HW4

311512064鄧書桓

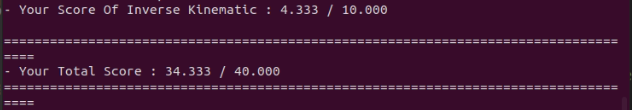
**Task1:**

**一張含有 文字 的圖片

自動產生的描述**

這邊僅附上結果得分圖，想法與如何實作將會在task做說明。

**Task2:**

****

這邊僅附上結果得分圖，想法與如何實作將會在task做說明。

**Task3:**

**1.code about fk.py**

一張含有 文字 的圖片

自動產生的描述

**1.1**

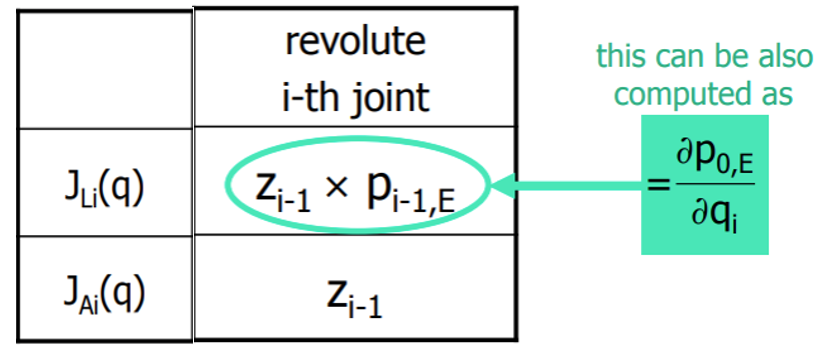
**A:**

一張含有 文字 的圖片

自動產生的描述

這邊根據已知的DH model、已知每個時間點個軸的轉速與投影片上推導出的轉移函數式子(上面的式子)可以得到各個轉軸間的關係(這邊利用for迴圈來實現個軸的關係T)，並以此來獲得題目所要求的A，也就是末端點的pose。

**Jacobin:**

一張含有 文字 的圖片

自動產生的描述

這邊的Jacobian是利用幾何解的形式來求得的，根據上課推導出(化簡)的式子能夠得出上面的關係式，接著利用for迴圈與上方的式子能夠得到每個軸各自的線速度與角速度。

**1.2**

D-H convention 也稱為 classic D-H convention，其將Oi定義在第i+1軸、將a與alpha 定義為第i 至 i+1軸之間的變化 ; Craig’s convention則是將Oi定義在第i軸、將a與alpha 定義為第i-1至 i軸之間的變化。這樣的差異會導致同個robot的D-H model不同，進而影響轉移矩陣的內容與其前乘後乘的順序。

**1.3**

一張含有 桌 的圖片

自動產生的描述

**2. code about ik.py** 一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

**2.1**



這邊是使用疊代法來找inverse Jacobian，會不斷逼近想要到的pose並直到誤差小於一個固定值才會停下來。在每次疊代中，都會使用到上面這個公式，其中的delta(x)求得的方式較不直觀，是使用向量(matrix)的形式來實現的，並利用下面這個概念: matrix 型態的delta(x) @ matrix 型態的your\_pose = matrix 型態的new\_pose，得到delta(x)之後只要根據上方的算式即可得到delta(theta)。接著要考慮變更後的角速度是否會超過joint limit與其誤差是否小於所設的疊代誤差。最終，當疊代至誤差小於固定值後，其所生成的inverse Jacobin與各軸角速度即是我們所想要的。

另外，我還有設計stepsize，用來將每次疊代過後的修正變得更小，以免出現震盪導致到不了目標點的現象。

**2.2**

我一開始的delta(x)是直接利用當下位置的6D pose與目標位置的6D pose相減而來，但是這樣得到的結果會超級差，在我將delta(x)印出來後發現在疊代的途中，僅有x,y,z會變小，角度都不太會收斂，不太確定是什麼原因，但我認為是因為角度無法直接相減所求得，需要利用上述類似向量的方式才能將其考慮進去。

另外，調整max iteration 與 stop threshold的效果很好，若僅使用預設值的話，最終的成果都不太理想，無法得到較高的分數，但直到我將值設置為0.01時總分能接近40分。

**3. manipulation.py**

**3.1**

1. **get\_src2dst\_transform\_from\_kpts:**

首先先將各自不同frame的dis 與 src keypoints 利用extrinsic轉到一樣的座標(world frame)，接著利用SVD分解的特性找出轉移與平移矩陣，以此將src的關鍵點轉移到dis 上。值得注意的是，本來利用SVD尋找相同key point的方法必須在同個frame下，因此，這編譯開始才需要使用外參將基於各自相機frame的關鍵點轉置世界座標。

1. **Template\_gripper\_transform:**

這邊的transform為從object frame到 gripper frame ，而這個是從助教給的.jason而來的，並不用自己求得。另外，此用途為之後做夾取時(robot\_dense\_action)會用到，到時會需要用到它的反矩陣，也就是gripper frame to object frame，以此來獲得該物件要被夾的地方位於gripper pose中為何點。

**3.2**

**Minimum number of keypoint:**

至少需要3個點，由於此robot的操作是位於三維空間中，因此至少需要三個點才能成功計算出轉移矩陣。

**3.3**

我認為造成誤差的最大原因為3個圖的keypoint圖點的不夠準確造成的，儘管這三張圖已經算是滿大的圖了，但是是人的話難免會有誤差，只要稍微有偏差就會導致pose matching不準確。改進的方式的話為由ML的方式來尋找物品的keypoint，以此來取代人為手動給予keypoint，這樣能確保同個物品給的keypoint相同率較高，進而提高matching accuracy。