Tutorial: How to Transfer Pieces from Solid Works to Matlab

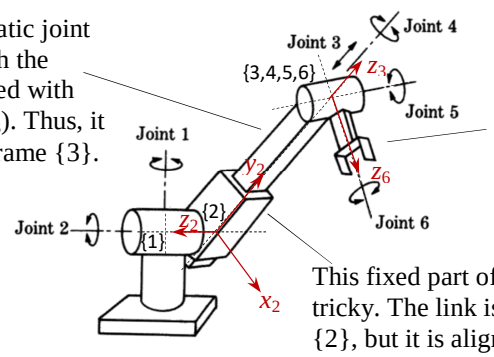
by Flavio Firmani

Identify Kinematic Properties of your Links (on paper)

- Attach reference frames to the manipulator
- Identify the Denavit and Hartenberg Parameters
- Define the centre of the physical link as the point on the link that coincides with the origin of a reference frame.
- Identify whether the link is a link offset or a link length
- Identify the frame associated to each physical link.

For example, for the Stanford manipulator shown below, the location of the frame origins $\{i\}$ are shown as the intersection of the joint axes.

The moving part of the prismatic joint end-effector is coincident with the origin of frame {3} and aligned with the axis of joint 3 (negative z_3). Thus, it is a link offset associated to frame {3}.



The end-effector is coincident with the origin of frame {6} and aligned with the axis of joint 6 (z_6). Thus, it is a link offset associated to frame {6}.

This fixed part of the prismatic joint is kind of tricky. The link is located at the origin of frame {2}, but it is aligned with joint 3. This special case that can be treated in two different ways:

- Link offset attached to frame {3}, but eventually be displaced by a distance $-d_3$.
- Link attached to frame {2} oriented along y_2.

Preparing a piece in SolidWorks

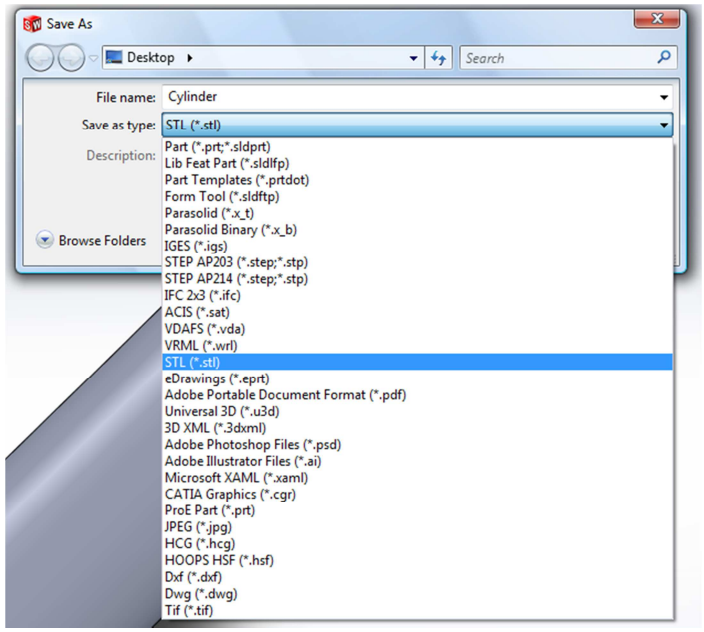
Each piece will be transferred to Matlab separately, *e.g.*, base piece, links (including end-effector), tools, or an external piece that the manipulator would have to pick.For the pieces that will be moving, such as links do the following:

- From your original SolidWorks file, you can either save multiple files with only one link or hide all of them but the one to be transferred.
- Transfer the link to the global reference frame.
- Rotate the link based of its kinematic representation:
  - If the link is a **link length** $\boldsymbol{a_{i-1}}$ align it along ***x* axis**.
  - If the link is a **link offset** $\boldsymbol{d_i}$ align it along ***z* axis**.
  - Make sure it is drawn along the correct direction (positive or negative)
  - Special links, like the fixed part of the prismatic joint, may be aligned along the ***y* axis**.

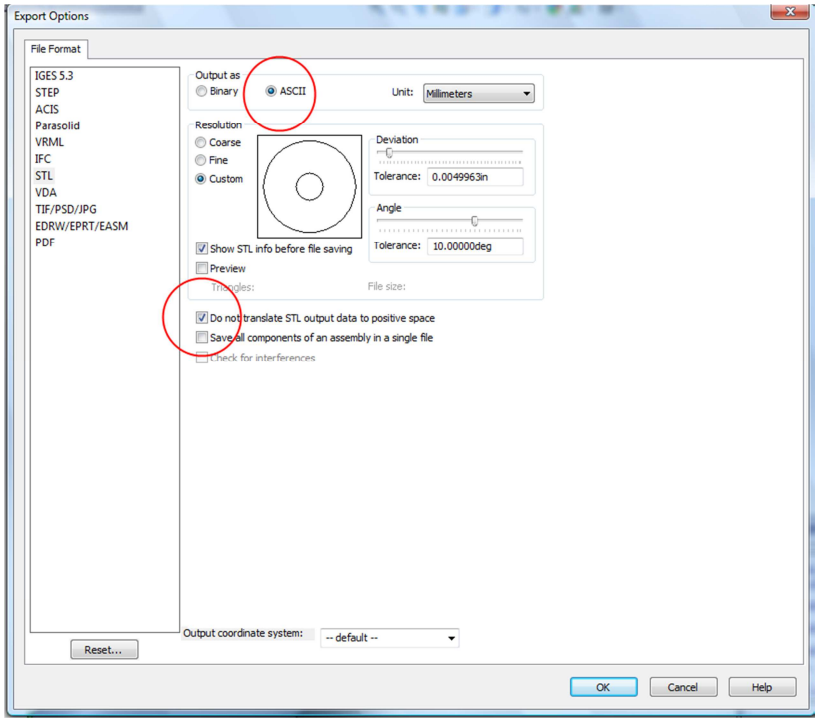
Converting piece as a STL file

STL files (STereoLithography) decomposes the surfaces into small triangles that connect all the vertices.

- In Solid Woks, save piece with **Save as** by giving a name (e.g. Cylinder) and select the following specifications.
- In **Save as type** select **STL (*.stl)**



- In **Options** check **ASCII** in **Output as** and also check the box that says **Do not translate STL output data to positive space**.



Transferring piece to Matlab

- Save the STL file in the same directory as all the Matlab files found in the course website under Matlab resources.
- Type in the Matlab command window:

  >> **stl2mat('Cylinder')** %Use the name you gave to the piece in Solid Works

- Run the following code to check the output piece in Matlab

```
close all
load Cylinder %Name of your Solid Works piece
setappdata(0,'object_data',object);

p = patch('faces',object.F, 'vertices',object.V); %Create its surfaces
set(p,'FaceColor', [0.5430    0.2695    0.0742]); %Color of surfaces
set(p,'EdgeColor','none'); %Do not show Edge line of triangular surfaces
drawnow

set(fig_handle,'Renderer','zbuffer','doublebuffer','on')% Renders your figures
light('color',[.99,.99,.99],'position',[5,0,2],'Style','infinite') %Light Position
lighting flat % Type lightintining
daspect([1 1 1]); %Axis Ratio
axis off; % Do not show axis
hold all;
plot3(0,0,0,'b*') %Plot origin of global reference frame
```

- If the output is not correct (the piece has some awkward triangular surfaces which you can see by rotating the piece with the 3D Rotate option of the toolbar ), open the **stl2mat.m** file, comment/erase the code shown below (lines 22-34), and run **stl2mat('Cylinder')** as before

```
% Reduce the number of Vertices
original_vertices=length(object.V(:,1)) %#ok<NASGU,NOPRT>
[b, m] = unique(object.V,'first','rows'); %#ok<ASGLU>
m1=sort(m); b1=object.V(m1,:); m2=1:1:length(m);
for i=1:length(m2)
    pv = findsubmat(object.V,b1(i,:));
        for j=1:length(pv)
            [ind_i,ind_j]=find(pv(j)==object.F);
            object.F(ind_i,ind_j)=m2(i);
        end;
end;
object.V=b1;
reduced_vertices=length(object.V) %#ok<NOPRT,NASGU>
```

- Once you are happy with the piece, load it in your main code main_project.m.