CMSC425 Spring 2019
Homework 2: More geometric exercises- CORRECTED (Qa7 and Qb1) March 18th
Assigned Tuesday, March 12th
Due by midnight on Tuesday, March 26th
Submit PDF on Elms (no paper required)

Part a. Warm up problems
These are intended as straightforward use of basic formulas to review and debug your understandings of the concepts. You may use Octave or another vector calculator, but should show the steps you use. Notice that the problems mix ordinary text (p=(1,2)) and MS Word equation mode ($v^{\perp}$). You can use either mode, or Latex, if the answer is clear.

1. Something we did not do in class, but is fairly straightforward. A homogenous point is represented by a four-tuple <x,y,z,w>, with w=1. Under most operations we keep w at 1. But, in some operations we do multiply w by a number. Eg, we might get p=<2,1,1,1> multiplied by 2 to get p2=<4,2,2,2>. But we do a special normalization of homogeneous points, dividing by w, which means that p2=<4,2,2,2>= <4/2,2/2,2/2,2/2>=<2,1,1,1>=p. In effect, multiplying a homogenous point by a scalar has no effect, and we always return w back to 1. (ADDED: A quick reading on homogenous coordinates and why we use them: https://prateekvjoshi.com/2014/06/13/the-concept-of-homogeneous-coordinates/)

Given this idea, show the value of q after multiplication, and then after normalization.

$$q = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

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?

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)?

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.

5. Give Unity Vector3 methods to carry out problem (4).

6. Give a Unity command to rotate 30 degrees around the vector <3,2,1,0>. Just one line …

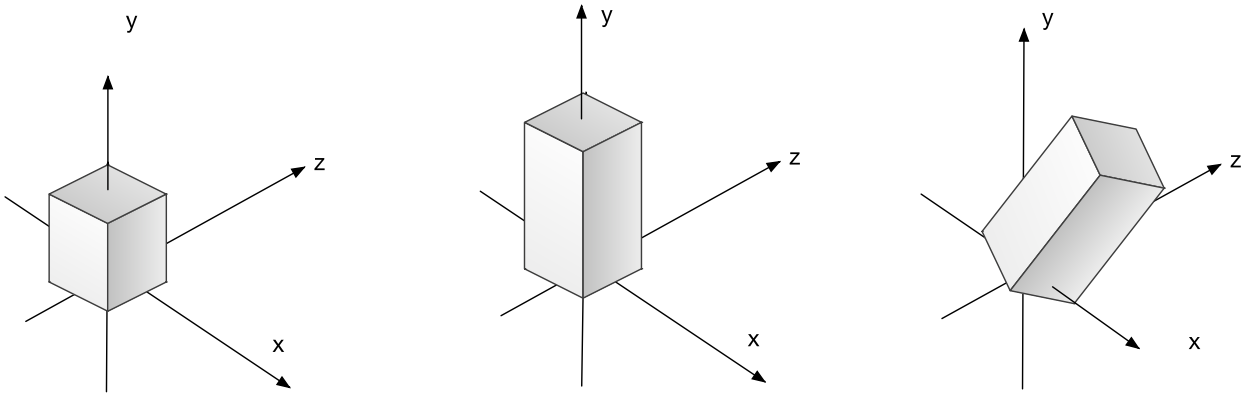
7. A final example of box rotation. This time we'll start with a box that is 1x1 located with its bottom center at the origin, with up along the y axis.

Correction: The original version of this problem is not solvable by techniques used so far this semester so it's been made optional extra credit, and a new version added. Do the new version instead of the original. The original problem had homogenous vectors, the new one doesn't, but that doesn't really change much.

This problem is supposed to be a straightforward application of the ideas in this handout from Feb. 25th. Once you figure out those notes, solving the problem should not take long. If it does you're using the wrong approach.
https://www.cs.umd.edu/class/spring2019/cmsc425/handouts/CMSC425RotationNotes.pdf

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>.

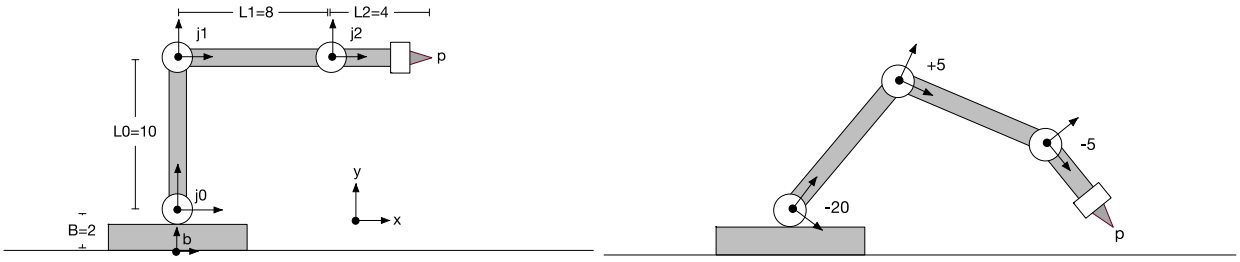
Original version (now optional): Give matrices to scale the box by 2 in the y direction, and then rotate it so the y axis aligns with the vector n=<1,1,1,0> and the old vector v=<1,0,1,0> aligns with the new vector u=<1,-1,1,0> .



Part b. Applications
These are intended as applications of the formulas to game design problems.

1. **Kinematics.** A digitizer arm is an unpowered robot-style arm that has a pointer at the end. You move the pointer in space to a location, and then record the location in 3D. Locating a set of points gives you the vertices in a mesh. We're cheating here in that you really care about inverse kinematics (move the arm to point at something and record the rotations), but for this problem we'll be doing forward kinematics in 2D.

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 coordinate systems is standard with x to the right, y up. See the diagram on the left.



Joint j0 is B=2 units above the base; joint j1 is 10 units above j0; joint j2 is 8 units right of j1; and the point p is 4 units right of j2. The point p is defined in ~~j0~~ **j2** coordinates as $p_{[j2]} = (4,0,1)$

Define the three local pose transformations as

$T_{[b \leftarrow j0]}$ is from j0 to b
$T_{[j0 \leftarrow j1]}$ is from j1 to j0.
$T_{[j1 \leftarrow j2]}$ is from j2 to j1.

If you transform the point p into base coordinates you get $p_{[b]} = (12,12,1)$.

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

(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.

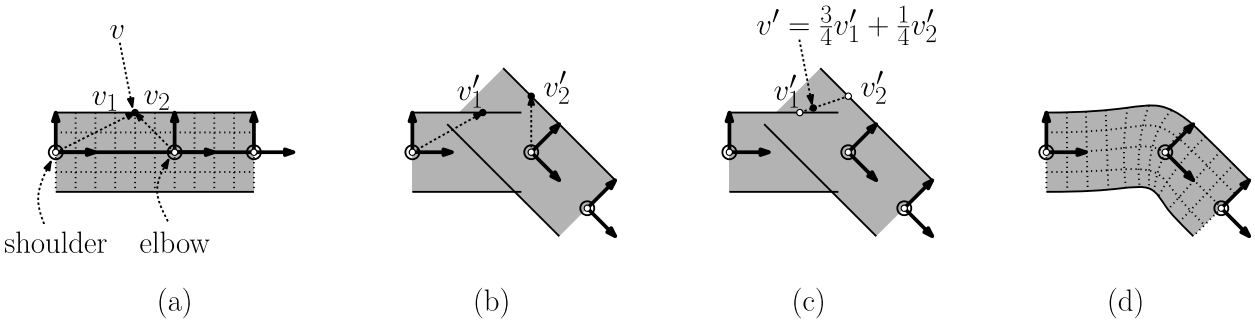
(c) Create three homogeneous rotation matrices R1, R2, R3 that represent rotations for each joint as given in the diagram on the right. You don't have to compute the cosines and sines, or multiply the matrices – just give the matrices using the standard formula for rotations.

(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.

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.

b) How would you generalize the plane sweep algorithm to 3D and a set of spheres? You don't need to pseudocode a full algorithm, just give a description of what is different from 2D.

3. **Linear-blend skinning**. In linear blend skinning you attach one skin vertex to two joints, move the vertex by both transformations, and then report the final position of the vertex by the linear (convex) combination of the two transformed vertices. Below is an example where v is the vertex and in the resting pose is at location v1 in the shoulder coordinates, and v2 in the elbow coordinates. Then v1' and v2' are the transformed points as shown below, and v' is the final reported position as the elbow rotates. Here we have the two weights w1=3/4 and w2=1/4.



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.