1 Introduction

In this project we are going to explore how matrices can be used to model the kinds of spatial transformation that are commonly used in robotics, computer vision and graphics. The assignment is split into 2 parts, in the first part you will be modeling the upper half of an articulated humanoid robot moving around in 2D, the second part involves modeling the action of a camera moving around a 3D scene.

2 Robot

The figure below shows the system you will be modeling. Really what we are doing is modeling our humanoid robot using a series of seven boxes that need to be rotated translated and scaled in the right way to produce the drawing.

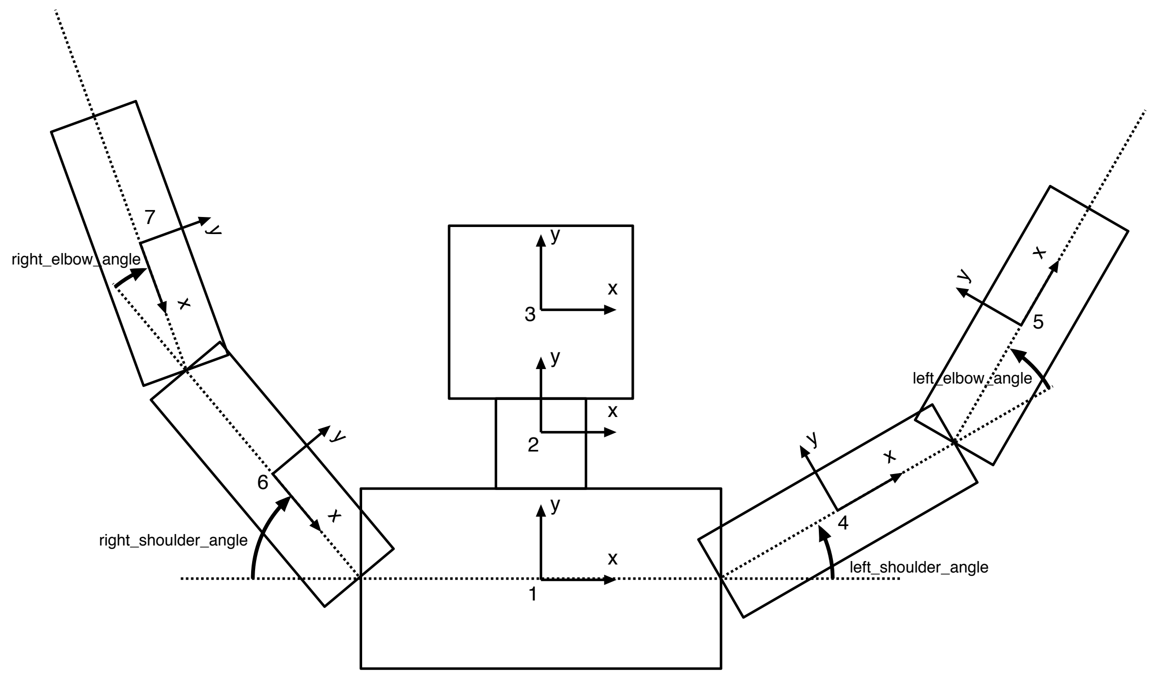

Figure 1: Model of an articulated robot torso

The dimensions of the boxes that comprise the robot model are as follows: the shoulder girdle module is 4 units wide and 2 units tall, the neck is a unit square, the head is a square 2 units on side. The forearms and upper arms are 3 units long and one unit high.

You are provided with a Matlab script entitled RobotScript.m that runs the animation, your ultimate job is to complete the code in this script to see the resulting robot in action. Before you can do that you will need to write the following Matlab functions:

2.1 Scale2D [4 points]

```
function g = Scale2D(sx, sy)
```

Returns the following 3×3 matrix representing a scaling in the plane: $\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$

2.2 Translation2D [4 points]

```
function g = Translation2D(tx, ty)
```

Returns the following 3×3 matrix representing a translation in the plane: $\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$

2.3 Rotation2D [4 points]

```
function g = Rotation2D(theta)
```

Returns the following 3×3 matrix representing a rotation in the plane, note that the input theta is specified in degrees: $\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$

3 RobotScript [28 points]

Once you have written these helper functions you should look at the script entitled `RobotScript.m` there you will find a template of the code you need to write. You will need to fill in the sections marked with question marks (??). You must use the variable names that we have selected so that we can auto grade your work but you can add extra variables and more lines of code if you wish.

As part of this assignment you will need to correctly model the coordinate transformation between the frames shown in the figure, remember that you can do this by breaking each transformation down into a sequence of simpler transformations with the aid of an appropriate figure. You may find it easier to proceed by concentrating on getting one box drawn in the correct place at a time instead of trying to complete the entire script at once.

If you do all correctly you will see the robot waving it's arms as shown in Figure 2.

4 Perspective Projection

For this section you are being asked to write code to model coordinate transformations in 3D. Then you will write code to model the action of a camera which projects the 3D world onto a 2D image using perspective projection. Your ultimate job is to complete the script entitled `DrawTowerScript` which models a movie camera flying around a tower. Here are the coding tasks you need to complete:

4.1 Scale3D [4 points]

```
function g = Scale3D(sx, sy, sz)
```

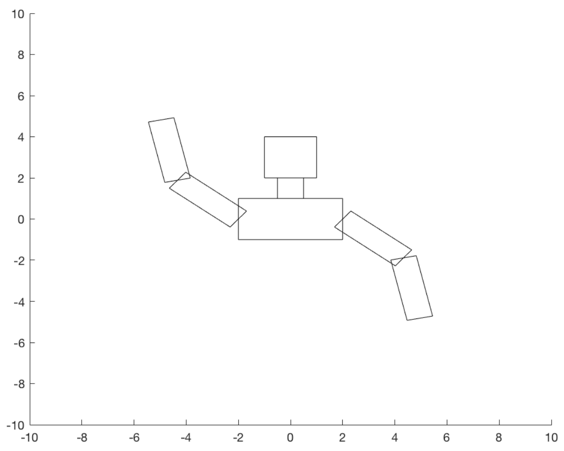

Figure 2: One frame from robot animation

Returns the following 4×4 matrix representing a scaling in 3D: $\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

4.2 Translation3D [4 points]

```
function g = Translation3D(tx, ty, tz)
```

Returns the following 4×4 matrix representing a translation in 3D: $\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$

4.3 Rotation3D [6 points]

```
function g = Rotation3D (axis, angle)
```

Returns a 4×4 matrix representing a 3D rotation in homogenous coordinates. It takes two parameters a single character denoting the axis and a real number denoting the rotation angle in degrees. That is `Rotation3D ('x', 90)` should produce a rotation about the x axis by 90 degrees. You will probably find Matlabs switch construct useful to switch between the various cases. The following equations should help:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

4.4 PerspectiveProjection [6 points]

```
function out = PerspectiveProjection(fv)
```

This function is similar in spirit to the function `ApplyTransform` which we have provided as part of this assignment. The `PerspectiveProjection` function takes as input a structure representing a 3D shape with two fields corresponding to the faces and the vertices. It returns as output another shape structure with the same set of faces but where the vertices have undergone a perspective projection. More specifically the vertices field of the input fv structure is an array with 3 columns the first column corresponds to the x coordinates of the vertices the second column to the y coordinates and the third to the z coordinates. In the output structure the vertices matrix should have the same number of rows but only 2 columns The first column should correspond to the x coordinates divided by the z coordinates, (x/z) and the second to the y coordinates divided by the z coordinates, (y/z). You may find Matlabs *pointwise* division operation , ./, useful here.

4.5 DrawTowerScript [20 points]

Once you have written these helper functions you should look at the script entitled `DrawTowerScript.m` there you will find a template of the code you need to write. We have provided the code to draw the first base block in a tower, the dimensions of this block are 8 units along the x axis, 6 units along the y axis and 4 units along the z axis where the z-axis corresponds to the vertical dimension. The tower should include 2 more blocks, a block half the size mounted on top of and centered with the first block and a wedge block with dimension 4 units along the x axis, 3 units along the y axis and 2 units along the z-axis that sits on top of the second block as shown in Figure 4.5.

To finish this code you will need to compute appropriate transformations between these blocks and the camera frame and correctly invoke your `PerspectiveProjection` function. If you do all this correctly you will see the projection of a building onto a virtual camera as the camera circles the scene. Again you may find it simpler to start simple and then add things to the script.

4.6 Extra Credit [10 points]

For extra credit you must add another building to the scene beside the one we have already added. This tower must consist of at least 3 blocks and must utilize at least one shape that we did not provide you, that is, it can't just consist of boxes and wedges it should contain other

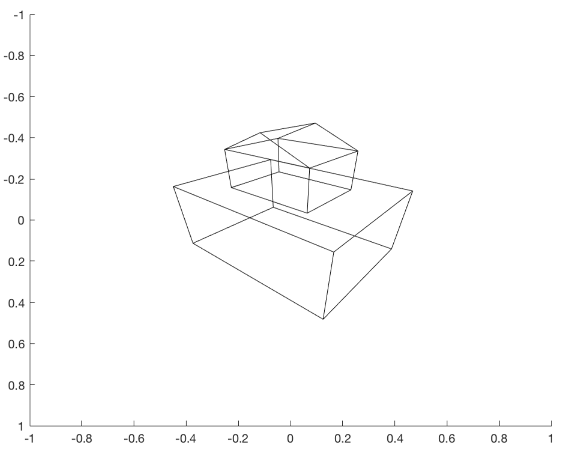

Figure 3: One frame from moving camera animation

shapes like frustums or pyramids or tessellated spheres. Include your animation code in a file called `ExtraCreditScript.m`. You can model it on the `DrawTowerScript`.

5 Submission Instructions

You will turn in all of the code for this project on Canvas. All of the code for the assignment should be in a single folder on your computer. You will create a zip archive of that folder by right clicking on the folder and selecting the appropriate action. Once the zip archive has been created you can upload it to Canvas. Note that the Canvas site will only accept files with a .zip extension. You should submit all of the functions and scripts we asked you to write or edit along with all of the functions we provided you with.