Template of Manipulator Short project: Skull tumor surgery

Authors: Pol Puigdemont & Marc Sardà

Team: G11A

Shared Link with the teacher:

I spect: 1) Pdf file, 2) Videos demostrating your successful task and your mlx file.

Notes. For better undestanding you can split the videos in the meaninful task.

Remember use the options of serial/link plot:

'workspace' for centering in the surgery task

'zoom' ... nice puma ratio aspect

'trail' .. to see the trajectory

etc..

See all at:

>> help SerialLink/plot

Table of Contents

The Robotic environment (10%)

Think that later on the enviroment will move to any place in a Univers Reference Frame {U}

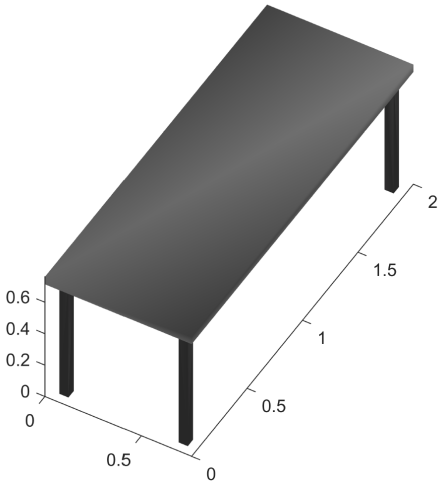
Use: 'c = uisetcolor' to chose your prefered colors

Operating table.

It can be raised, lowered, and tilted in any direction, and an auxiliary table for the tools. Define: Vertices and Faces and use 'patch' functions to model it. See help patch to find and example.

Think that later on the enviroment will move to any place in a Univers Reference Frame {U}

Expected results



Construction of the table:

```
clear
close all
clf
% creates a cube -> starting point for table and legs
v = [0 0 0;1 0 0;1 1 0;0 1 0;0 0 1;1 0 1;1 1 1;0 1 1];
f = [1 2 6 5;2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];

% scales the cube to compose the surface of the table
H=2;
W=0.9;
D=0.08;
v_1=[W 0 0;0 H 0;0 0 D]*v';

% elevate the table at a certain table height
table_height = 0.6;
table_vertices = transl(0,0,table_height) * [v_1 ; ones(1, 8)];

% obtain the max values for the x and y axis in the table
t_max_x = max(table_vertices(1, :));
```

```
t_max_y = max(table_vertices(2, :));
t_max_z = max(table_vertices(3, :));

% define the dimensions of a leg and it's offset wrt to the table limits
H_leg = 0.05;
W_leg = 0.05;
D_leg = 0.6;
leg_offset = 0.05;

% construct a leg scale from the cube vertices
leg_vertices = [W_leg 0 0; 0 H_leg 0; 0 0 D_leg] * v';

% create the four legs from the leg_vertices
leg_1 = transl(leg_offset, leg_offset, 0) * [leg_vertices; ones(1,8)];
leg_2 = transl(leg_offset, t_max_y - leg_offset - H_leg, 0) * [leg_vertices; ones(1,8)]
leg_3 = transl(t_max_x - leg_offset - W_leg, leg_offset, 0) * [leg_vertices; ones(1,8)]
leg_4 = transl(t_max_x - leg_offset - W_leg, t_max_y - leg_offset - H_leg, 0) * [leg_ve

% join elements in the table var
table = [table_vertices, leg_1, leg_2, leg_3, leg_4];
```

Table Reference

```
table_sep_x = 0;
table_sep_y = 0;
table_sep_z = 0;

% table reference frame wrt to the origin
T_table_0 = transl(table_sep_x, table_sep_y, table_sep_z); % can add rotation for examp

% operate the vertices
table = T_table_0 * table;

% update max values
t_max_x = max(table(1, :));
t_max_y = max(table(2, :));
t_max_z = max(table(3, :));
```

Auxiliar Table

```
% create the aux table
aux_table_H_scale = 0.25;
aux_table_W_scale = 0.45;
aux_table_D_scale = 1;

aux_table_sep_x = -1;
aux_table_sep_y = 2;
aux_table_sep_z = 0;

v3 = [aux_table_W_scale, 0, 0; 0, aux_table_H_scale, 0; 0, 0, aux_table_D_scale] * tabl
```

```
v3 = [v3; ones(1,40)];
```

AuxTable References

```
% aux table reference wrt to the main table
T_auxtable_table = transl(aux_table_sep_x, aux_table_sep_y, aux_table_sep_z);

% aux table reference wrt to the origin
T_auxtable_0 = T_table_0 * T_auxtable_table;

% operate the vertices
aux_table = T_auxtable_0 * v3;
```

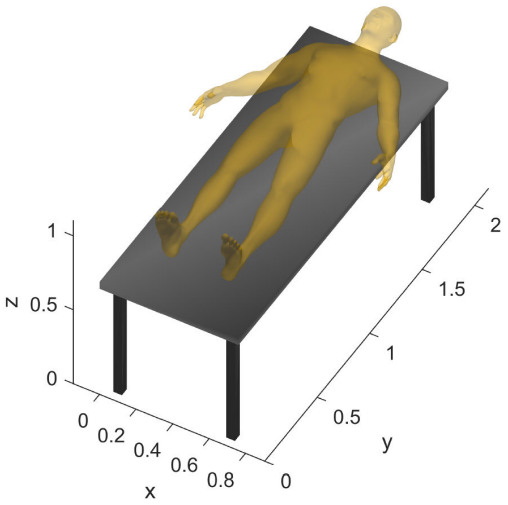
Tables Display

```
figure
% display tables (provisional: ask in lab how to structure faces to plot at once)
patch('Vertices',aux_table(1:3,1:8)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','bl
patch('Vertices',aux_table(1:3,9:16)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','b
patch('Vertices',aux_table(1:3,17:24)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','
patch('Vertices',aux_table(1:3,25:32)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','
patch('Vertices',aux_table(1:3,33:40)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','
patch('Vertices',table(1:3,1:8)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','black'
patch('Vertices',table(1:3,9:16)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','black
patch('Vertices',table(1:3,17:24)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','blac
patch('Vertices',table(1:3,25:32)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','blac
patch('Vertices',table(1:3,33:40)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','blac
grid on
xlabel('x');
ylabel('y');
zlabel('z')
axis equal
view([38.720 37.262])
hold on

% display the tables reference frames
trplot(T_table_0,'framelabel','T' ,'color', 'black','arrow','width',0.4);
trplot(T_auxtable_0,'framelabel','T' ,'color', 'black','arrow','width',0.4);
```

3D model of a human body

Situate the human model on the operating table.

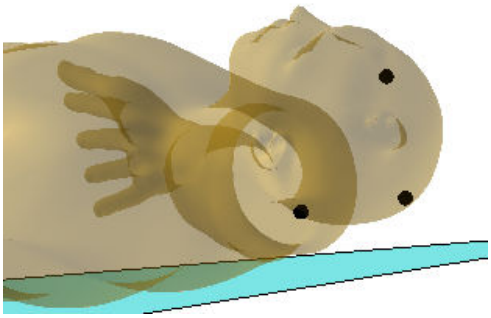
Expected results




```
%% load the human object
human = load('F_V_Humanbody.mat');

% Scale human
human_scale = 1;
human_vertices = human_scale .* human.Vh';

% Transformation of the position of the human
h_min_z = min(human_vertices(3, :));
body_z_faces_offset = 0.15;
T_body_table = transl(t_max_x/2, 0.5, t_max_z - h_min_z + body_z_faces_offset) * trotx(
T_body_0 = T_table_0 * T_body_table;
human_vertices = (T_body_0 * [human_vertices; ones(1, length(human.Vh))])';

b_min_z = min(human_vertices(3, :));

% Display the human
patch('Vertices', human_vertices(:, 1:3), 'Faces', human.Fh, ...
    'FaceColor', [0.9290, 0.6940, 0.1250], ...
    'FaceAlpha', 0.3, ...
    'EdgeColor', 'none', ...
    'FaceLighting', 'gouraud', ...
    'AmbientStrength', 0.15);
grid on
xlabel('x');
ylabel('y');
zlabel('z')
axis equal
view([38.720 37.262])
trplot(T_body_0,'framelabel','B' ,'color', 'r','arrow','width',0.4);
```

Fiducials

The Radiology Department before to take a Computer Tomography (CT) of the brain, fix three fiducials in the head of the patient for registering purpose, visit: https://en.wikipedia.org/wiki/Fiducial



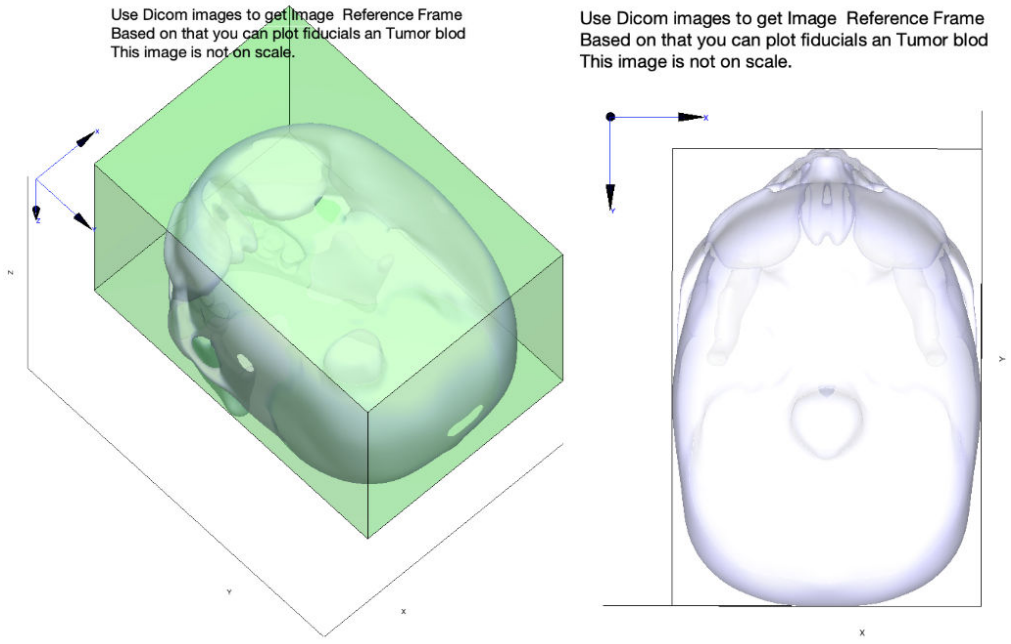
Dicom image vs Image Reference frame {I}

Get familiar with Dicom Images, Visit: https://www.imaios.com/en/Imaios-Dicom-Viewer#!

Use a container Box of the skull to infer the Image Reference Frame {I}

See: 6_Plot_Box_Cone.mlx and 7_Help_Image_RF_Containig_Box.fig to inspire yourselt

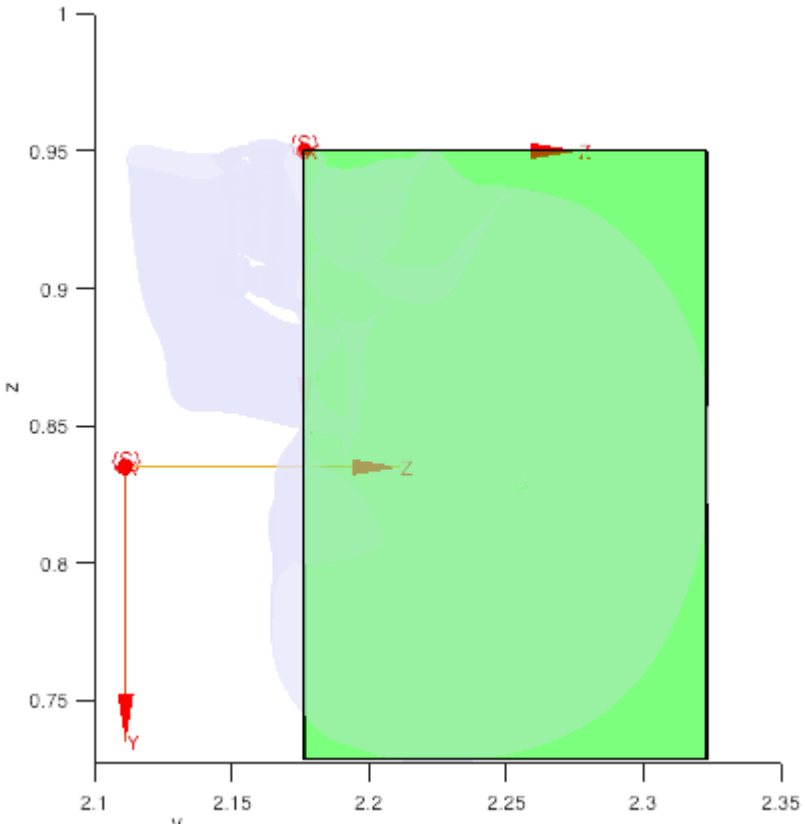
Expected results




```
skull = load("F_V_Skull.mat");

% Scale Skull
skull_scale = 1;
skull_vertices = skull_scale .* skull.Vs';
```

```
% Transformation of the position of the skull
skull_z_offset = 0.15;
T_skull_body = transl(0, 0, max(human_vertices(3, :) - max(skull_vertices(3, :))));
T_skull_0 = T_body_0 * T_skull_body;
skull_vertices = (T_skull_0 * [skull_vertices; ones(1, length(skull.Vs))])';
```

```
skull_cube = ([
    max(skull_vertices(:,1))-min(skull_vertices(:,1)), 0, 0;
    0, max(skull_vertices(:,2))-min(skull_vertices(:,2)) + 0.009, 0;
    0, 0, (max(skull_vertices(:,3))-min(skull_vertices(:,3))) / 1.55;
    ] * v');
T_I_skull = transl( ...
    -(max(skull_vertices(:,1))-min(skull_vertices(:,1)))/2, ...
    -(max(skull_vertices(:,2))-min(skull_vertices(:,2)))/2 - 0.009, ...
    0.065);
T_I_0 = T_skull_0 * T_I_skull;
skull_cube = (T_cube_0 * [skull_cube; ones(1,8)])';
clf
figure
hold on;
% Display the box
patch('Vertices', skull_cube(: ,1:3), 'Faces', f, ...
    'FaceVertexCData', hsv(6), ...
    'FaceColor', 'g', ...
    'FaceAlpha',0.3);

% Display Skull
patch('Vertices', skull_vertices(:, 1:3), 'Faces', skull.Fs, ...
    'FaceColor', '#E6E6FA', ...
    'FaceAlpha', 0.5, ...
    'EdgeColor', 'none', ...
    'FaceLighting', 'gouraud', ...
    'AmbientStrength', 0.15);

trplot(T_skull_0,'framelabel','S' ,'color', 'r','arrow',['length'],0.1);
trplot(T_cube_0,'framelabel','S' ,'color', 'r','arrow','length',0.1);
zlabel('z'); ylabel('y'); xlabel('x');
view([90.207 0]);
```

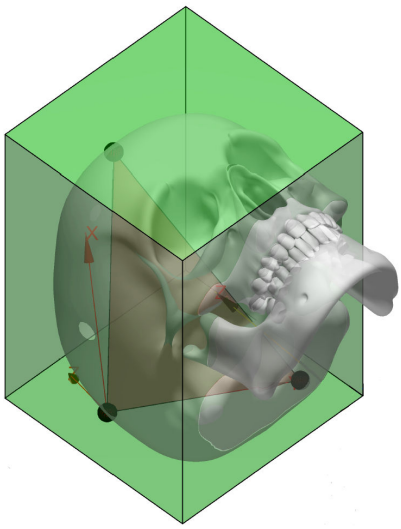


Fiducials wrt {I}

Use the Dicom images to place the fiducial relative to Image Reference Frame {I}.

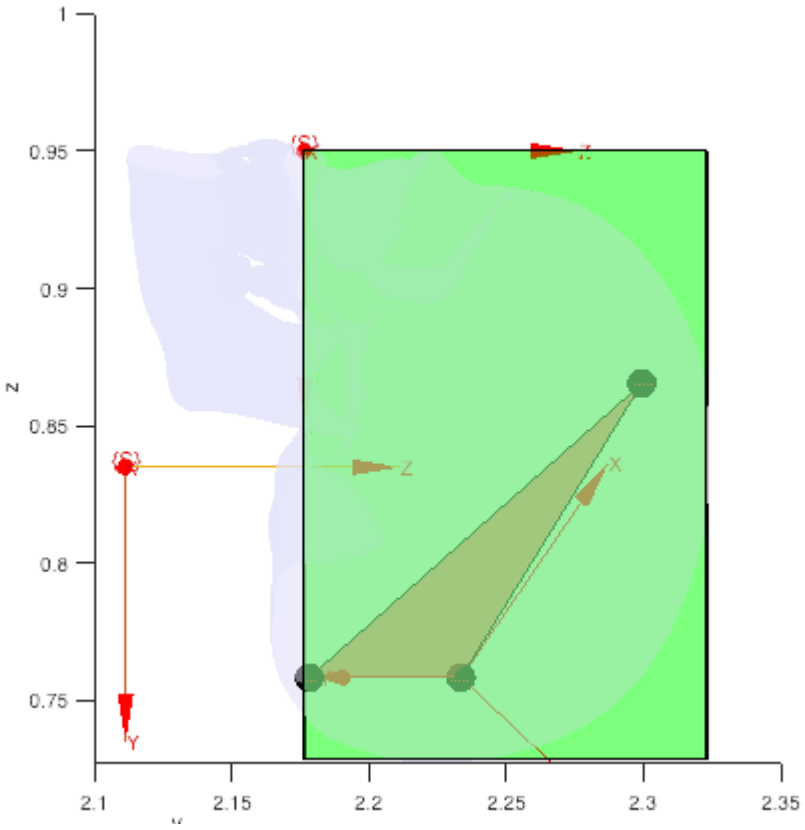
See: 5_Skull_pose_estimation.mlx and use the skull to make the exercise.

Expected results



```
hold on
fiducial_1 = T_I_0 * [0.1181; 0.1917; 0.0020; 1];
```

```
fiducial_2 = T_I_0 * [0.0268; 0.1917; 0.0569; 1];
fiducial_3 = T_I_0 * [0.0836; 0.0846; 0.1230; 1];

% Displays the fiducials
[X,Y,Z] = sphere;
r=0.005;        % 5 mm
surf(X*r + fiducial_1(1), Y*r + fiducial_1(2), Z*r + fiducial_1(3), 'FaceColor', [1 0 0
surf(X*r + fiducial_2(1), Y*r + fiducial_2(2), Z*r + fiducial_2(3), 'FaceColor', [1 0 0
surf(X*r + fiducial_3(1), Y*r + fiducial_3(2), Z*r + fiducial_3(3), 'FaceColor', [1 0 0
triangle = [fiducial_1(1:3), fiducial_2(1:3), fiducial_3(1:3)];
fill3(triangle(1, :), triangle(2, :), triangle(3, :)', 'r', 'FaceAlpha', 0.5)

% Estimates the position between the fiducials and the skull
AB = (triangle(1:3,1) - triangle(1:3,2)) / norm(triangle(1:3,1) - triangle(1:3,2)); % V
CB = (triangle(1:3,3) - triangle(1:3,2)) / norm(triangle(1:3,2) - triangle(1:3,3)); % V
Z_B = cross(CB, AB) / norm(cross(CB, AB));                                          % P
O_B_A = oa2r(AB, Z_B);                                                              % A
T_fiducials_0 = [O_B_A fiducial_2(1:3); 0 0 0 1];
T_skull_fiducials = inv(T_fiducials_0) * T_skull_0;

trplot(T_fiducials_0, 'length', 0.1, 'arrow', 'color', 'r');
```

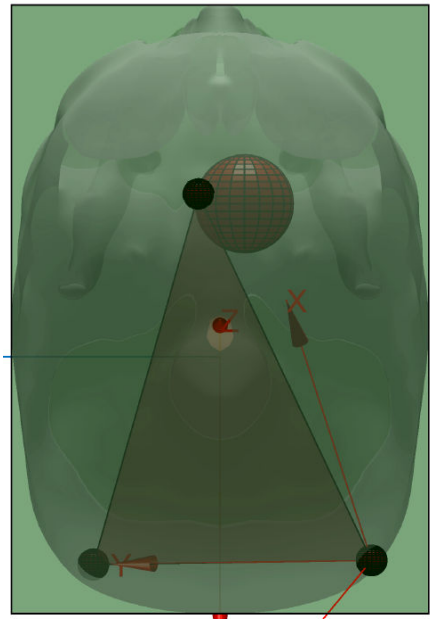


Tumor points wrt {I}

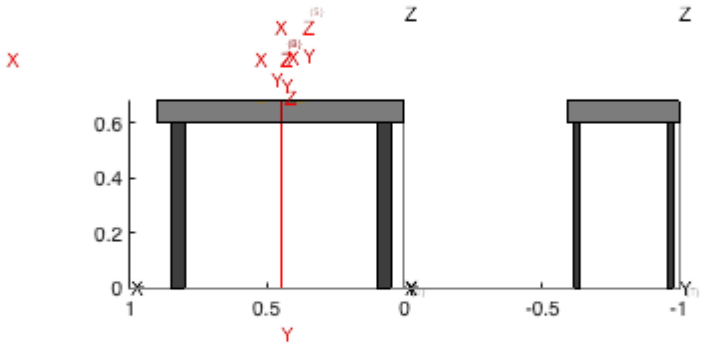
Use the Dicom images to get the points of the outer perimeter of the tumor relative to Image Reference Frame {I}.

You can simplify the tumor information by defining the center of mass and estimate an equivalent diameter.

Expected results



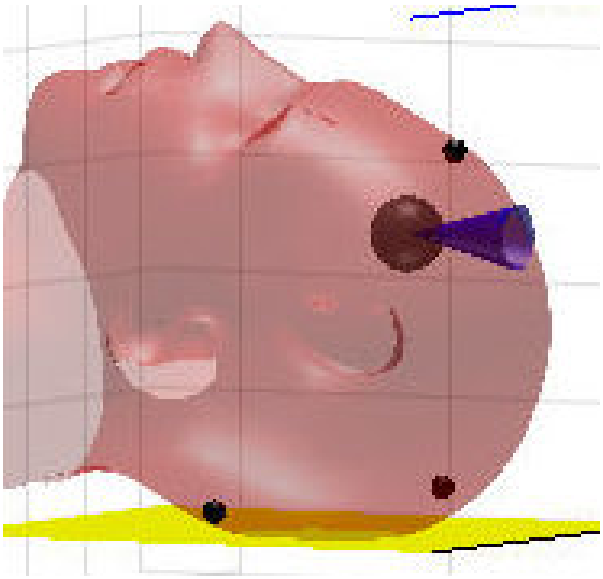
```
T_tumor_I = [0.0685, 0.0851, 0.1014, 1];

tumor = T_I_0 * T_tumor_I';

T_tumor_0 = transl(tumor(1), tumor(2), tumor(3));

% Displays the tumor
[X,Y,Z] = sphere;
r = 0.016;          % 16mm radius tumor
surf(X*r + tumor(1), Y*r + tumor(2), Z*r + tumor(3),'FaceColor',[1 0 0])
view(-180,0)
```



Fiducials and Tumor wrt Human Reference Frame

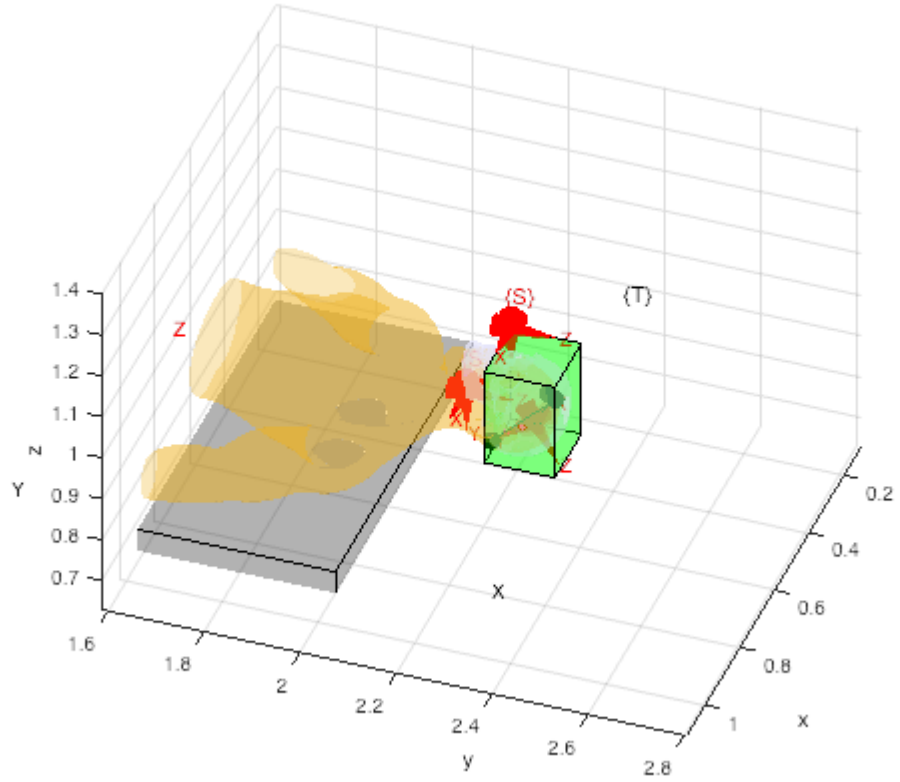
Place fiducial and tumor in the head of the human. You will have to re-do the containing box secction.

Expected results



```
% Display the human as we have constructed everything with reference frames
% since the beggining and we had the skull aligned we don't need to re-do
patch('Vertices', human_vertices(:, 1:3), 'Faces', human.Fh, ...
    'FaceColor', [0.9290, 0.6940, 0.1250], ...
```

```
    'FaceAlpha', 0.3, ...
    'EdgeColor', 'none', ...
    'FaceLighting', 'gouraud', ...
    'AmbientStrength', 0.15);
trplot(T_body_0,'framelabel','B' ,'color', 'r','arrow','width',0.4);
grid on
xlabel('x');
ylabel('y');
zlabel('z')
axis equal
xlim([0.101 1.107])
ylim([1.588 2.807])
zlim([0.628 1.405])
view([109.763 36.716])
```



First approach (10%)

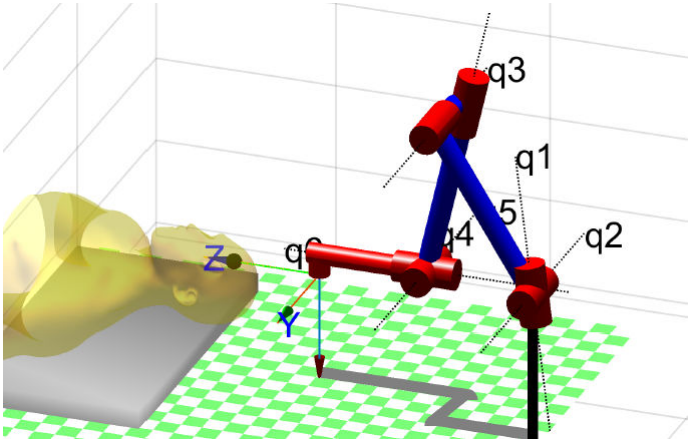
Asume that the ZX plane of the Robot is aligned with the plane of symmetry of the human body.

Robot manipulator

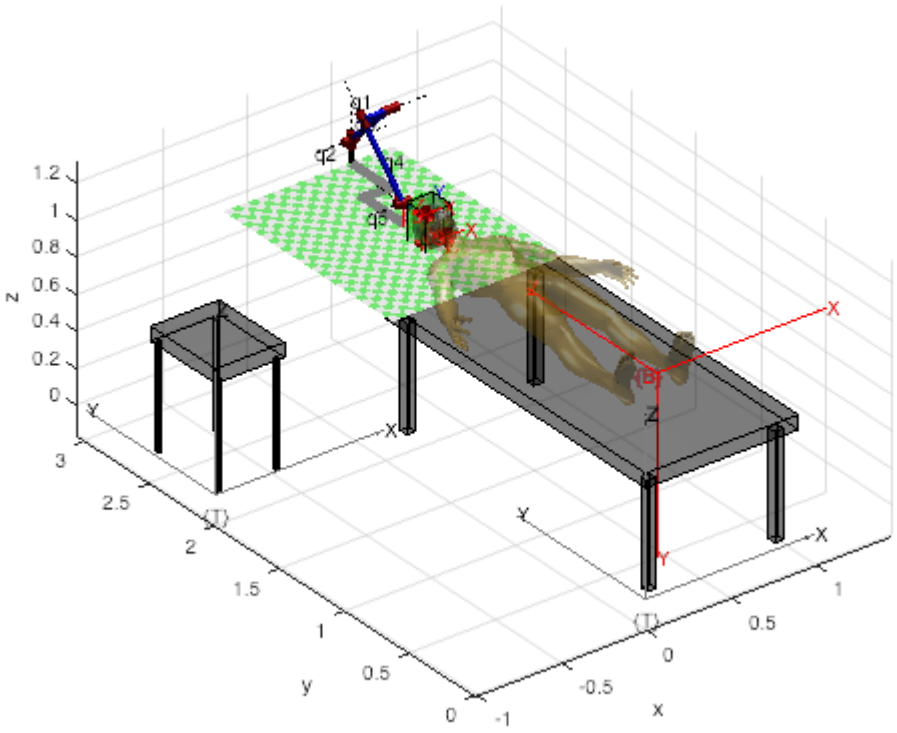
Consider the best position of manipulator to be nearby the operating table to warranty that the head is in the reachable work space. Use a Puma 560. Use p560.teach to play.

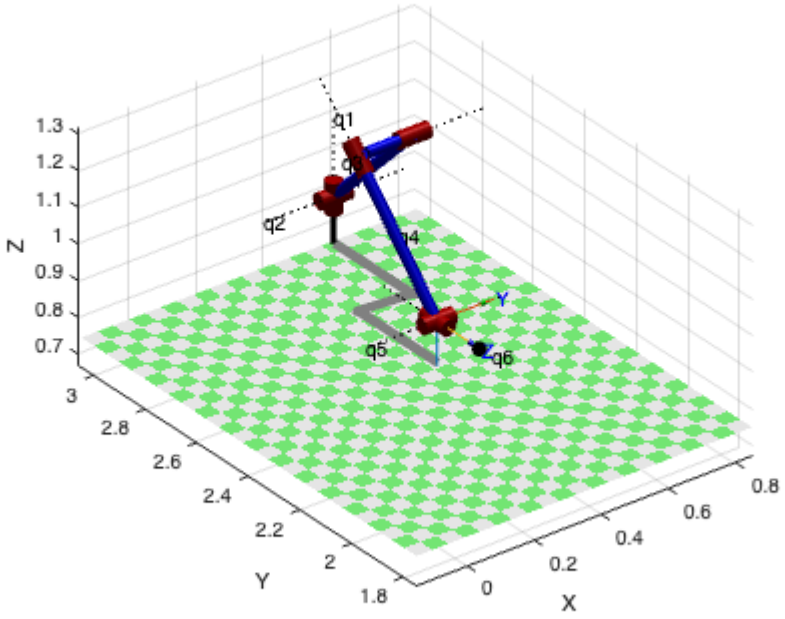
Use: p560.base & p560.tool to locate the Puma and add the tools.

Expected results

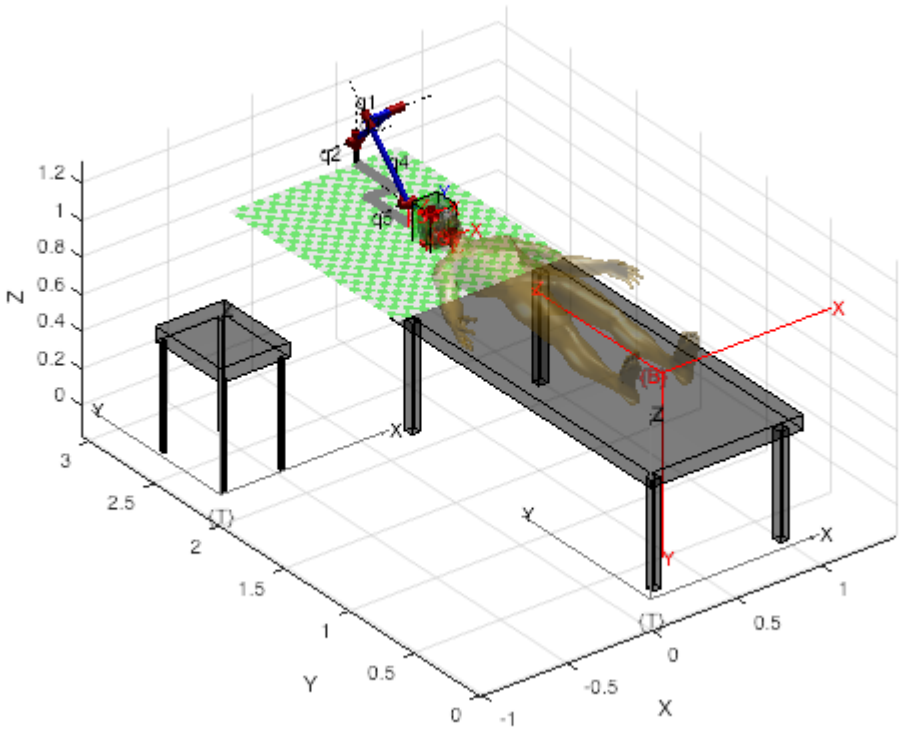



```
% Displays the robot (without any tool)
hold on
mdl_puma560
xbase = T_tumor_I(1) + p560.links(3).d;
zbase = T_tumor_I(3) + sqrt(2) * p560.links(2).a + 0.15;
ybase = T_tumor_I(2) - r;

p560.base = T_I_0 * transl(xbase, ybase, zbase) * trotz(-pi/2) * troty(pi/2);
workspaceX = [T_I_0(1, 4) - 0.5, T_I_0(1, 4) + 0.5];
workspaceY = [T_I_0(2, 4) - zbase/2, T_I_0(2, 4) + zbase];
workspaceZ = [T_I_0(3, 4) - 0.2, T_I_0(3, 4) + 1];

T_R_0 = T_I_0 * transl(xbase, ybase, zbase) * trotz(-pi/2) * troty(pi/2);

p560.plot(qn, 'zoom', 2.5, 'workspace', [workspaceX, workspaceY, workspaceZ], 'jaxes');
```





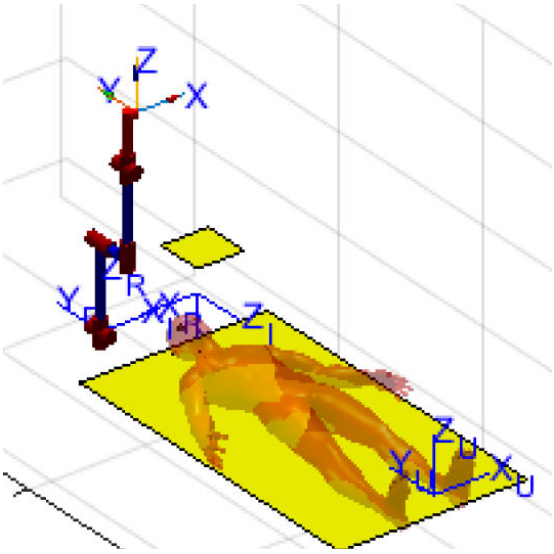

```
axis equal
```



Reference Frames

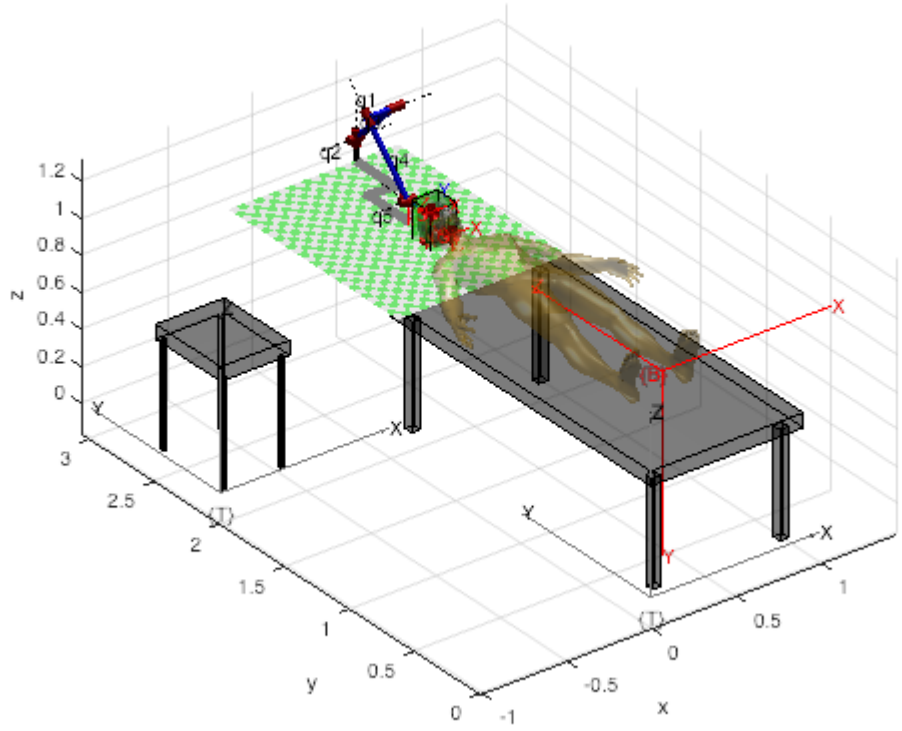
Display all necessary reference frame. Use best scale to see it.

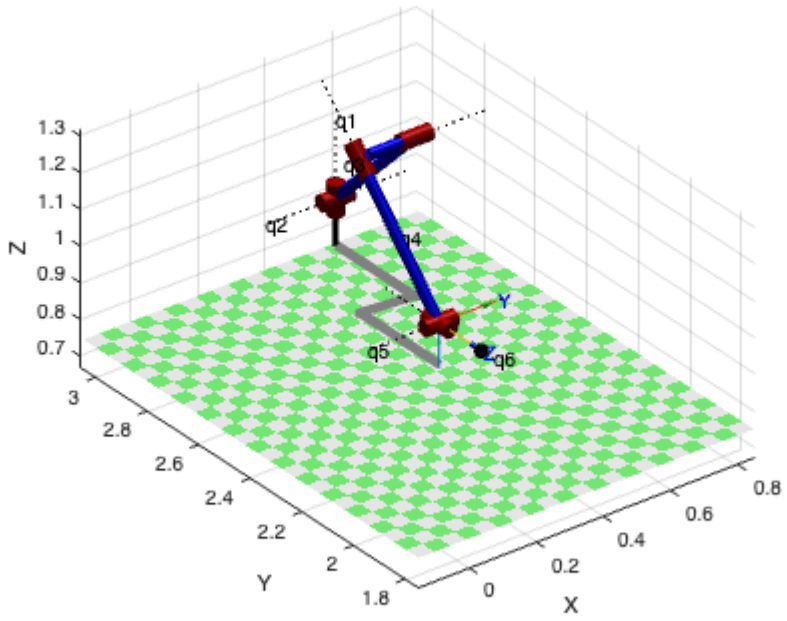
- {U} Univers [0 0 0]
- {R} Robot
- {I} Image
- {Tb} Table_body
- {Tt} Table tool
- {EE} End Efector
- others
- ...



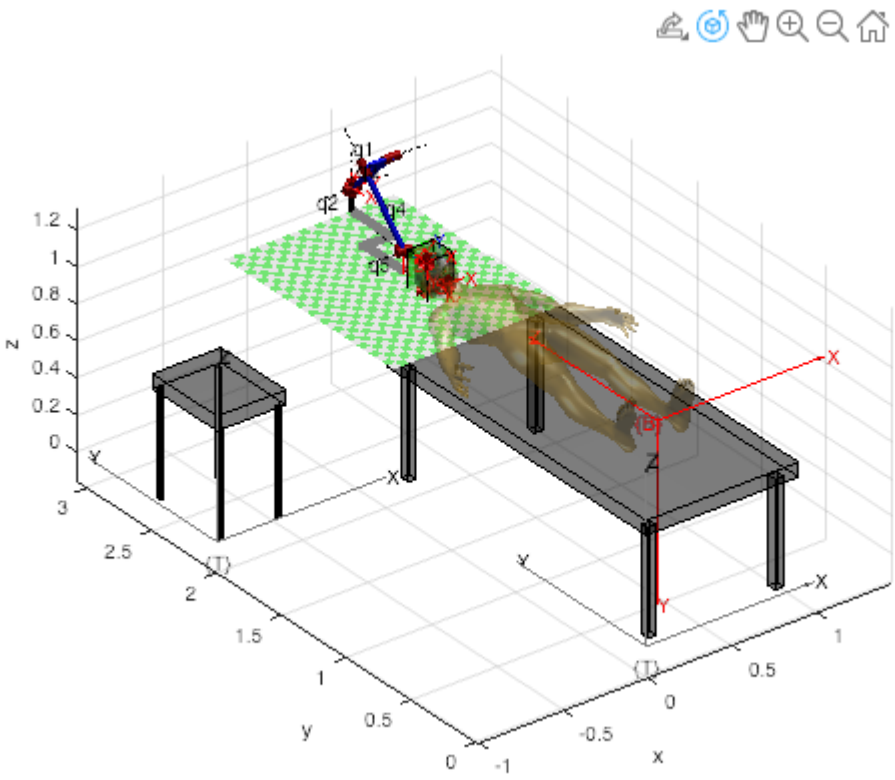

```
clf;
% complete plot of the space and reference frames up to this point
figure;
hold on;
% display tables (provisional: ask in lab how to structure faces to plot at once)
patch('Vertices',aux_table(1:3,1:8)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','bl
patch('Vertices',aux_table(1:3,9:16)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','b
patch('Vertices',aux_table(1:3,17:24)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','
patch('Vertices',aux_table(1:3,25:32)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','
patch('Vertices',aux_table(1:3,33:40)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','
patch('Vertices',table(1:3,1:8)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','black'
patch('Vertices',table(1:3,9:16)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','black
patch('Vertices',table(1:3,17:24)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','blac
patch('Vertices',table(1:3,25:32)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','blac
patch('Vertices',table(1:3,33:40)','Faces',f,'FaceVertexCData',hsv(6),'FaceColor','blac
% display the tables reference frames
trplot(T_table_0,'framelabel','T' ,'color', 'black','arrow','width',0.4);
trplot(T_auxtable_0,'framelabel','T' ,'color', 'black','arrow','width',0.4);

% Display the human
patch('Vertices', human_vertices(:, 1:3), 'Faces', human.Fh, ...
    'FaceColor', [0.9290, 0.6940, 0.1250], ...
    'FaceAlpha', 0.3, ...
    'EdgeColor', 'none', ...
    'FaceLighting', 'gouraud', ...
    'AmbientStrength', 0.15);
trplot(T_body_0,'framelabel','B' ,'color', 'r','arrow','width',0.4);

% Display the box
patch('Vertices', skull_cube(: ,1:3), 'Faces', f, ...
    'FaceVertexCData', hsv(6), ...
    'FaceColor', 'g', ...
    'FaceAlpha',0.3);
```

```
% Display Skull
patch('Vertices', skull_vertices(:, 1:3), 'Faces', skull.Fs, ...
    'FaceColor', '#E6E6FA', ...
    'FaceAlpha', 0.5, ...
    'EdgeColor', 'none', ...
    'FaceLighting', 'gouraud', ...
    'AmbientStrength', 0.15);

trplot(T_skull_0,'framelabel','S' ,'color', 'r','arrow',['length'],0.1);
trplot(T_cube_0,'framelabel','S' ,'color', 'r','arrow','length',0.1);

% Display the fiducials
surf(X*r + fiducial_1(1), Y*r + fiducial_1(2), Z*r + fiducial_1(3), 'FaceColor', [1 0 0
surf(X*r + fiducial_2(1), Y*r + fiducial_2(2), Z*r + fiducial_2(3), 'FaceColor', [1 0 0
surf(X*r + fiducial_3(1), Y*r + fiducial_3(2), Z*r + fiducial_3(3), 'FaceColor', [1 0 0
fill3(triangle(1, :), triangle(2, :), triangle(3, :)', 'r', 'FaceAlpha', 0.5)
trplot(T_fiducials_0, 'length', 0.1, 'arrow', 'color', 'r');

% Display tumor
surf(X*r + tumor(1), Y*r + tumor(2), Z*r + tumor(3),'FaceColor',[1 0 0])
trplot(T_tumor_0, 'length', 0.1, 'arrow', 'color', 'r');
trplot(T_R_0, 'length', 0.1, 'arrow', 'color', 'r');
trplot(T_I_0, 'length', 0.1, 'arrow', 'color', 'r');
% Display puma
p560.plot(qn, 'zoom', 2.5, 'workspace', [workspaceX, workspaceY, workspaceZ], 'jaxes');
```






```
grid on
xlabel('x');
ylabel('y');
zlabel('z');
axis equal
```



Transformations

Enumerate the transformation you will need.

```
% we can compute any combination knowing the rf wrt the origin
% using transform compounds
T_table_0; % table rf
T_body_0; % body rf
T_I_0; % image rf
T_fiducials_0; % fiducials rf
T_skull_0; % skull rf
T_auxtable_0; % auxtable rf
T_R_0; % robot rf
T_tumor_0; % tumor rf
```

Tumor points in Robot Frame.

Remember the Transform compound exercise

```
T_tumor_R = inv(T_R_0) * T_tumor_0
```

```
T_tumor_R = 4×4
         0   -1.0000         0    0.7607
    1.0000         0         0   -0.1501
         0         0    1.0000   -0.0160
         0         0         0    1.0000
```

Second approach: (25%)

Modify your code to repeat the exercise if the table with the patient is given as happend in the Rosa video.

To know the head relative pose with respect to the Puma Robot ...

```
open('3_Second_approach_Patient_pose.fig')
```

Surgery (55%)

Biospy

Prepare a script that perform a biopsy. Zoom in the scene and record a video with the best view.

Take into account:

- Use a tool that has the following Transformation: transl(0.05 0 0.25)
- Use 'trail' option of plot to visualize the trajectory.
- The speed of biopsy function that you design ought to be a parameter to satisfy the surgeons.

Answer these questions:

- Display in a figure the displacement, velocity and aceleration of the tool i End Efector Reference frame.
- How much enter the tool in the patient brain.
- What are the speed of the tool in Wordl Reference Frame.

```
%% put your code Here
```

Trepanation

Prepare a script that perform trepanation. Zoom in the scene and record a video with the best view.

Take into account:

- Use a tool that has the following Transformation: transl(0 0 0.2)
- Use 'trail' option of plot to visualize the trajectory.
- Place a 45º cone on top of the trepanation to better understand.
- See: 6_Plot_Box_Cone_fiducials.mlx. You will have to scale it. Play with transparency.

Answer these questions:

- Display in a figure the lineal displacement, velocity and aceleration of the tool in End Efector Reference frame.
- Display in a figure the manipulabitlity of the trepanation function either for translation and rotation.
- Find an alternate robot location for improving your manipulabitity.

```
%% put your code Here
```

Tumor burning

Prepare a script that perform tumor burning with the laser. Zoom in the scene and record a video with the best view.

You ought to think in an algorithm, that in order, fill up the tumor's equivalent sphere with small burning spheres of 4mm diameter.

Use a tool that has the following Transformation: transl(0 0 0.2)

Answer these questions:

- Display in a figure the lineal displacement, velocity and aceleration of the tool in End Efector Reference frame.
- How long it takes your burning function to burn the tumor

```
%% put your code Here
```