Add your own robot to the library

...a step by step guide...

The ARTE library has been designed to let the students add freely more robots to the library. The following example presents how to do so:

1 Obtain 3D CAD files to represent the robot

Robot manufacturers provide 3D CAD files for their robots. They are in different formats. Normally, you can either download separate files for each of the links or the whole robot arm in a single file. You will find different CAD formats available. You should choose the STL format preferably, in particular ASCII STL format. In case you have downloaded the arm as a whole, divide it in their independent links. Link0_base is the base, link1_base, link2_base … etc. Each of these links should be exported to STL (STereoLithography) format This is a common format for many CAD software. You can find more information under http://en.wikipedia.org/wiki/STL_(file_format)

For example, download the files for the ABB IRB6650S 90 390. Go into: http://www.abb.com/product/us/9AAC910011.aspx

Or navigate to www.abb.com→ products→ robotics.

→ **IRB 6640**
Payload: 130 - 235 kg
Reach: 2.55, 2.75, 2.8, 3.2 m

→ **IRB 6650S**
Pay load: 125 - 200 kg
Reach: 3.0 m, 3.5 m

→ **IRB 6660 for press tending**
Pay load: 130 kg
Reach: 3.10 m

→ **IRB 6660 for pre machining**
Pay load: 205 kg
Reach: 1.93 m

→ **IRB 760**
Pay load: 450 kg
Reach: 3.18 m

→ **IRB 7600**
Pay load: 150 - 500 kg
Reach: 3.5, 3.1, 2.8, 2.55, 2.3 m

Download CAD files. Beware of the robot versión you are downloading. You can either download the `compl` (whole arm), or `joint` (separate links). In this example we will download the joint version. Extract the files and save them under robot/abb/IRB6650S_90_390

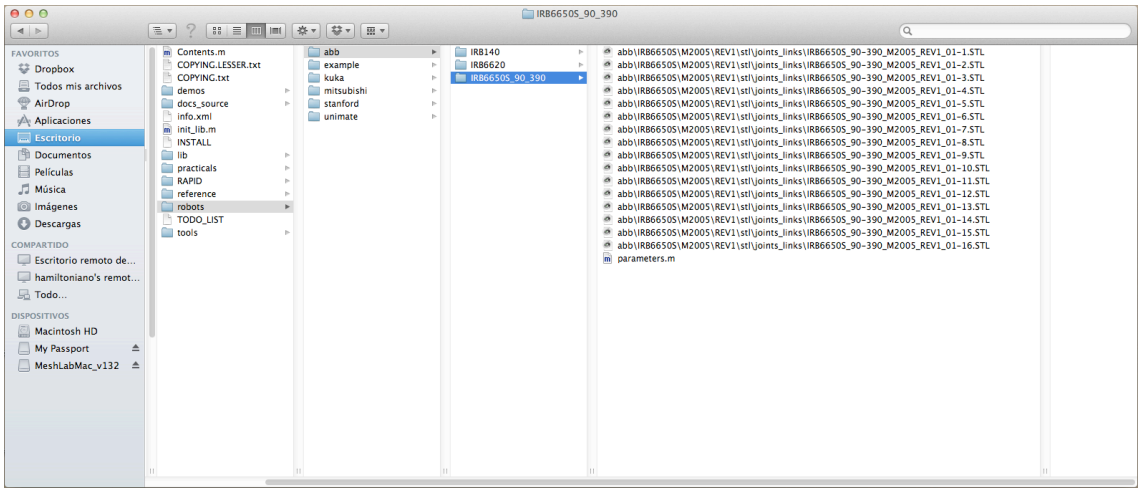
IRB 6650S

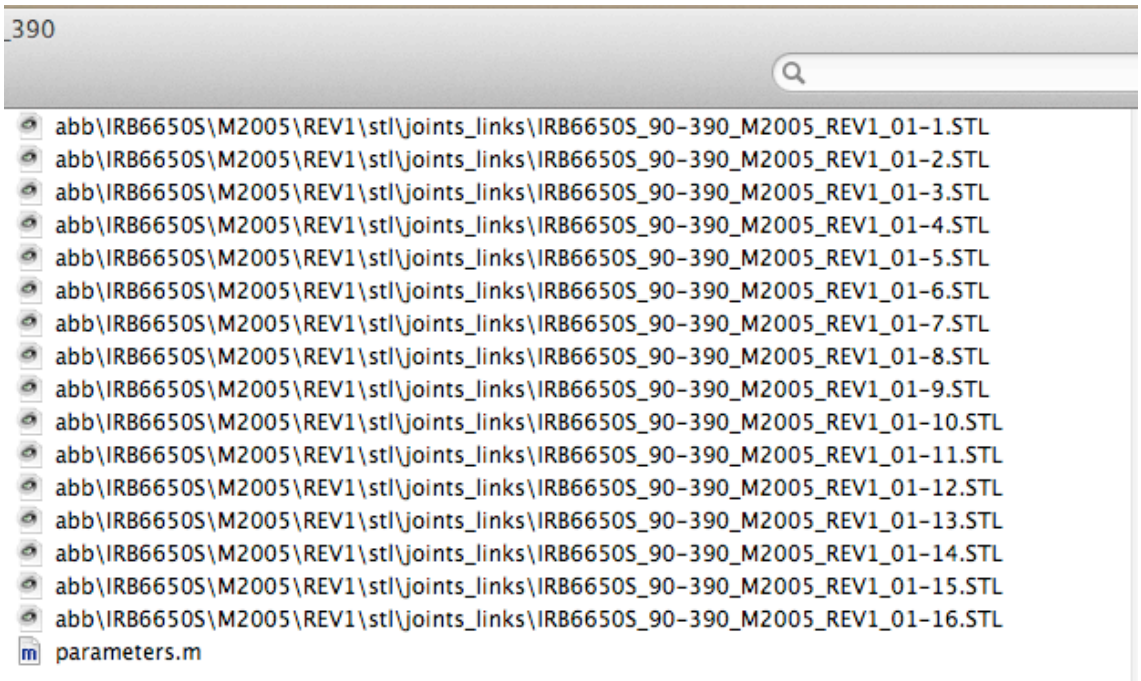
Validation	Ver.	DXF DWG 2D	SAT	SW	STEP	PARA-SOLID	VDA	IGES	STL	VRML	ROB CAD	Robot Studio	IGRIP
	200-300		compl joint	compl joint	compl joint	compl joint	joint	compl joint	compl joint	compl joint	joint		sim
	125-350		compl joint	compl joint	compl joint	compl joint	joint	compl joint	compl joint	compl joint	joint		sim
	90-390	compl	compl joint	compl joint	compl joint	compl joint	joint	compl joint	compl joint	compl joint		sim	sim
	-	compl											

2 Copy the files to the directory of your robot

You may place all the CAD files in the following directory:

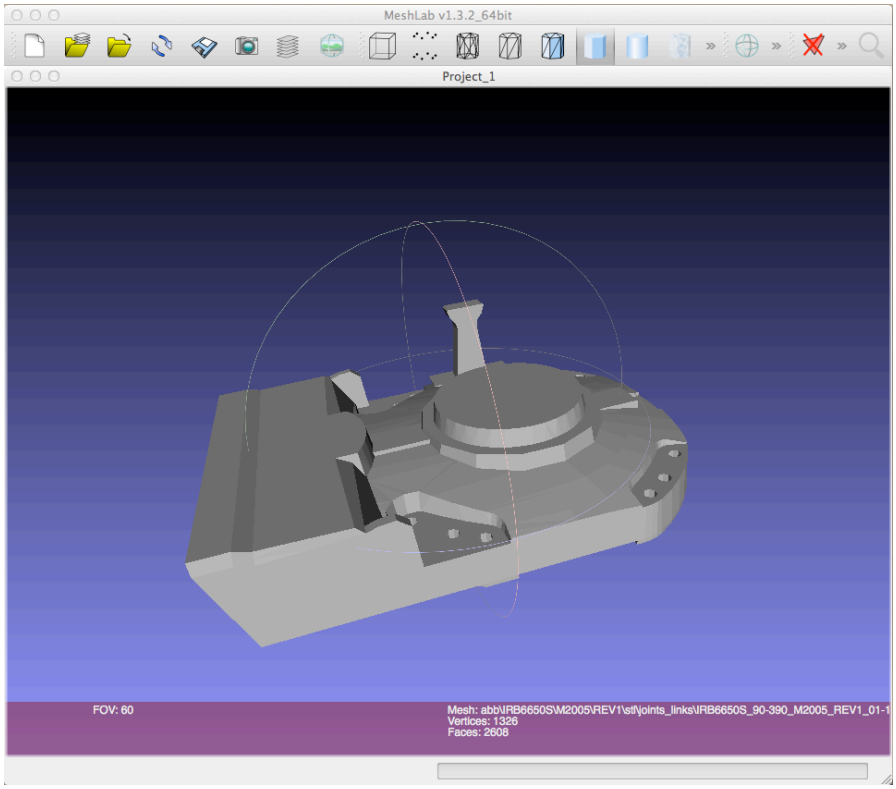
Arte_libX.X/robots/abb/IRB6650S_90_390, if we have a look at it:



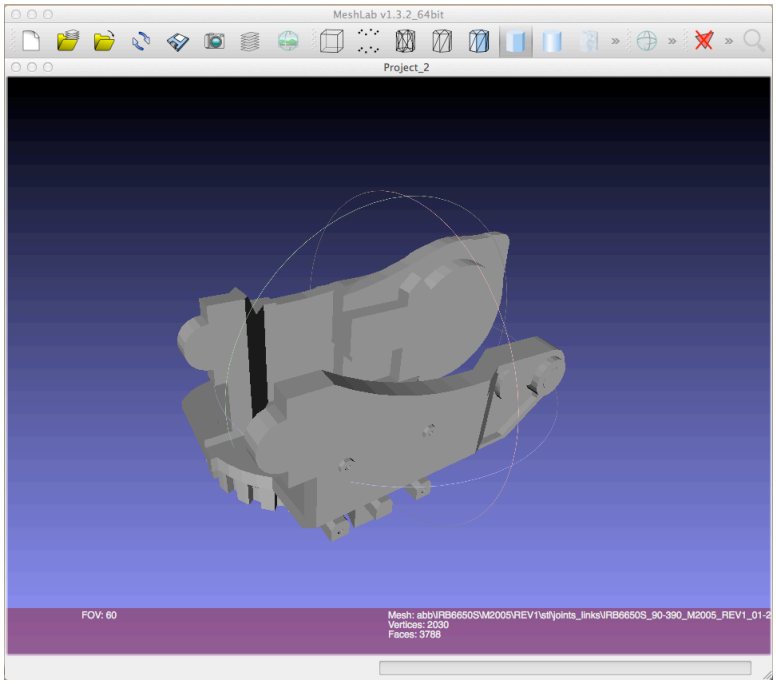


You should also copy a `parameters.m` file belonging to other robot and place it in the same directory.

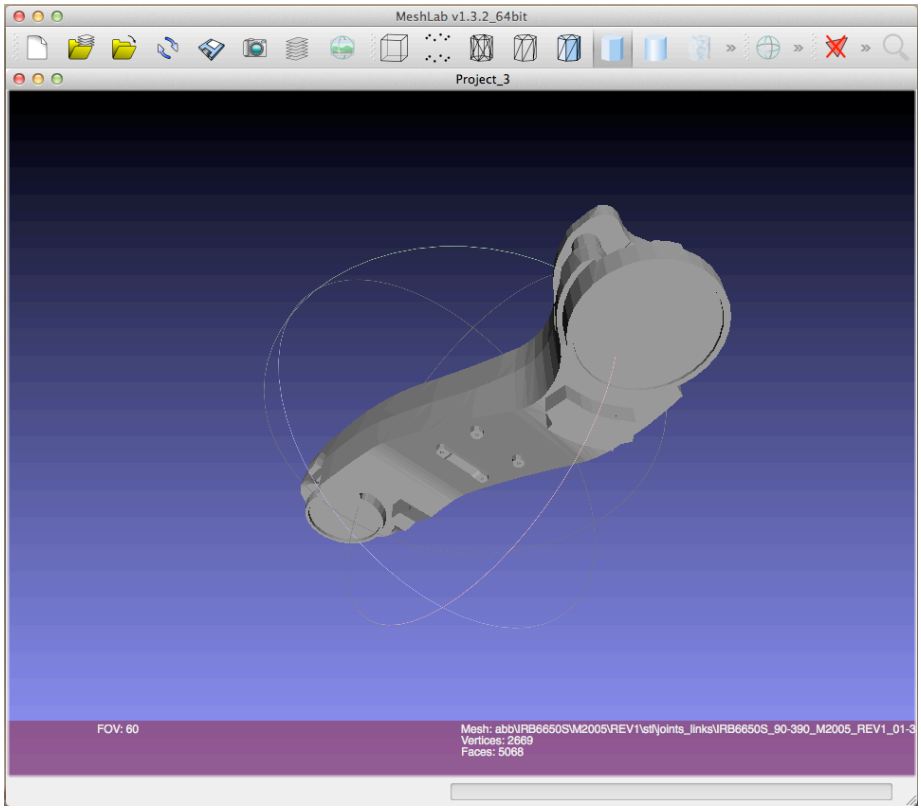
Most CAD programs will read the STL files. For example, download and install Meshlab: `meshlab.sourceforge.net`, for Windows, Mac or Linux. You should now have a look at the files. In Meshlab, open each of the files to view them. If you open the first file, you should find something like:



Or, the second file:



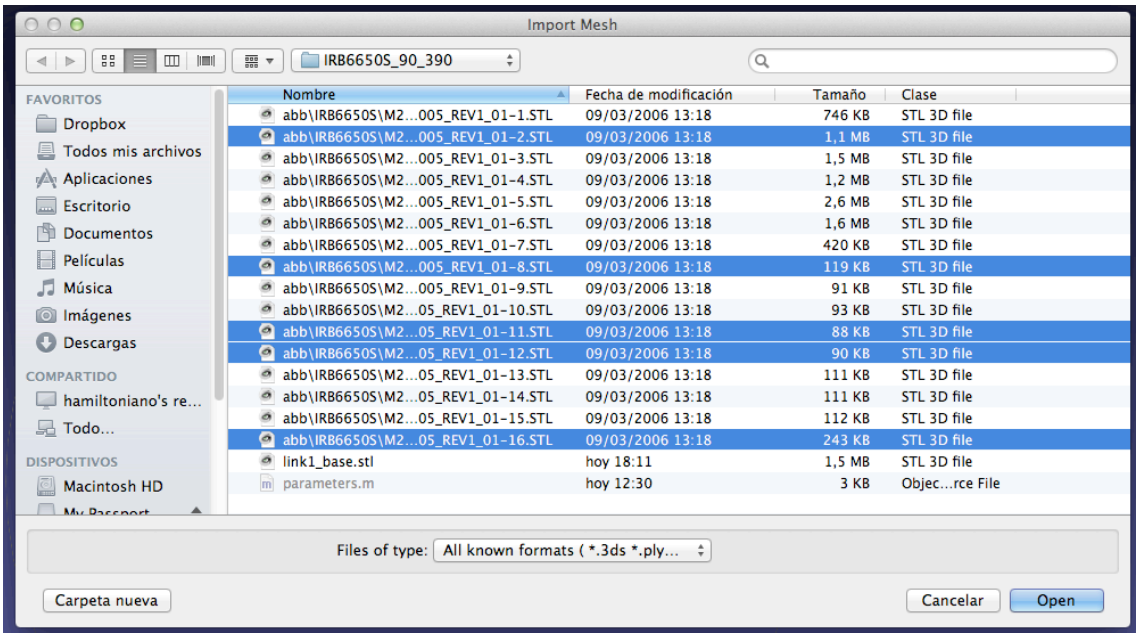
Third file:



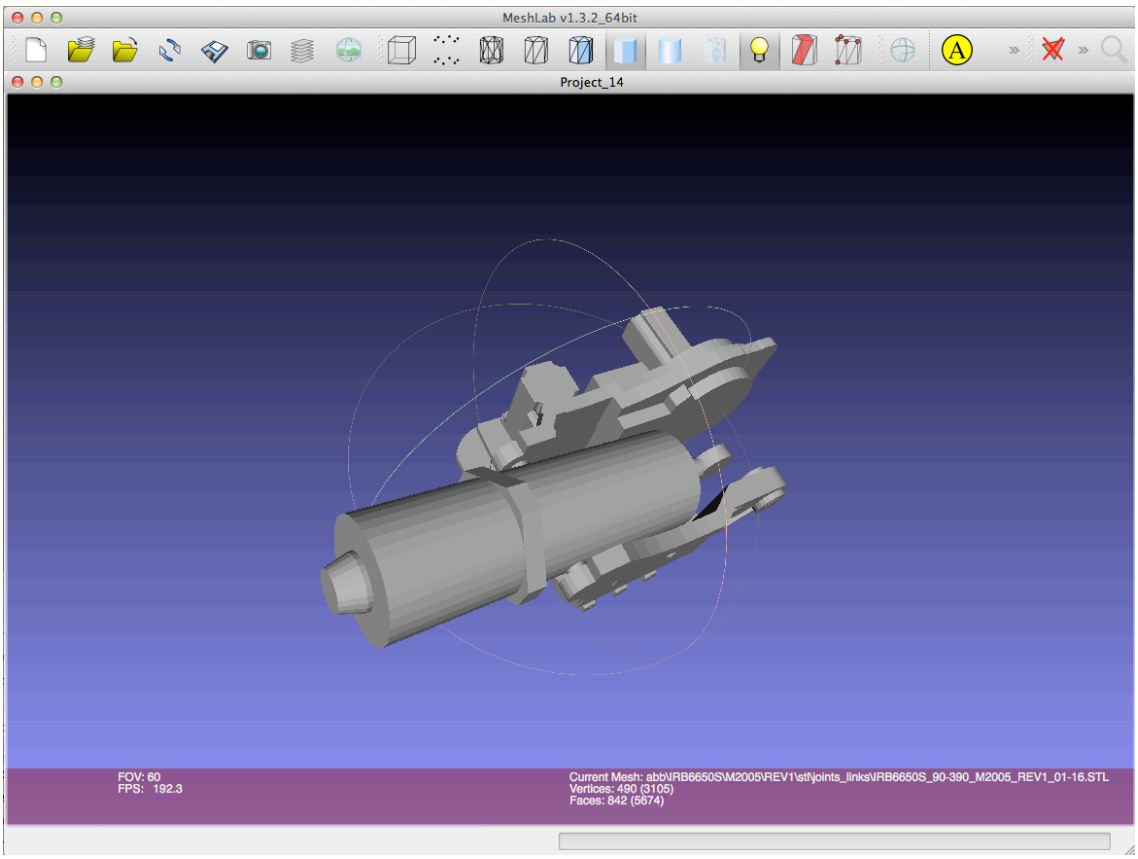
The reader should observe that some of the files constitute the basic shape of some of the robot links, whereas other files are details. You can merge some files into a single file to obtain a more detailed view of any of the links. To do so, you should have a look at the complete robot:

For example, start by opening the files with indexes 2, 8, 11, 12, 16. To do so, click on **File→ Import Mesh** 

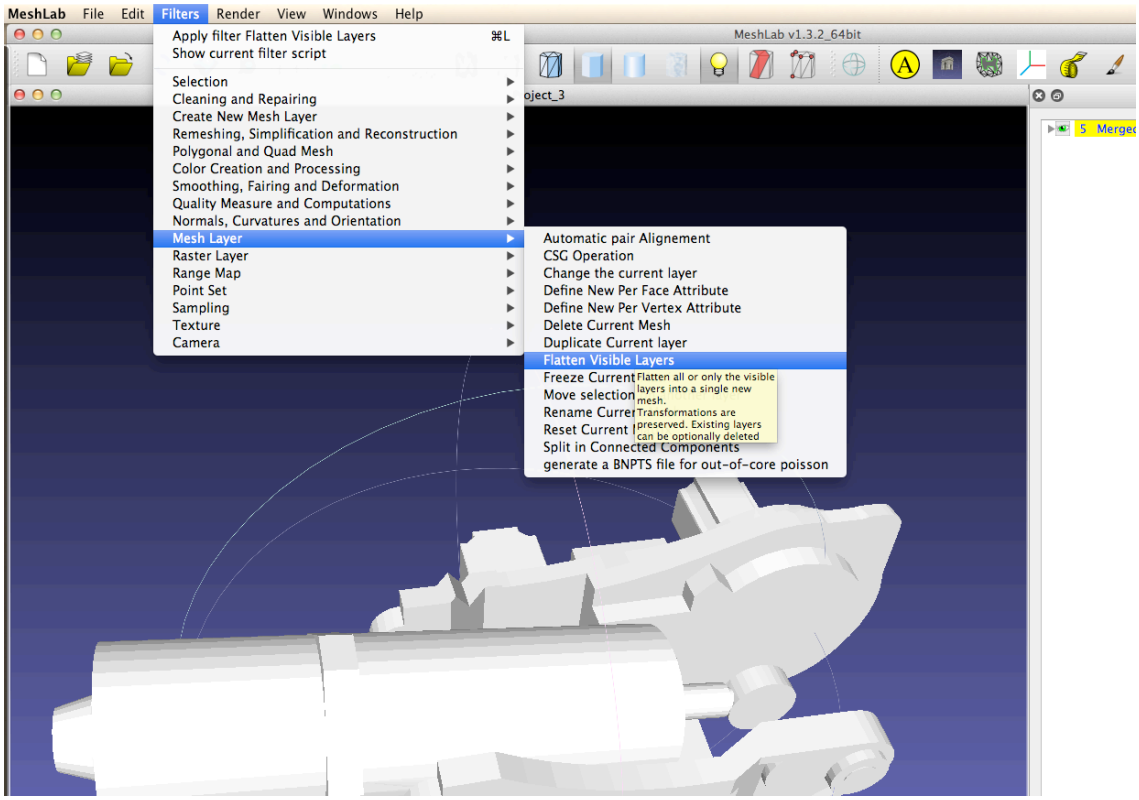
Select several files by clicking on Ctrl or cmd button.



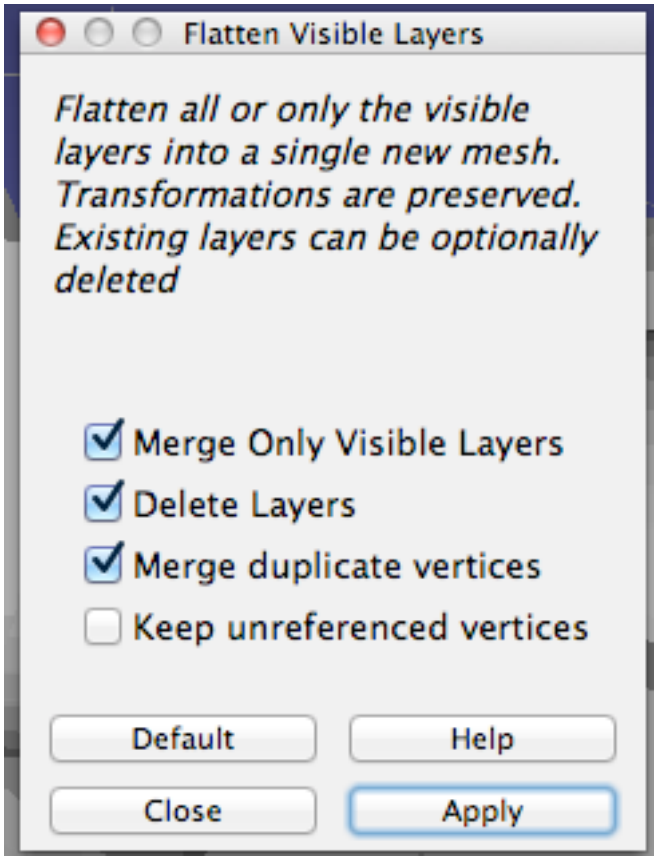
For example, after opening the files with index 2, 8, 11, 12, 16, you should have the following view of link 1.



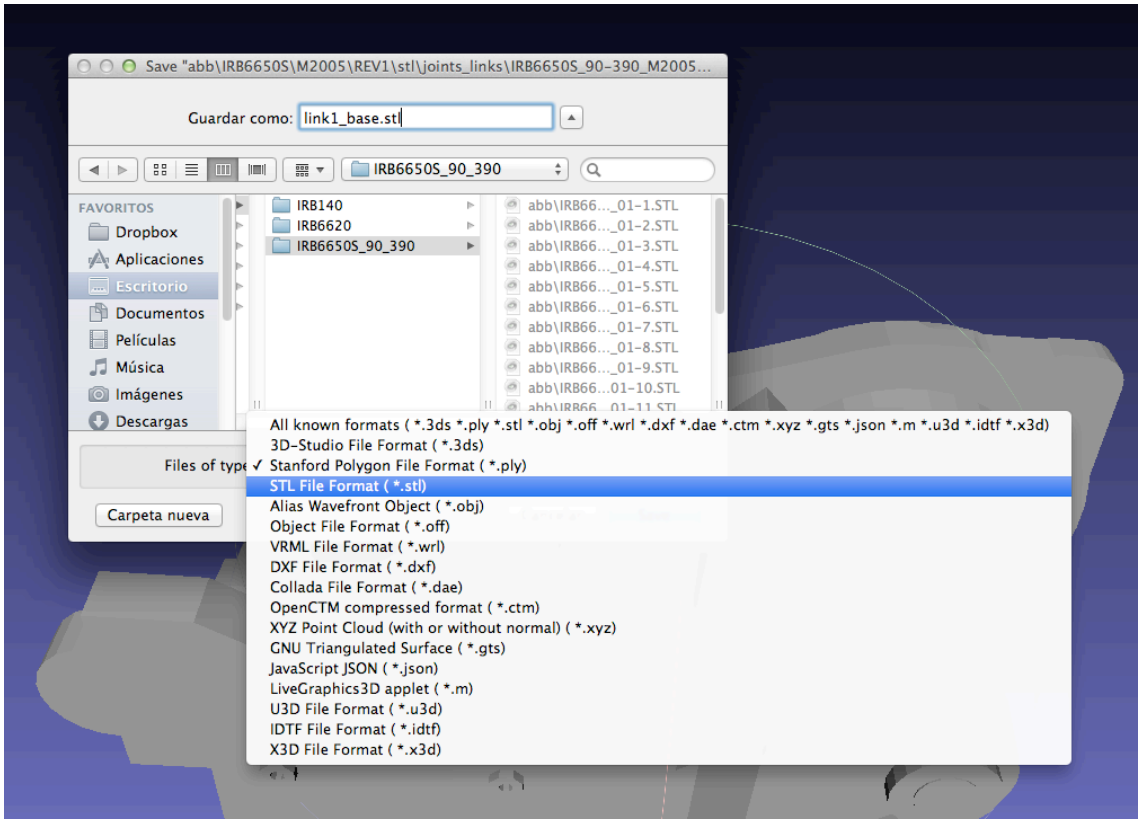
Next, open files 2, 8, 11, 12 and 16. Additionally, you should go to **Filters→Layers→ Flatten visible layers**



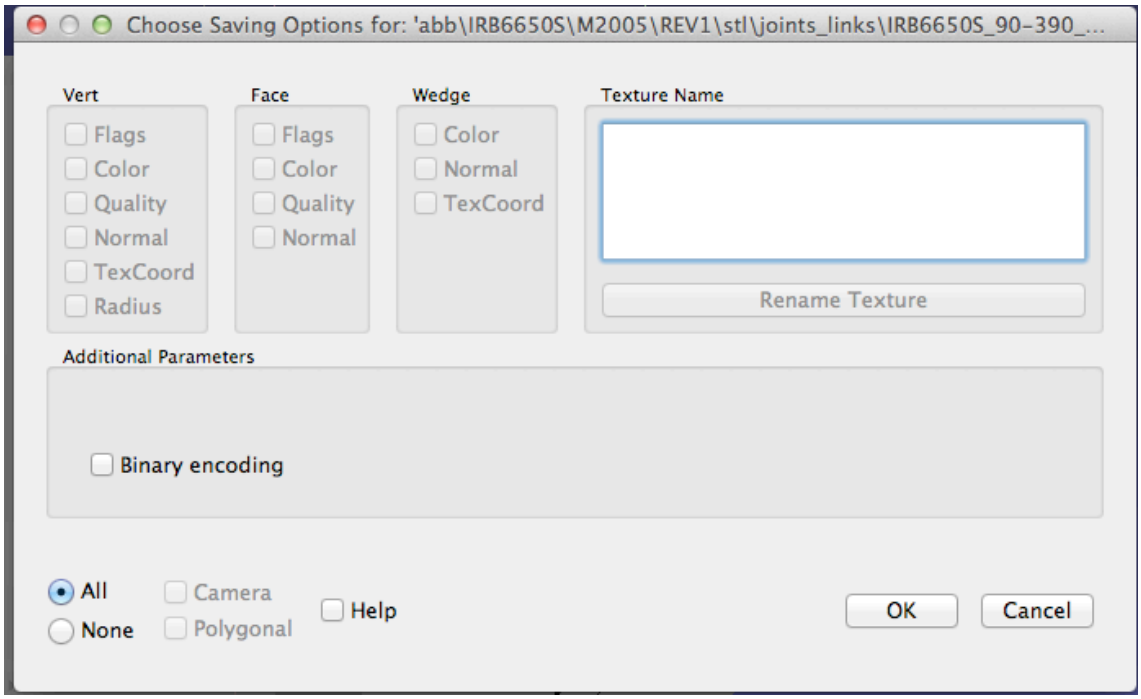
Next, click on Apply to combine the meshes.



Finally, save the file by selecting File→Export Mesh As. Select STL as format. And name the file as link1_base.stl. The suffix base means that the points in the STL file are referred to the base reference system.



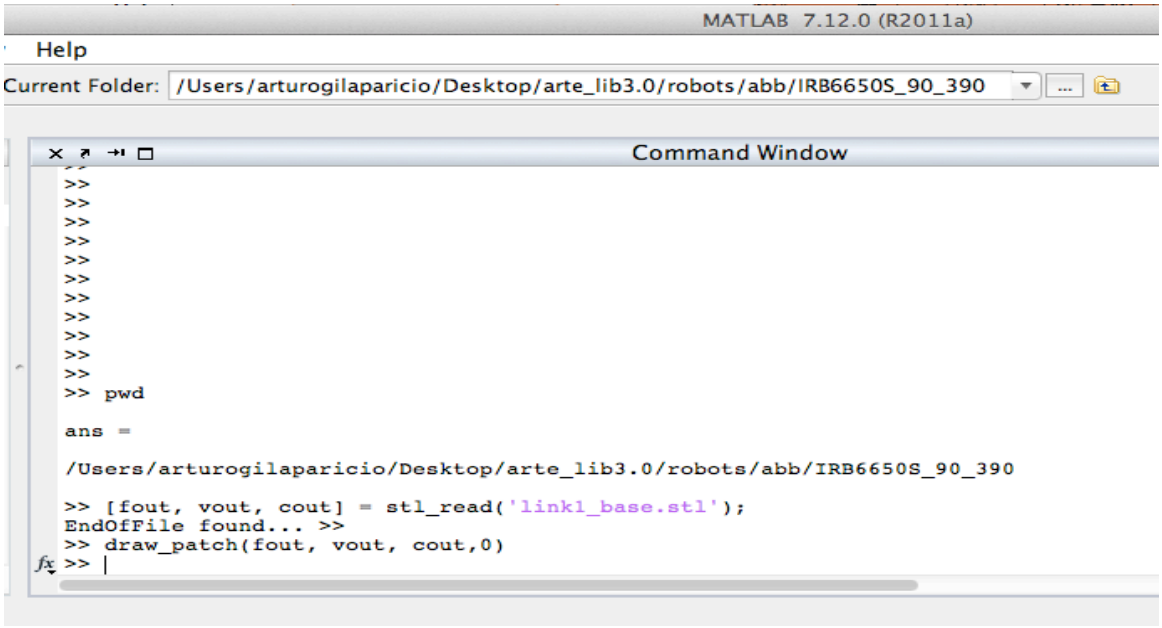
Next, de-select the Binary encoding option, which stores the file in ASCII format, that can be later read by Matlab.



Repeat the process for the rest of the files. The table shows the file indices that should be merged (flattenned) together.

Robot ARTE file	**Robot file index**
link0_base.stl Base, system 0	1
link1_base.stl Link 1, system 1.	2, 8, 11, 12, 16
link2_base.stl Link 2, system 2.	3, 9
link3_base.stl Link 3, system 3.	4, 10, 13
link4_base.stl link 4, system 4.	5, 14, 15
link5_base.stl link 5, system 5.	6
link6_base.stl link 6, system 6.	7

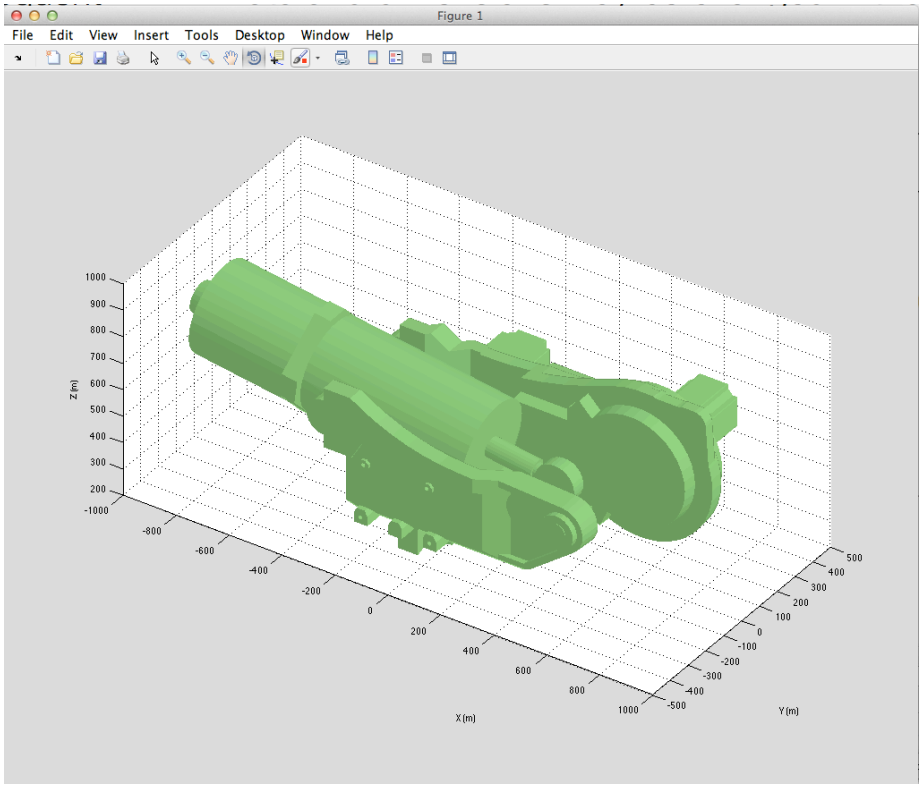
You can now try to visualize the files in Matlab. To do so, place the current folder in arte_libX.X/robots/IRB6650S_90_390 and type the following commands:




```
>>[fout, vout, cout] = stl_read('link0_base.stl');

>>draw_patch(fout, vout, cout,0)
```

After calling the `draw_patch` function, the following figure appears:

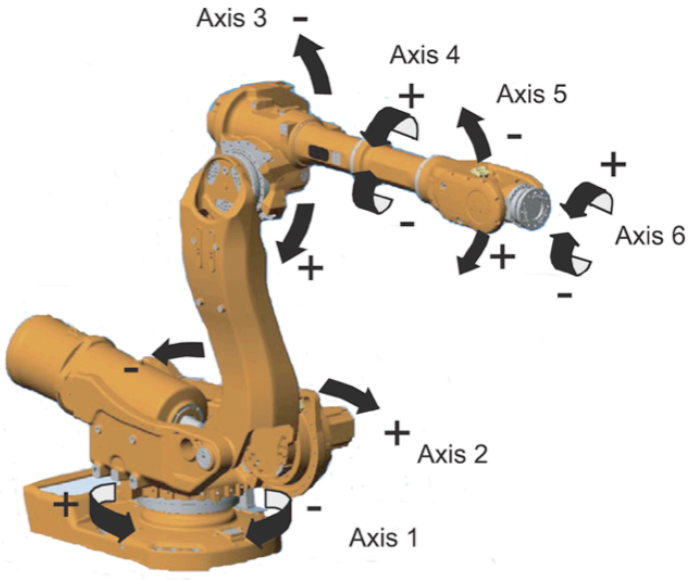


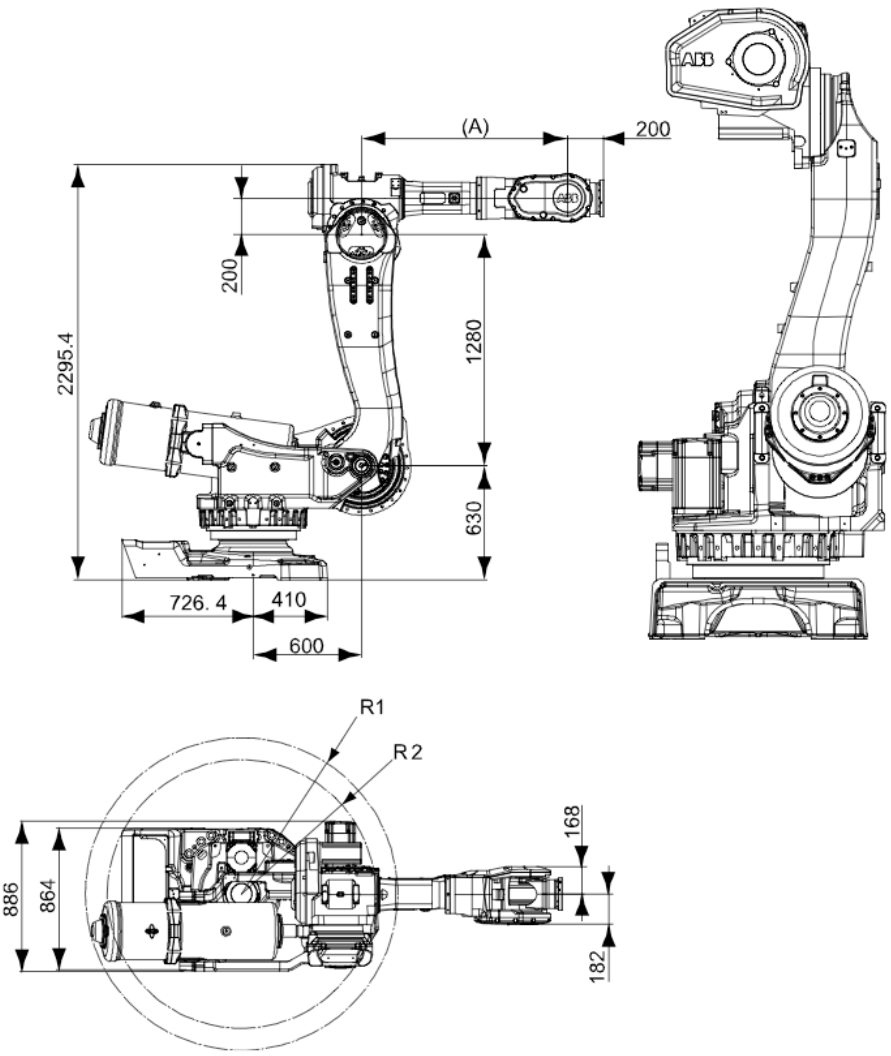
Please, note the scale (mm) and that the origin of the figure is not placed inside the link (points are referred to the base reference system).

3 Transform the files to a different reference system

Now, you should edit the parameters.m file to include the basic parameters of the arm, namely the D-H parameters, axes ranges, etc. In order to do so, copy a `parameters.m` file from a different robot and edit it. You should read the datasheet of the robot to obtain the main data.

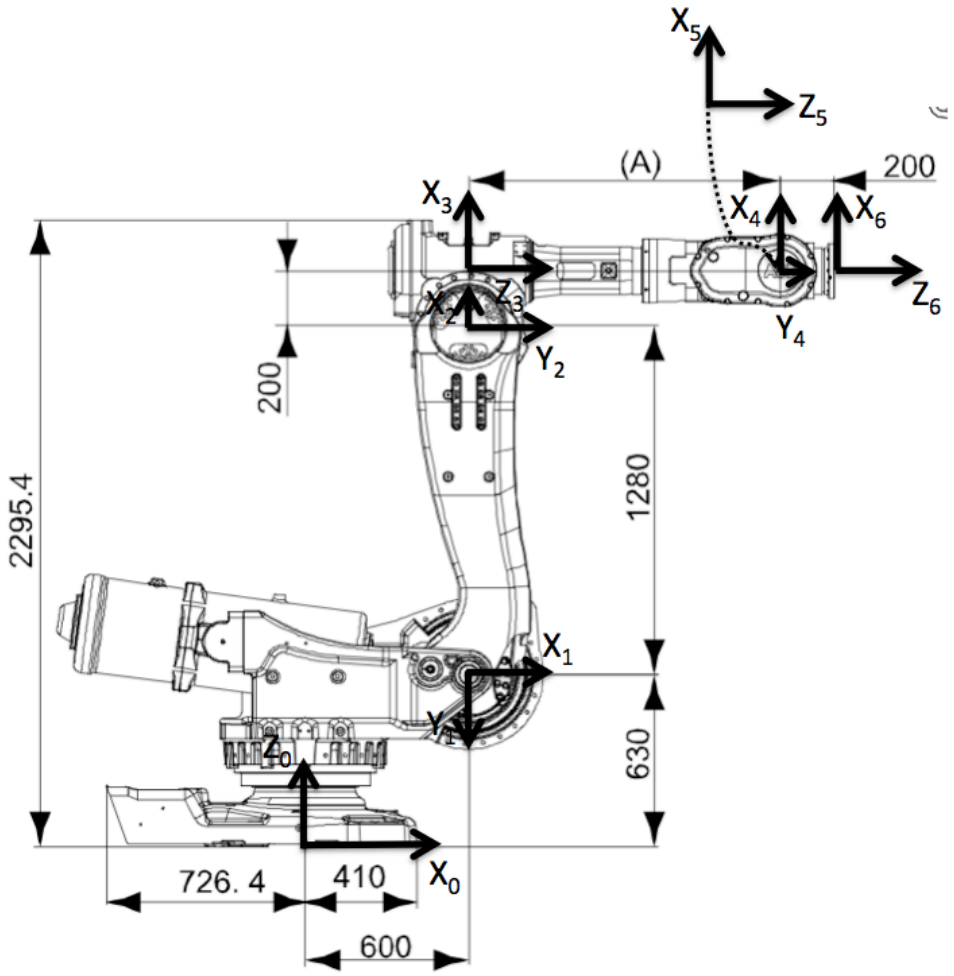
From the datasheet, we observe the positive turn direction for each joint and the main dimensions.





Robot variant	A
IRB 6650S-90/3.9	2042
IRB 6650S-125/3.5	1592
IRB 6650S-200/3.0	1142

In our case, A=2.042 meters. According to the version of the arm. Using the drawing above, place the D-H systems and write a D-H table. A possible placement for the standard D-H systems is shown next.



Which has the following D-H table associated:

θ	**d**	**a**	**α**
θ_1	0.630	0.600	-π/2
θ_2-π/2	0	1.280	0
θ_3	0	0.200	-π/2
θ_4	2.042	0	π/2
θ_5	0	0	-π/2
θ_6	0.200	0	0

Edit the parameters.m file to indicate the previous table. Note the variables, `robot.DH.theta`, `robot.DH.d`, etc.:

```
function robot = parameters()

robot.name= 'ABB_IRB6650S_90_390';

%Path where everything is stored for this robot
robot.path = 'robots/abb/IRB6650S_90_390';

robot.DH.theta= '[q(1) q(2)-pi/2 q(3) q(4) q(5) q(6)]';
robot.DH.d='[0.630 0 0 2.042 0 0.2]';
robot.DH.a='[0.600 1.280 0.2 0 0 0]';
robot.DH.alpha= '[-pi/2 0 -pi/2 pi/2 -pi/2 0]';

robot.J=[];


robot.inversekinematic_fn = 'inversekinematic_irb6650S_90_390(robot,
T)';

%number of degrees of freedom
robot.DOF = 6;

%rotational: 0, translational: 1
robot.kind=['R' 'R' 'R' 'R' 'R' 'R'];

%minimum and maximum rotation angle in rad
robot.maxangle =[deg2rad(-180) deg2rad(180); %Axis 1, minimum, maximum
                 deg2rad(-40) deg2rad(160); %Axis 2, minimum, maximum
                 deg2rad(-180) deg2rad(70); %Axis 3
                 deg2rad(-300) deg2rad(300); %Axis 4:
                 deg2rad(-120) deg2rad(120); %Axis 5
                 deg2rad(-360) deg2rad(360)]; %Axis 6:

%maximum absolute speed of each joint rad/s or m/s
robot.velmax = [deg2rad(100); %Axis 1, rad/s
                 deg2rad(90); %Axis 2, rad/s
                 deg2rad(90); %Axis 3, rad/s
                 deg2rad(150); %Axis 4, rad/s
                 deg2rad(120); %Axis 5, rad/s
                 deg2rad(235)];%Axis 6, rad/s
% end effectors maximum velocity
robot.linear_velmax = 0.0; %m/s, unavailable from datasheet

%base reference system
robot.T0 = eye(4);
```

Each STL file stores the 3D position of a set of points belonging to the link in coordinates of its own DH reference system. For example `link0.stl` has the position of the base points in coordinates of DH system 0. `Link1.stl` stores the position of the points belonging to link1 in coordinates of system 1, etc. However, the files, link1_base.stl, link2_base.stl… etc store the points in the base reference system. In addition, the units in this files are millimeters. The function `transform_to_own` will help you transform each stl file for each of

the reference systems, and at the same time, transform the scale to meters. Before executing `transform_to_own`, you should have created the `robot.DH` table before. The function reads the files link0_base.stl, link1_base.stl… and transforms each link to its own D-H reference system. Finally, the links are wrote to the same directory as link0.stl, link1.stl, etc. Read the transform_to_own help for more details.

```
>> help transform_to_own
```

Now, execute transform_to_own for the robot. You should obtain the following

```
>> transform_to_own('abb','IRB6650S_90_390', 1000)

ans =

/Users/arturogilaparicio/Desktop/arte_lib3.0/robots/abb/IRB6650S_90_39
0

Reading link0_base.stlEndOfFile found... Wrote 2608 facets
Reading link1_base.stlEndOfFile found... Wrote 5674 facets
Reading link2_base.stlEndOfFile found... Wrote 5388 facets
Reading link3_base.stlEndOfFile found... Wrote 4764 facets
Reading link4_base.stlEndOfFile found... Wrote 9759 facets
Reading link5_base.stlEndOfFile found... Wrote 5522 facets
Reading link6_base.stlEndOfFile found... Wrote 1476 facets
```

4 Test everything

Test everything! First, load the robot:

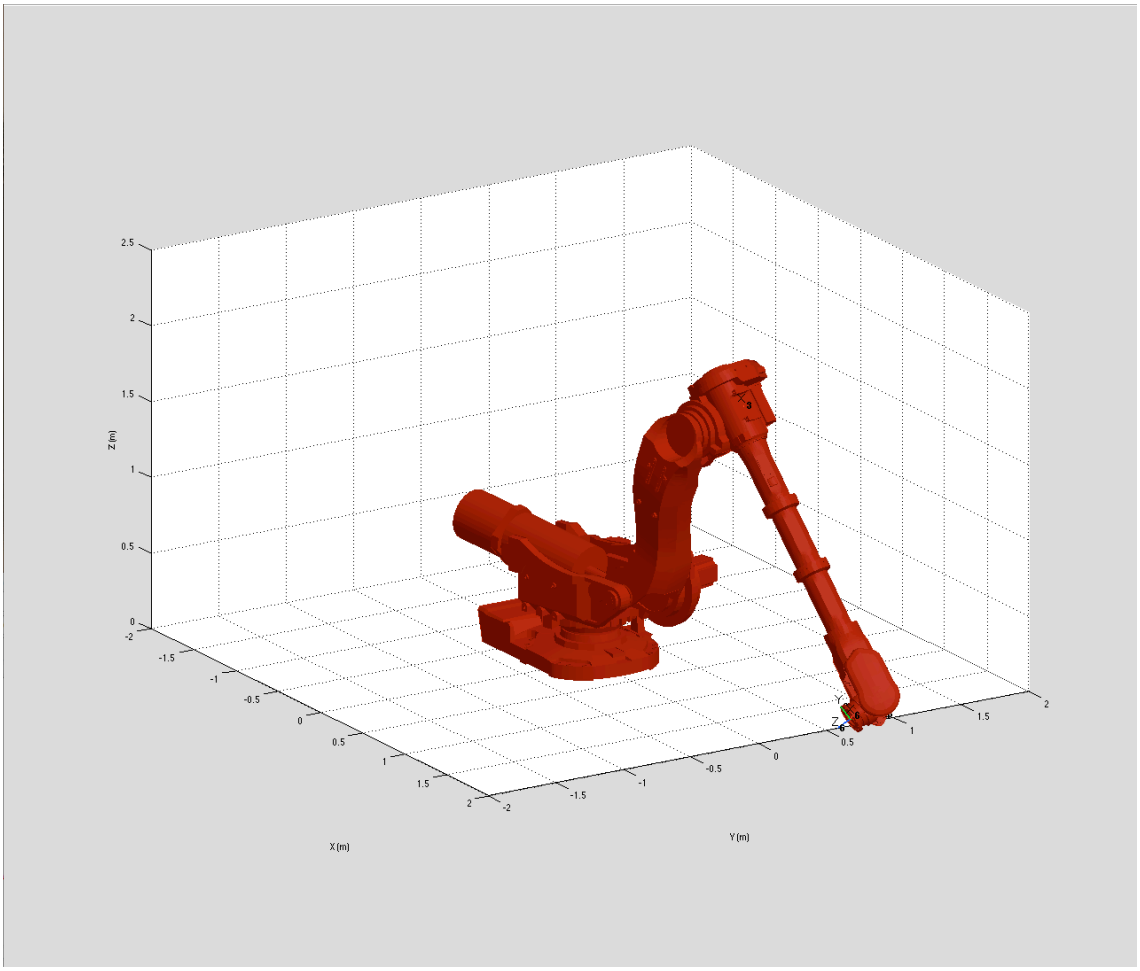
```
>> robot=load_robot('abb', 'IRB6650S_90_390');
ans =

/Users/arturogilaparicio/Desktop/arte_lib3.0/robots/abb/IRB6650S_90_39
0


Reading link 0
EndOfFile found...
Reading link 1
EndOfFile found...
Reading link 2
EndOfFile found...
Reading link 3
EndOfFile found...
Reading link 4
EndOfFile found...
Reading link 5
EndOfFile found...
Reading link 6
EndOfFile found...
```

Now, you can use the teach GUI to test your robot. Use the sliders to move the joints and test that everything is correct.

```
>> teach
```

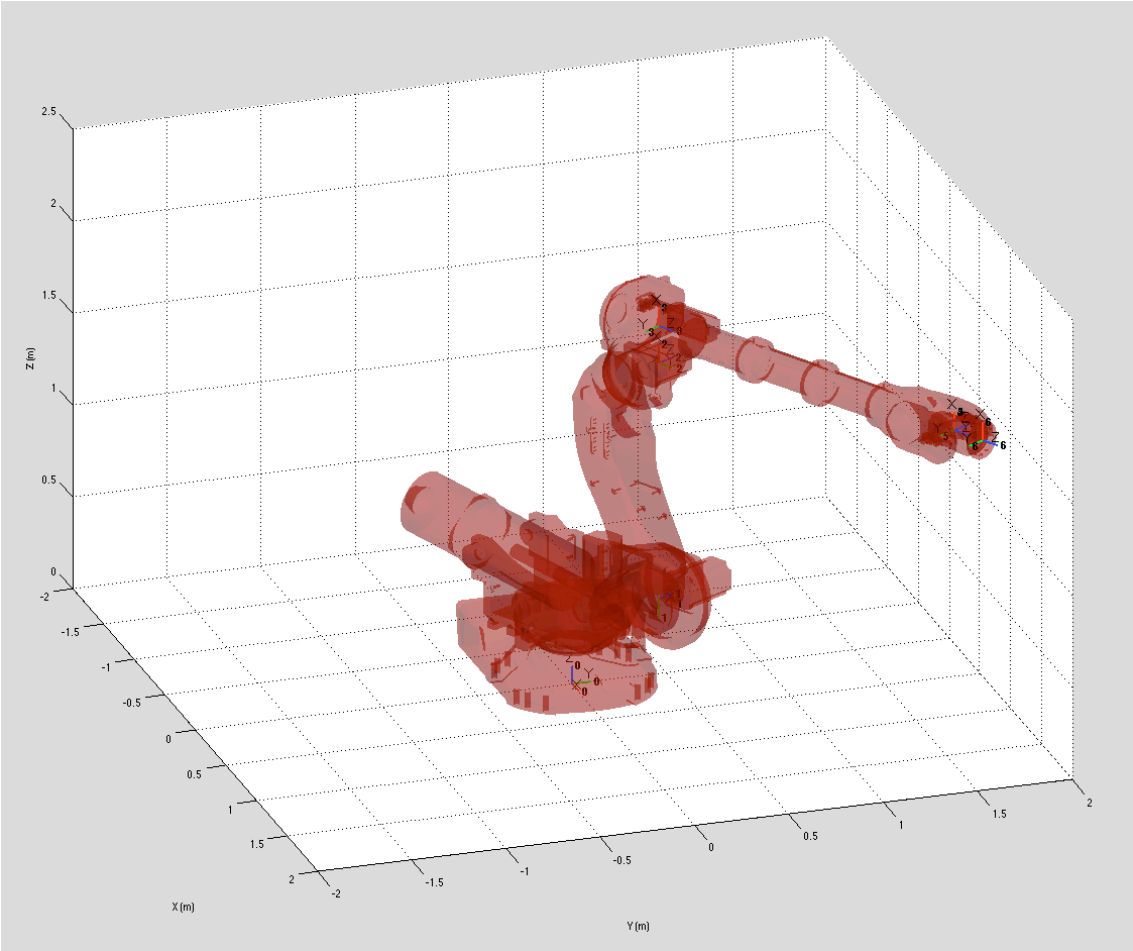


Large robots may require that you modify the axis in the matlab figure. Edit the `parameters.m` file and modify the robot.axis variable:

```
robot.axis=[-3 3 -3 3 0 3];
```

Set robot.graphical.draw_transparent=1 if you prefer to observe the D-H systems and a transparent representation of the robot.

```
>> robot.graphical.draw_transparent=1
```



5 What's next?

During the next practical sessions you will modify the `parameters.m` file to add more functionality to your robot, solving the inversekinematic problem, adding dynamics and programming the robot.