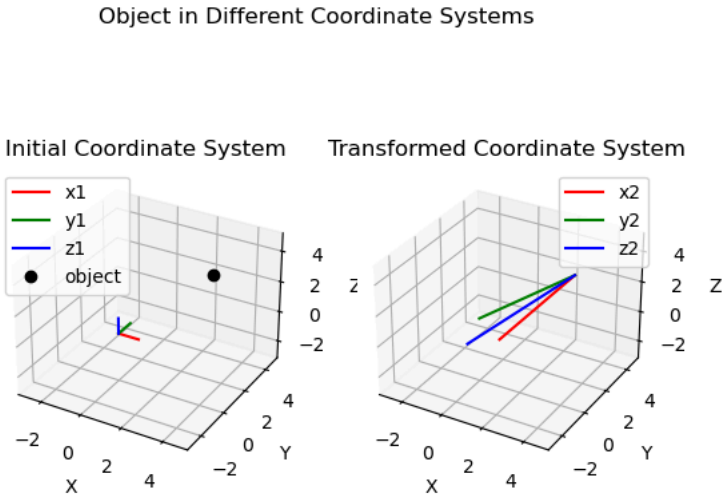
$$ \{B\} = \{{}^{A}_{B}R, {}^{A}P_{BORG}\} $$

${}^{A}P_{BORG}$：确定位姿$\{B\}$的原点的位置矢量，ORG为Origin，即原点之意

3.如何描述刚体的平移与旋转?

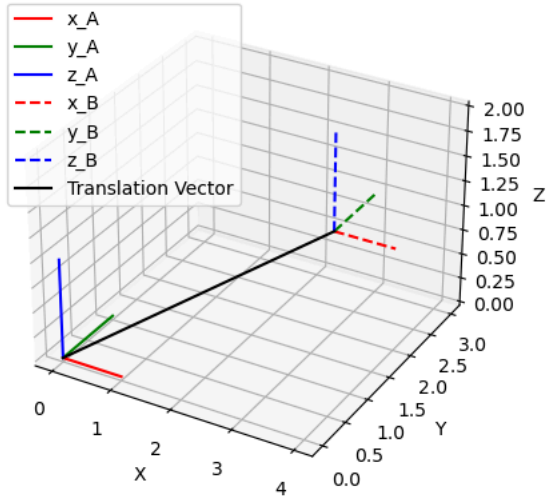
坐标平移

在了解完位置、姿态、位姿的概念后，我们将学习如何从一个坐标系变换到另一个坐标系，通常被叫做“**映射**”，“映射”使得mini pupper的刚体零件可以在不同的参考坐标系中被表达为一个相同的量。



图片2：不同坐标系下的Object

下面，就让我们来尝试将一个坐标平移来体会四足机器狗mini pupper上的刚体零件的运动。
坐标平移是简单的运动，在$\{A\}$和$\{B\}$的姿态相同时，$\{B\}$不同与$\{A\}$的只有平移，所以可以用一个位置矢量${}^{A}P_{BORG}$来描述$\{B\}$的原点相对于$\{A\}$的位置。



图片3：坐标平移
以下为当姿态相同时，矢量相加的办法求点P相对于$\{A\}$的表示${}^{A}P$:

$$ {}^{A}P = {}^{B}P + {}^{A}P_{BORG} $$

位置矢量${}^{A}P_{BORG}$来描述$\{B\}$的原点相对于$\{A\}$的位置。

在映射中，空间中的点本身没有改变，而是描述关系改变了，即点P的坐标相对于坐标系B的关系变换到了相对于坐标系A的关系。在姿态相同的情况下，矢量${}^{A}P_{BORG}$包含了变换所需的信息，它定义了这个平移的映射。

坐标旋转

在刚体的姿态描述中我们了解到，姿态可以用坐标系三主轴的单位矢量来描述，那么将这三个单位矢量依次排列，可以得到一个$3x3$的旋转矩阵。

通常认为，$\{B\}$相对于$\{A\}$的旋转矩阵表示为${}^A_BR$
因为旋转矩阵各列的模为1，各单位矢量均相互正交，在线性代数中，正交阵的逆矩阵等于它的转置矩阵，可得：

$$ {}^A_BR = {}^A_B\ R^{-1} = {}^B_A\ R^T $$

所以一个旋转矩阵可以为三个量为一组的行向量或者是三个量为一组的列向量。

${}^A_BR$矩阵的各列为$\{B\}$的单位矢量在$\{A\}$中的描述，即将$\{B\}$的单位矢量投影到$\{A\}$的单位矢量方向上。投影是由矢量的点积计算的。

$$ {}^A_BR = ({}^A\hat{X}_B \quad {}^A\hat{Y}_B \quad {}^A\hat{Z}_B) = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{bmatrix} \qquad (旋转矩阵) $$

${}^A_BR$：坐标系B相对于参考坐标系A的旋转矩阵
$\hat{X}_B$：坐标系B的x方向的单位矢量
$\hat{X}_A$：坐标系A的x方向的单位矢量

因此，对于原点位置相同的两个坐标系$\{A\}$和$\{B\}$，若要将一点P相对坐标系$\{B\}$的关系转化为对坐标系$\{A\}$的关系，可得

$$ {}^AP = {}^A_BR\ {}^BP $$

坐标的一般变换

对于mini pupper上的各零部件之间的变换，通常不只有坐标的平移或变换，而是两者兼有，这就是坐标的一般变换。一般变换采用先旋转，再平移的方法来变换坐标系。
将旋转和平移合并来看，可得

$$ {}^AP = {}^A_B\ R\ {}^BP + {}^AP_{BORG} $$

可以用更简洁的形式：一个4×4矩阵形式的算子来表示一个坐标系到另一个坐标系的映射。
可以看到式子中用了两个个4×1的位置矢量来分别表示${}^AP$和${}^BP$，这种位置矢量的末尾分量为1，有没有末尾的这个1取决与这个位置矢量与$3x3$还是$4x4$的矩阵相乘。

$$ \begin{pmatrix} {}^AP \\ 1 \end{pmatrix} = \begin{bmatrix} {}^A_BR & {}^AP_{BORG} \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{pmatrix} {}^BP \\ 1 \end{pmatrix} $$

这个4×4矩阵被称为**齐次变换矩阵**，它以普通矩阵的形式表示了一般变换中的旋转以及平移。
简写为：

$$ {}^AP = {}^A_BT\ {}^BP $$

一般来说，常用旋转矩阵定义“姿态”，而齐次变换矩阵常用于定义坐标系，例如坐标系$\{B\}$相对于$\{A\}$的变换就可描述为${}^A_BT$

平移算子

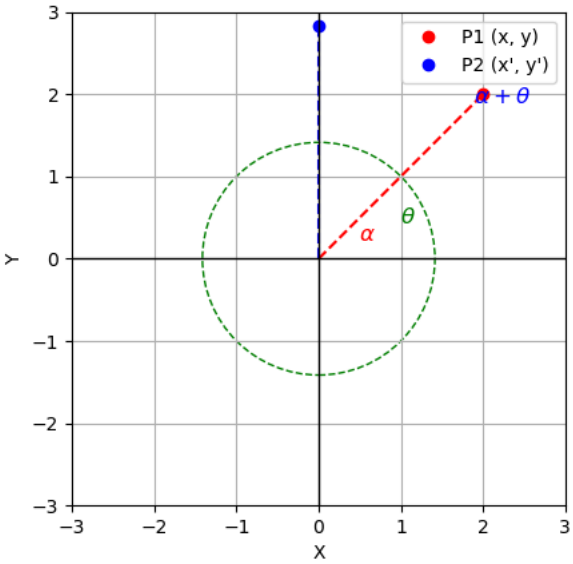
算子是坐标系间点映射的通用数学表达式，和前面所描述的方法类似，算子有点平移算子、矢量旋转算子、一般变换算子。
平移算子可以对点进行平移，矢量${}^AQ$给出了平移的信息，q是沿矢量${}^AQ$方向平移的数量。q_x、q_y、q_z都是平移矢量Q的分量。
例如从${}^AP_1$点平移到${}^AP_2$点

$$ {}^AP_2 = D_Q(q){}^AP_1 $$

$$ D_Q(q) = \begin{bmatrix} 1 & 0 & 0 & q_x \\ 0 & 1 & 0 & q_y \\ 0 & 0 & 1 & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

旋转算子

对于平面内图形或坐标系的旋转，可以视作这个图形或坐标系上各点相对于原坐标系的旋转得来。
如图所示，坐标为(x, y)的点P旋转角度θ到点P'，旋转后坐标为(x', y')。



图片4：坐标旋转

$$r = \sqrt{(x^2 + y^2)}$$

$$x = r \cdot Cos\alpha \quad y = r \cdot Sin\alpha$$

$$x' = r \cdot Cos(\alpha + \theta) \quad y' = r \cdot Sin(\alpha + \theta)$$

可得平面点绕原点旋转公式：

$$x' = xCos\theta - ySin\theta \quad y' = xSin\theta + yCos\theta$$

绕z轴旋转θ的旋转算子R将矢量${}^{A}P_1$旋转变换为${}^{A}P_2$

$${}^{A}P_2 = R{}^{A}P_1$$

该旋转算子的各列可看做旋转后的各单位矢量在旋转前单位矢量上的投影，可以发现和平面点绕原点旋转公式是逻辑一致的。

$$R_z(\theta) = \begin{bmatrix} Cos\theta & -Sin\theta & 0 & 0 \\ Sin\theta & Cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

一般变换算子

同理，结合旋转与平移，可得一般变换算子

$${}^{A}P_2 = T{}^{A}P_1$$

算子、映射、位姿描述都是齐次变换矩阵的解释，具体倾向哪个解释则取决于你对该矩阵的用途是倾向于描述还是变换。

任务2：学习matplotlib及numpy基本的使用方法

1. matplotlib

matplotlib是Python语言及其数值计算库NumPy的绘图库。它的设计与MATLAB非常类似，能够在Python中方便地绘制图像。
参考链接：matplotlib

2. numpy

NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库，因此可以用来计算坐标系的变换。
参考链接：numpy

3. 绘制向量的函数

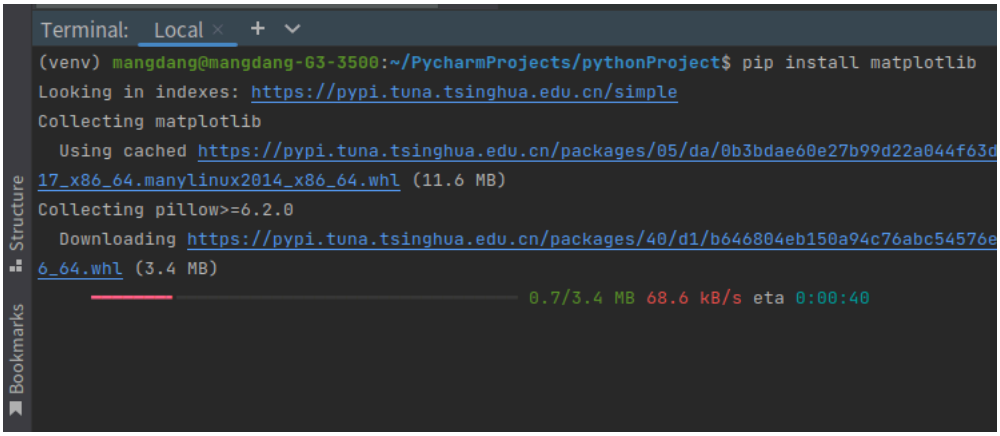
```
ax.quiver(起点x, 起点y, 起点z,x方向数值, y方向数值, z方向数值,arrow_length_ratio=箭头箭身比, color="颜色")
```

4. 用户的输入部分

```
move_pre= input("请输入坐标系原点相对参考坐标系原点平移的x、y、z分量：")
move = [int(n) for n in move_pre.split()]
print(move)
rotate_pre= input("请输入坐标系绕参考坐标系x轴、y轴、z轴依次旋转的角度：") # 旋转采用Fixed Angles模式
rotate = [int(n) for n in rotate_pre.split()]
print(rotate)
```

5. 安装matplotlib和numpy环境

```
sudo apt install pip # 安装pip
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple # 换源
sudo pip install matplotlib # 安装matplotlib
sudo pip install numpy # 安装numpy
```



图片5：配置环境

任务3：编写Python程序 vector_rotation.py

```
#!/usr/bin/python
# coding:utf-8
# vector_rotation.py
# Rotate a specified vector in the reference coordinate system by a specified angle around the x, y, and z axes, and draw the original vector a
import matplotlib.pyplot as plt
import numpy as np


def rotate_X(x, y, z, alpha):
    alpha_rad = np.radians(alpha)
    x_r = x
    y_r = np.cos(alpha_rad) * y - np.sin(alpha_rad) * z
    z_r = np.sin(alpha_rad) * y + np.cos(alpha_rad) * z
    return x_r, y_r, z_r


def rotate_Y(x, y, z, beta):
    beta_rad = np.radians(beta)
    x_r = np.cos(beta_rad) * x + np.sin(beta_rad) * z
    y_r = y
    z_r = -np.sin(beta_rad) * x + np.cos(beta_rad) * z
    return x_r, y_r, z_r


def rotate_Z(x, y, z, gamma):
    gamma_rad = np.radians(gamma)
    x_r = np.cos(gamma_rad) * x - np.sin(gamma_rad) * y
    y_r = np.sin(gamma_rad) * x + np.cos(gamma_rad) * y
    z_r = z
    return x_r, y_r, z_r


def draw_vector(ax, origin, vector, color):
    ax.quiver(origin[0], origin[1], origin[2],
              vector[0], vector[1], vector[2],
              arrow_length_ratio=0.1, color=color)


def draw_axes(ax, origin, length):
    x_axis = [length, 0, 0]
    y_axis = [0, length, 0]
    z_axis = [0, 0, length]
    draw_vector(ax, origin, x_axis, 'black')
    draw_vector(ax, origin, y_axis, 'black')
    draw_vector(ax, origin, z_axis, 'black')


def main():
    # Initialize plot
    fig = plt.figure()
    ax = fig.add_subplot(projection="3d")
    ax.set(xlim3d=(0, 5), xlabel='X')
    ax.set(ylim3d=(0, 5), ylabel='Y')
    ax.set(zlim3d=(0, 5), zlabel='Z')

    # Initialize variables
    origin = [0, 0, 0]
    length = 4
    draw_axes(ax, origin, length)

    vector_input = input("Please enter the x, y, z components of the vector, separated by spaces:")
    vector = [float(n) for n in vector_input. split()]
    rotation_input = input("Please enter the angle of rotation of the vector around the x, y, z axis (in degrees):")
    rotations = [float(n) for n in rotation_input.split()]

    draw_vector(ax, origin, vector, 'red')
```

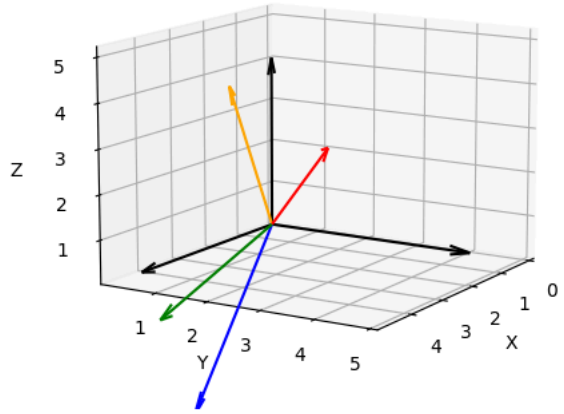
```
    vector_rotated_x = rotate_X(*vector, rotations[0])
    draw_vector(ax, origin, vector_rotated_x, 'orange')
    vector_rotated_xy = rotate_Y(*vector_rotated_x, rotations[1])
    draw_vector(ax, origin, vector_rotated_xy, 'green')
    vector_rotated_xyz = rotate_Z(*vector_rotated_xy, rotations[2])
    draw_vector(ax, origin, vector_rotated_xyz, 'blue')

    plt.show()


if __name__ == "__main__":
    main()
```

任务4： 运行程序，观察效果

在vector_rotation.py的目录下执行以下命令：

```
sudo python vector_rotation.py
```

输入对应的转动角度，此时应观察到向量绕各轴依次转动后的变化。



图片6：vector_rotation.py效果

总结

经过本知识点的学习和实验操作，你应该能达到以下水平：

知识点	内容	了解	熟悉	掌握
刚体	刚体的平动、转动基础知识		✓	
位姿	位姿的描述		✓	
坐标变换	坐标系之间的变换	✓		

背景信息和资源

本教程基于以下资源编写：
参考链接：matplotlib

参考链接：numpy

mini pupper相关内容可访问：https://github.com/mangdangroboticsclub