CMSC425 Spring 2019
Homework 2: More geometric exercises
COMMON ERRORS

The most common error on all problems was to get the sequence of transformation matrices wrong – to get a translation and rotation switched from what they should be. Generally you want to rotate or scale before translation. Rotations and uniform scales commute so order doesn't matter.

On problems to rotate or scale around a point p:

Always to the negative translation -p first to get t the origin, then the rotations and scale, then +p back. So it's T_p * M * T_(-p) * point

On problems to rotate a point at the end of a skeleton:

Generally you rotate, then translate, then rotate, then translate through the joints. Rotate a joint, move to the next joint, rotate it, etc. You mix the translations and rotations: M = R_j1 * T_to_j1*R_j2 .

You don't do all the translations after the rotations - they mix, in MC= T1*R1*T2*R2*T3*R3..

Part a. Warm up problems
1. Given this idea, show the value of q after multiplication, and then after normalization.
MOST GOT THIS RIGHT

2. In 2D two points can define an axis aligned square – see Mount 8, colliders, figure 1a with p+ and p-. Can two points in 3d define an axis aligned box – the solid equivalent of a square? What's the length of a side for the box?
MOST GOT THIS RIGHT, BUT IT'S SQRT(3), NOT SQRT(2) IN 3D.

3. Assuming we have a 2 by 2 square centered at the 2D homogenous point p=(1,3,1), give a sequence of homogeneous matrices to rotate the square 45 degrees clockwise around its center, and show the result. Where the point after applying the sequence (eg, its coordinates)?
COMMON ERROR:
Translating (1,3,1) to the origin starts with the translation (-1,-3,-1), then the rotation, then translating back with (1,3,1). Most common error was to reverse this.

4. Do similar actions for a box in 3D centered at p=(1,3,1,1), but this time do the following. Scale the box by 2 in the y direction, so it has a preferred direction, and rotate it by 45 degrees around the x-axis and then 30 degrees around the z-axis.
COMMON ERROR:
A confusion that's the problem's fault is that generally you translate back to the origin first before the scale, but the question wasn't clear on this.

5. Most were ok.

6. Give a Unity command to rotate 30 degrees around the vector <3,2,1,0>. Just one line … (Two if you define the axis in advance).
COMMON ISSUE
rotateAround does the work in the world coordinate system, angleAxis in the current.

If you got a point taken off for rotateAround we'll see about giving it back.

7. A final example of box rotation. This time we'll start with a box that is 1x1 located with its bottom center at the origin, with up along the y axis.

New version: Give matrices to scale the box by 2 in the y direction, and then rotate It so you have the old X, Y and Z axes align respectively with the new directions u=<1,1,1>, v=<1,-2,1> and n=<3,0,-3>. (Note that to apply the approach in the notes you need an orthonormal set of axes. The old axes are X=<1,0,0>, Y=<0,1,0> and Z=<0,0,1>.
COMMON ERRORS
a) Failing to normalize u, v AND n before putting them in the matrix. Not <1,1,1> but <1,1,1>/sqrt(3).

b) Transposing the rows and columns. Normalized u, v and n should be the columns of the new matrix, not the rows.

Part b. Applications
1. **Kinematics.** The arm has a base (b) and three joints (j0,j1,j2). In the bind or rest configuration the first three line up in the y direction, the second two in the x direction. Each of the

(a) Express the three local pose transformations as 3 x 3 homogeneous matrices T1, T2, T3.
MOST GOT THIS RIGHT

(b) Show that by multiplying these matrices you get one M so that $p_{[b]} = M * p_{[j2]}$. Verify your result by showing M works on the given values for p.
MOST GOT THIS RIGHT

(c) Create three homogeneous rotation matrices R1, R2, R3 that represent rotations for each joint as given in the diagram on the right. You don't have to compute the cosines and sines, or multiply the matrices – just give the matrices using the standard formula for rotations.
MOST GOT THIS RIGHT

(d) Give but do not calculate the combined matrix MC that represents the full transform that takes $p_{[j2]}$ into $p_{[b]}$ with the local pose and rotation transformations. All that's required is the sequences of matrix multiplications needed to compute MC – show that, but don't multiply.
COMMON ERROR:
To

2. **Plane sweep algorithm**.

a) Given a set of n discs (filled in circles) in the 2D plane, and you can assume in the upper right quadrant (all points positive in x and y), pseudocode an algorithm to find and report all intersections in better than $O(n^2)$ time. You can assume that the number of intersections is small relative to n^2, say k, so you don't have n^2 in just reporting intersections.

COMMON ERRORS

A number of you sorted into a list L by the x coordinate, and then compared L[i] and L[i+1]. But you need to compare more since a circle may intersect more than the adjancent ones.

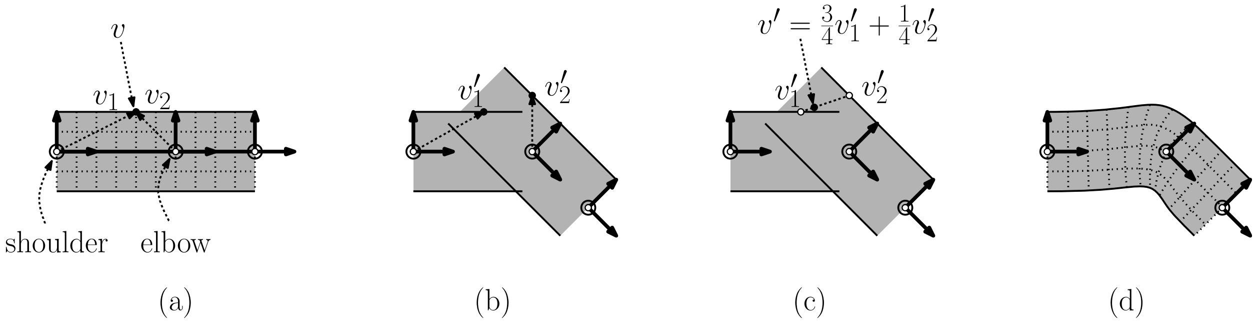
On the other hand, you don't want to compare circle L[i] to all that follow, that's O(n^2) Need to keep active list of circles that overlap in x as you sweep left to right.

b) Do in 3d.

COMMON ERROR:

Not noting that you can't sort circles on a plane, there's no natural order.

3. **Linear-blend skinning**.



For this case, of two joints j1 (shoulder) and j2 (elbow), and bind/resting pose $T_{[j1\leftarrow j2]}$ given, expand the given linear combination for v' out so you have a formula for v' in terms of the rotation matrices for the shoulder and elbow (call them M1 and M2), the bind pose transform $T_{[j1\leftarrow j2]}$, the two weights and the original points v1 and v2. Use general weights w1 and w2, not the specific constants in the example.

COMMON ERRORS

The key is that both v1 and v2 both need to be transformed into the common coordinate system j1 – otherwise you're averaging points in different systems.

So –

> *if you rotate v1 by joint rotation M1, you need to rotate v2 by M1, too*
> *you need to translate v2 into coordinate system j1 by T*
> *you can't translate v1 by T.*