CMPUT 414: Winter 2016

Lab 3, due date Jan 30, 2016, 11:55 PM

Inverse Kinematics

In this lab, you need to implement the Cyclic Coordinate Descent (CCD) algorithm to solve the IK problem. You will code this method by using the Autodesk MotionBuilder (MoBu) python API.

Creating IK Chain:

- You do not need to create it yourself. Use the script **chain_creator.py** (part of Lab 3 resources) to create the chain and the goal. Your method will solve the IK problem for a chain of **n >= 2** nodes. The length of each bone **l > 0**.
- You can manually move the goal around the scene to test your method. When marking your method, I will have a script that will move the goal to different parts of the scene, and then call your method to solve the IK.

Suggested CCD exit conditions:

- Maximum number of iterations: 10 times number of bones in chain
- Minimum distance between end node and goal: 0.01

Deliverable:

I. Python script file with your code. Your file should be named **ik.py**, and it should respect the following rules:
   a. It should have a method with the following signature: **ccd(goal,chain_base)**. The parameter **goal** represents the end effector, while **chain_base** is the first joint of the chain.
   When testing you method, I should be able to call it by running the following lines of code.

```
from pyfbsdk import *
import ik

goal = FBFindModelByLabelName('Goal')
chain_base = FBFindModelByLabelName('Node')

ik.ccd(goal,chain_base)
```

I will make sure your script will be inside one of the directories from my PYTHONPATH.

b. Place any other method you may have created inside the same file. Do not submit any other files.

Restrictions: You cannot use MoBu's chain IK constraints for this exercise.

This lab will be marked as follows:

I. Correctness. Does it produce the expected output? **(50/100)**
II. Documentation. Did you comment important part of your code? **(20/100)**
III. Conciseness and Readability. Is your code clear and objective? Does it perform only necessary operations? Are your methods cohesive? **(20/100)**
IV. Deliverable. Did you follow the specification for the deliverable? **(10/100)**

Help:

MoBu_cheat_sheet.pdf (part of Lab 3 resources) contains some important tips that can help you get started on MoBu's Python API.

Hints:

Vector alignment: Given 2 unit 3D vectors **a** and **b**, we can rotate **a**, so that its orientation matches **b** by

1. Find axis and angle:

$$v = a \times b$$

$$s = ||v||$$

$$c = a \cdot b$$

2. Rotation matrix is given by:

$$R = I + [v]_\times + [v]_\times \frac{1-c}{s^2}$$

Where

$$[v]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

You can find the source code for the equation above in **math_utils.py** (located in Lab 3 resources). You can copy and paste that code into your solution, if necessary.