Tutorial 35: Creation of Kinematic Chains and Robot Structures

2017-07-04: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-25

Contents

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function

Motivation for this tutorial: (Originally SolidGeometry 4.0 required)

Already in the tutorials 11 and 12 kinematic chains were presented. This tutorial is about creating tree-like structures for robotic systems. The example uses the structures of the robot JACO. function VLFL_EXP35

1. Loading STL Files or Surface Data

The Elements of the JACO were prepared by reading STL data in and save the variables using the save command. Now the surface data is available but also those surfaces have already defined frames "B" for base and "F" for follower. clear all

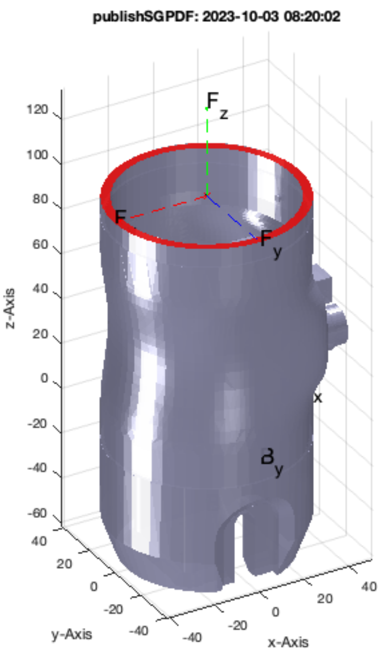
```
loadweb JACO_robot.mat
whos
```

```
loadweb: Access path to changed from "www.mimed.mw.tum.de" to "www.mw.tum.de/mimed/" in 2020 Aug.
loadweb: Access path to changed from "www.mw.tum.de/mimed/" to "www.mec.ed.tum.de/mimed/" in 2021 Nov.
Downloading "https://www.mec.ed.tum.de/fileadmin/w00cbp/mimed/Matlab_Toolboxes/JACO_robot.mat" into: /Volumes/LUETH-WIN/WIN AIM Matlab Libraries/SolidG
ans =
    '/Volumes/LUETH-WIN/WIN AIM Matlab Libraries/SolidGeometry-Code/downloaded_JACO_robot.mat'
  Name           Size                    Bytes  Class                             Attributes

  A              1x1                     91580  struct
  A0             1x2                        16  double
  A1             1x2                        16  double
  B              1x1                     91580  struct
  B0             1x2                        16  double
  B1             1x2                        16  double
  BPL          274x3                      6576  double
  C              1x1                     98204  struct
  C1             1x2                        16  double
  C2             1x2                        16  double
  C3             1x2                        16  double
  CPL          607x2                      9712  double
  CPLB         273x2                      4368  double
  CPLN         535x2                      8560  double
  D              1x1                     98204  struct
  D1             1x2                        16  double
  D2             1x2                        16  double
  D3             1x2                        16  double
  EL           531x2                      8496  double
  FL           727x3                     17448  double
  FLB           41x3                       984  double
  FLT           69x3                      1656  double
  FLW          934x3                     22416  double
  FN             1x69                      138  char
  FN2            1x60                      120  char
  FZG            1x6                   2639928  cell
  GPL          351x2                      5616  double
  I            400x400                 1280000  double
  I1          1188x1411x3              5028804  uint8
  ID             1x1                    977928  struct
  IE             1x1                   5029140  struct
  IM             1x1                   5029140  struct
  IT             1x1                   5029140  struct
  JACO           1x8                   6370464  cell
  JC0            1x1                   1100542  struct
  JC00           1x1                   1465854  struct
  JC01           1x1                    369566  struct
  JC1            1x1                    843774  struct
  JC2            1x1                    757014  struct
  JC3            1x1                    695054  struct
  JC4            1x1                    477742  struct
  JC5            1x1                    477742  struct
  JC6            1x1                   3731654  struct
  JC61           1x1                   1431846  struct
  JCF            1x1                    220606  struct
  L              1x4                        32  double
  L1             1x1                         8  double
  L2             1x1                         8  double
  L3             1x1                         8  double
  L4             1x1                         8  double
  LMax           1x1                         8  double
  LMin           1x1                         8  double
  NPL          632x3                     15168  double
  PL            16x2                       256  double
  PLA          504x2                      8064  double
  PLB          554x2                      8864  double
  PLU0           5x2                        80  double
  PLU1           9x2                       144  double
  PLU2          11x2                       176  double
  Ri             1x1                         8  double
  Ro             1x1                         8  double
  SG             4x1                    630704  cell
  SG1            1x1                   3830904  struct
  SG2            1x1                  26171232  struct
  SG3            1x1                   2174640  struct
  SG4            1x1                  10665648  struct
  SG5            1x1                    474528  struct
  SG6            1x1                   1170384  struct
  SGN            4x1                    712784  cell
  T              4x4x81                  10368  double
  T1             4x4                       128  double
  T2             4x4                       128  double
  V            512x512x126            66060288  uint16
  VL            10x3                       240  double
  VLB           30x3                       720  double
  VLR            8x3                       192  double
```

```
VLr         13x3                  312  double
VM          128x128x128      16777216  double
a           216x216x126      47029248  double
ans         1x88                  176  char
as          1x3                    24  double
conn        1x1                     8  database.jdbc.connection
d           1x2                    16  double
i           1x1                     8  double
l           1x1                     8  double
l1          1x1                     8  double
l2          1x1                     8  double
l3          1x1                     8  double
ms          1x3                    24  double
p           3x1                    24  double
phi         1x1                     8  double
simOut      1x1                  7370  Simulink.SimulationOutput
slot        1x1                     8  double
smbsys      1x13                   26  char
ta          81x1                  648  double
tb          81x1                  648  double
v           3x1                    24  double
vname       1x69                  138  char
vs          1x3                    24  double
xout        1x1                    25  Simulink.SimulationData.Dataset
```

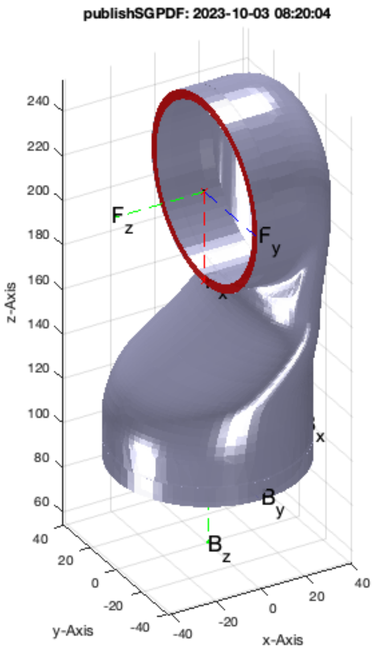
Plot the controller module/base of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC0);
```



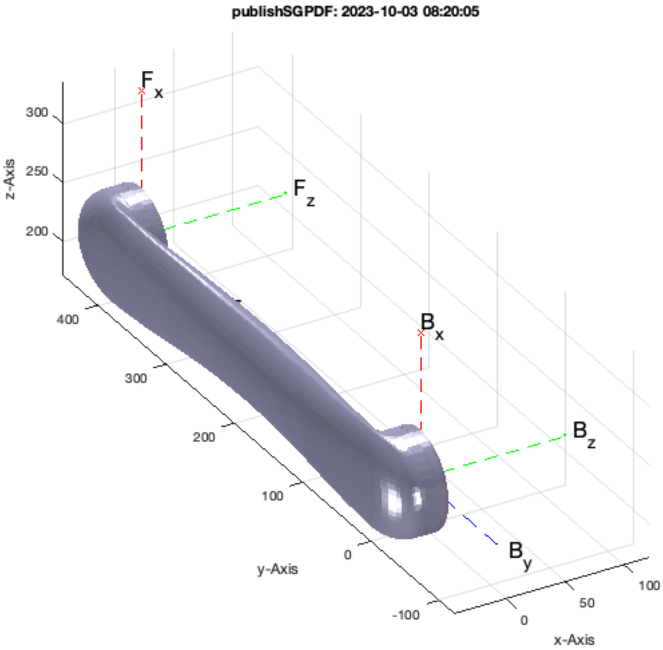
Plot the arm segment 1 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC1);
```



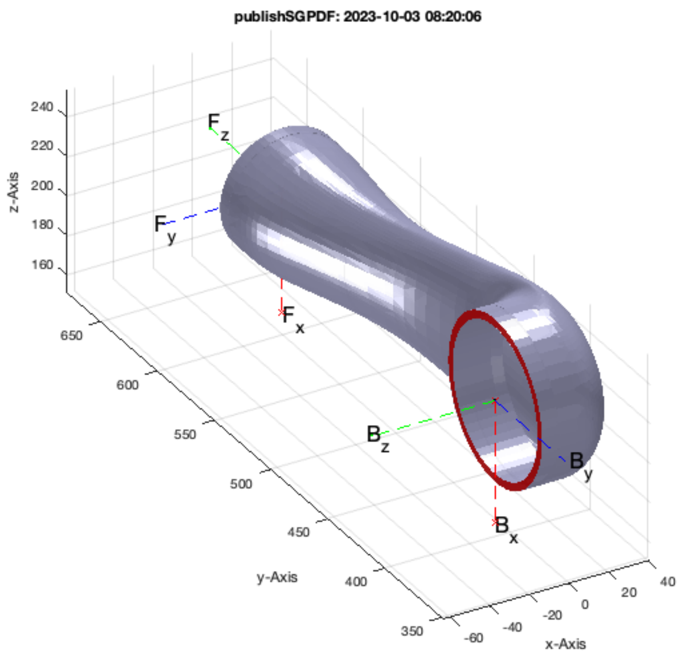
Plot the arm segment 2 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC2);
```



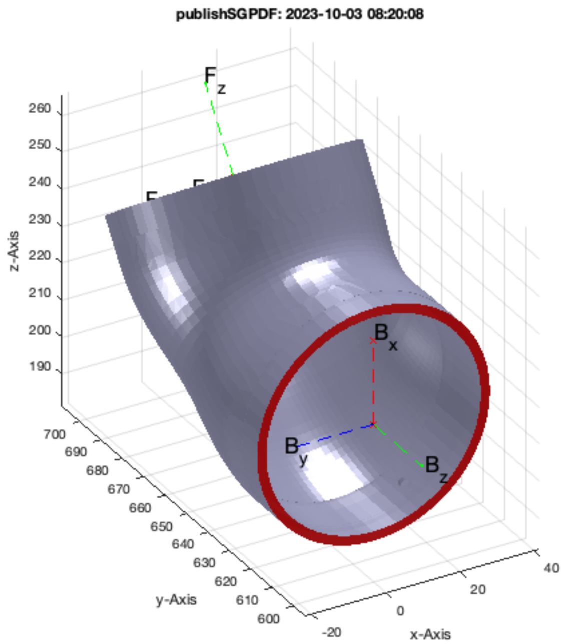
Plot the arm segment 3 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC3);
```



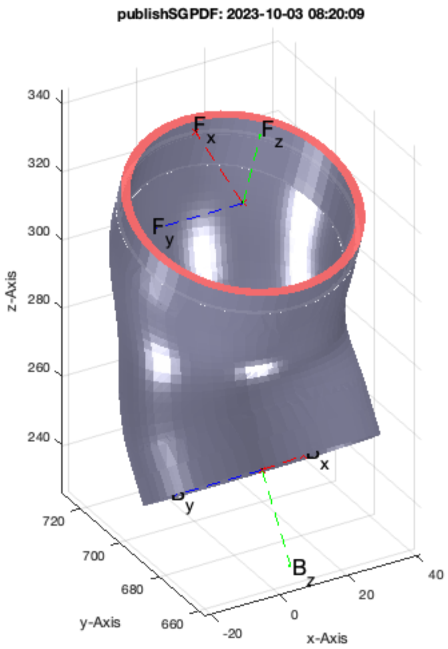
Plot the arm segment 4 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC4);
```



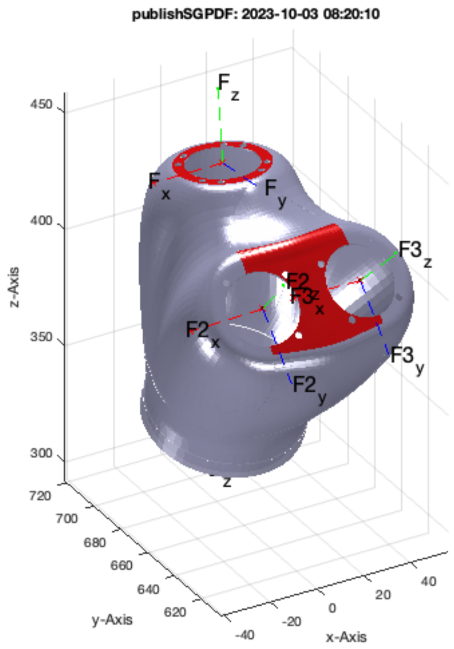
Plot the arm segment 5 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC5);
```



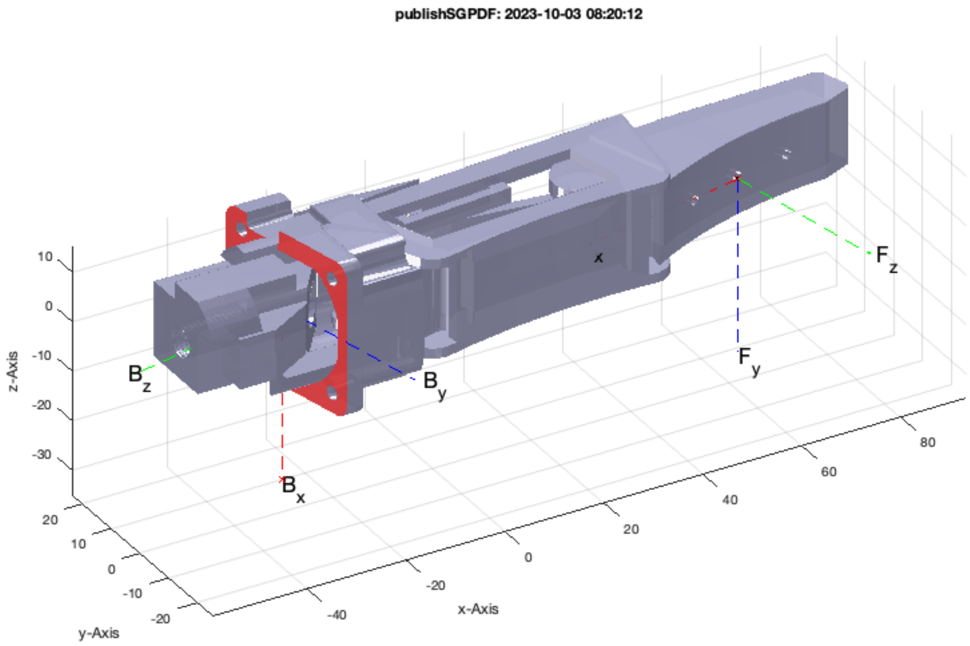
Plot the arm segment 6/the hand of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC61);
```



Plot one finger segment 3 of the Jaco robot's hand

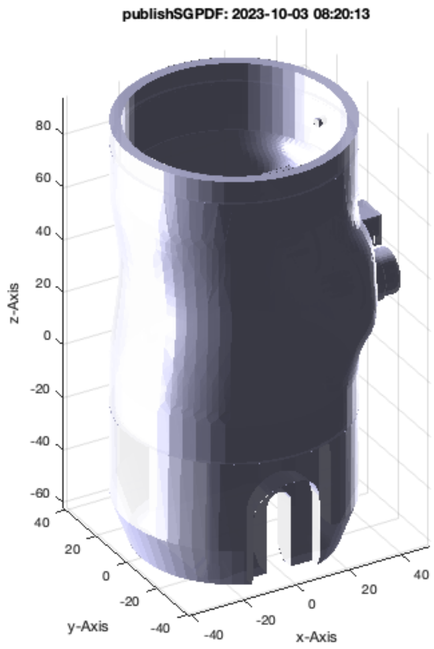
```
SGfigure; view(-30,30); SGTplot(JCF);
```



2. Attaching Frames to a Surface Model

To learn how to attach frames, we make a copy of only the surface of jaco's base.

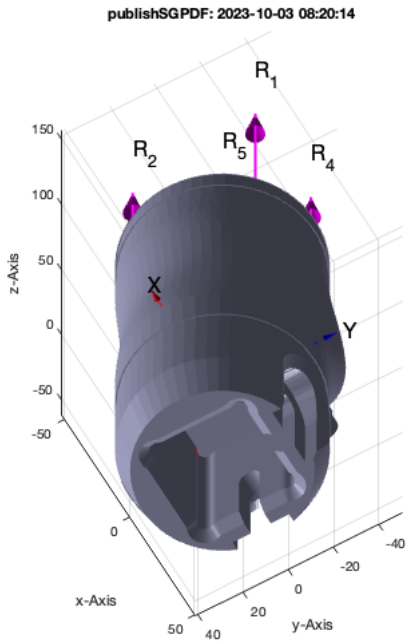
```
clear SG;
SG.VL=JC0.VL; SG.FL=JC0.FL; SG.col='w'; SG.alpha=0.9;

SGfigure; view(-30,30); SGplot(SG);
```



Now use SGTui to specify a planar or freeform surface by klicking on the surface. Turn the object before the klick into the desired orientation Now try to create a base frame by clicking on the lower surface. If you touch a freeform surface it may take while until the surfaces are autoamtically selected

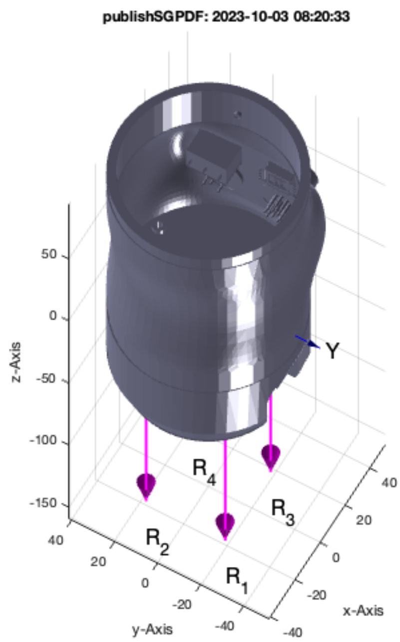
```
SGfigure; SG=SGTui(SG,'B'), view(-60,-60);
```

```
SG =
  struct with fields:

       VL: [15230×3 double]
       FL: [30472×3 double]
      col: 'w'
    alpha: 0.9000
    Tname: {'B'}
        T: {[4×4 double]}
     TFiL: {[206×1 double]}
     TFoL: {[]}
```




```
SGfigure; SG=SGTui(SG,'F'), view(-60,+60);
```

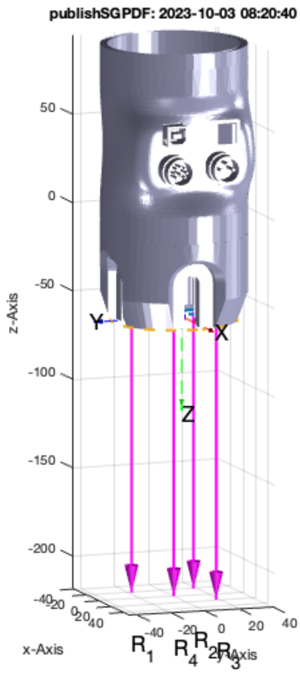
```
SG =
  struct with fields:

       VL: [15230×3 double]
       FL: [30472×3 double]
      col: 'w'
    alpha: 0.9000
    Tname: {'B'  'F'}
        T: {[4×4 double]  [4×4 double]}
     TFiL: {[206×1 double]  [86×1 double]}
     TFoL: {[]}
```



You may have notices that not only a surface but also the center of circular contours were detected and those can also be used for selection

```
SGfigure; SG=SGTui(SG,'C'), view(70,+10);
```

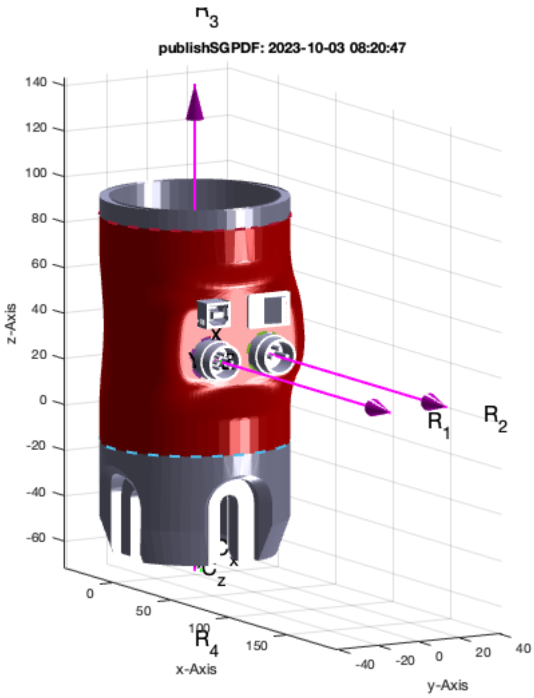
```
SG =
  struct with fields:

        VL: [15230×3 double]
        FL: [30472×3 double]
       col: 'w'
     alpha: 0.9000
     Tname: {'B'  'F'  'C'}
         T: {[4×4 double]  [4×4 double]  [4×4 double]}
      TFiL: {[206×1 double]  [86×1 double]  [108×1 double]}
      TFoL: {[]}
```



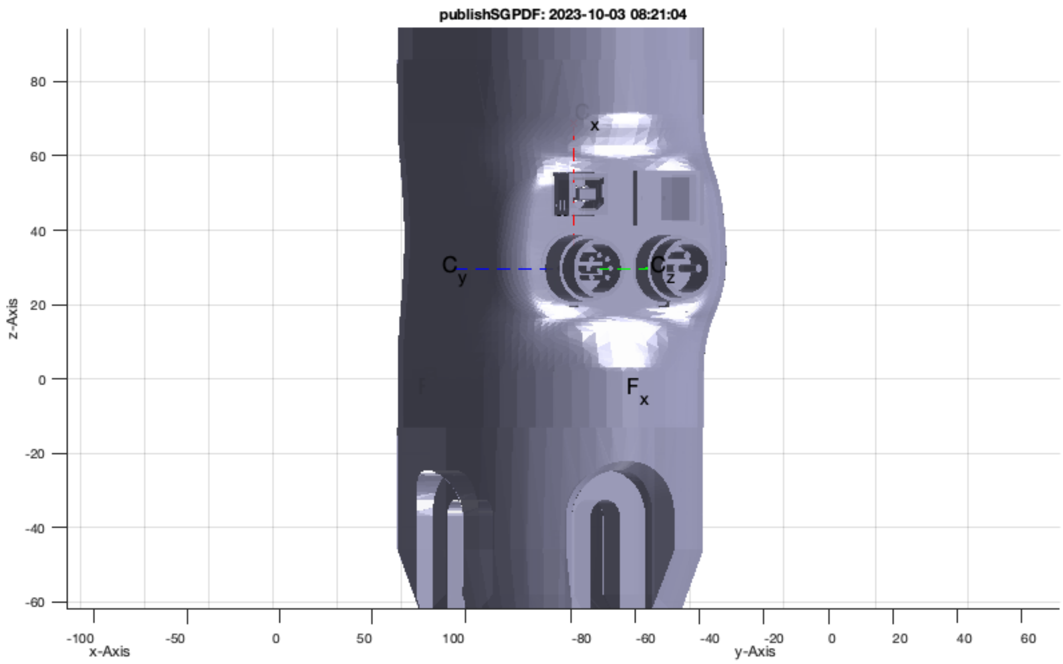
There is a slight difference between the center of the faces and the circle R1. By using 'R1' as parameter, the R1 coordinate system is used for the frame "C"

```
SGfigure; SG=SGTui(SG,'C','','R1'), view(60,+10);
```

```
SG =
  struct with fields:

        VL: [15230×3 double]
        FL: [30472×3 double]
       col: 'w'
     alpha: 0.9000
     Tname: {'B'  'F'  'C'}
         T: {[4×4 double]  [4×4 double]  [4×4 double]}
      TFiL: {[206×1 double]  [86×1 double]  [6776×1 double]}
      TFoL: {[]}
```

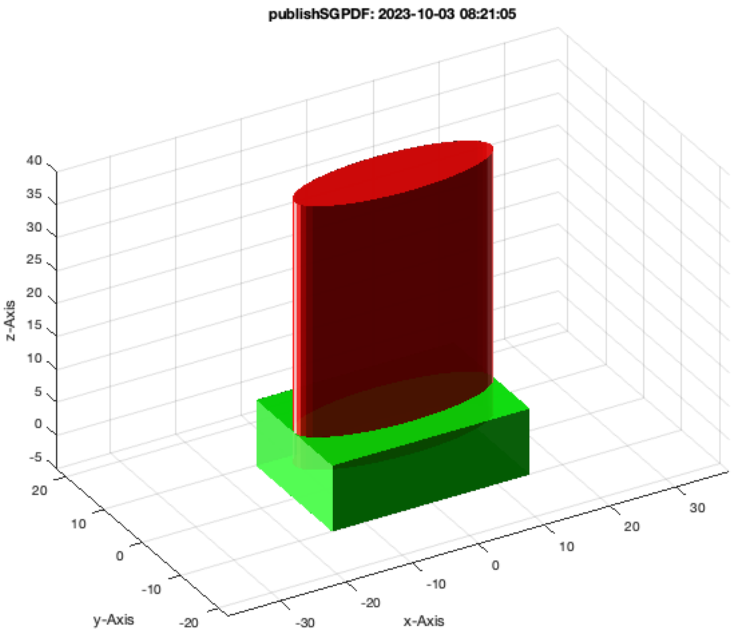



```
SGfigure;  SGTplot(SG,'C'); view(60,+0);
```



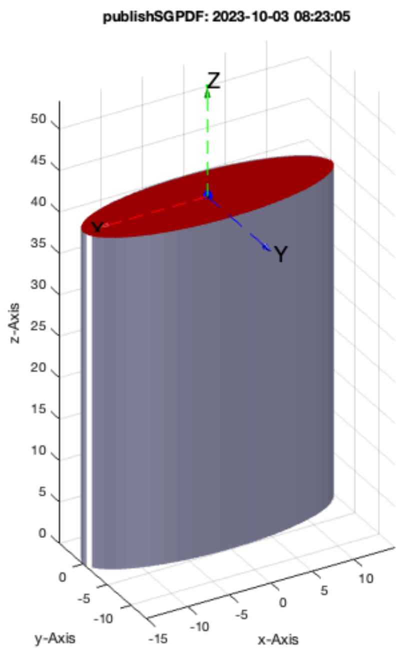
3. Spatial Arrangment of Solids relative to Frames

```
A=SGbox([30,20,10]); A.col='g'; A.alpha=0.9;
B=SGofCPLz(PLcircle(15,'','',5),40); B.col='r'; B.alpha=0.9;
SGfigure; SGplot({A,B}); view(-30,30);
```

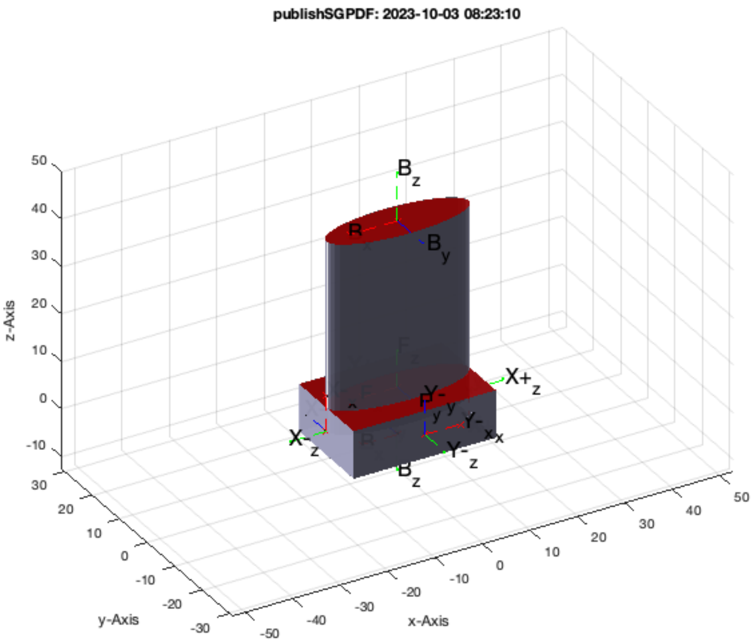


Now attach frame to both solids

```
A=SGTui(A,'F');   % Follower Frame
B=SGTui(B,'B');   % Base Frame
```



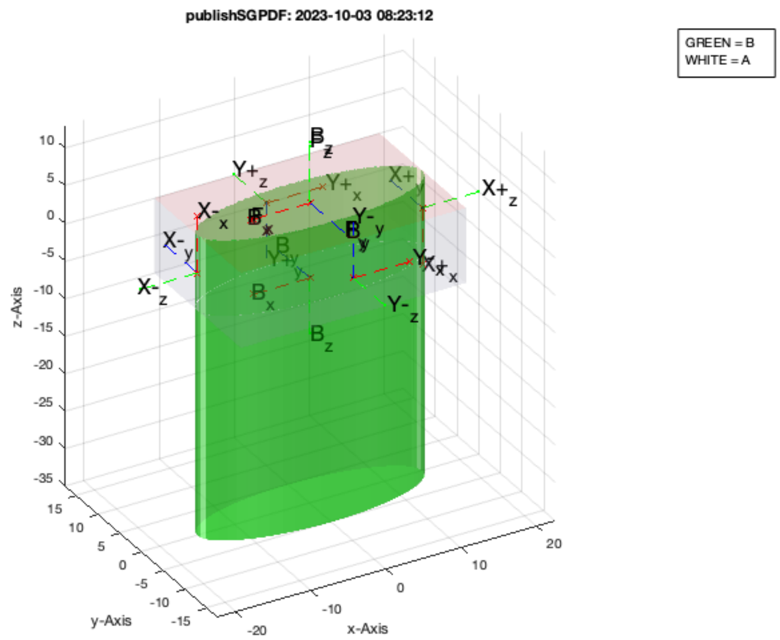

```
SGfigure; SGTplot(A); SGTplot(B); view(-30,30);
```



Now position solid B that its Frame 'B' matches with Frame 'F' of Solid A Afterwards, both Frames overlap completely.

```
SGtransrelSG(B,A,'matchT',{'B','F'})
```

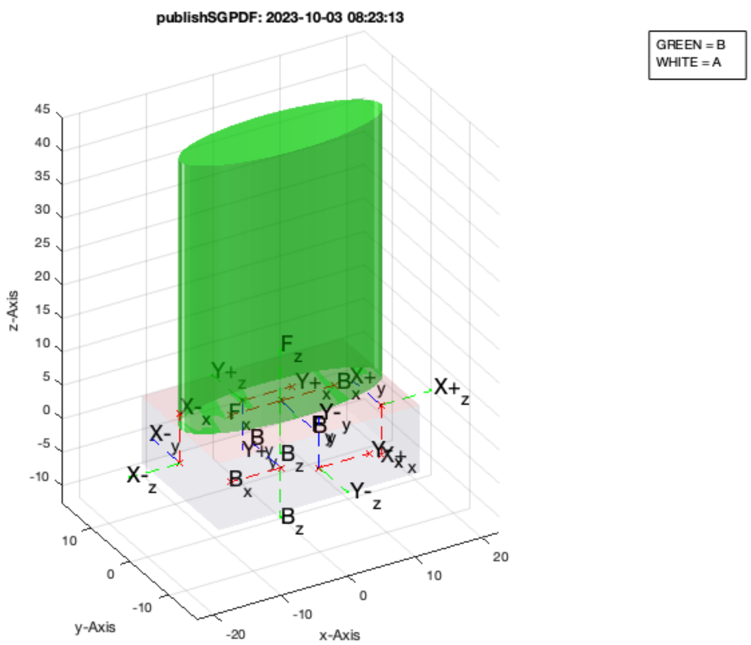
```
ans =
  struct with fields:

       VL: [110×3 double]
       FL: [216×3 double]
      col: 'r'
    alpha: 0.9000
    Tname: {'B'}
        T: {[4×4 double]}
     TFiL: {[53×1 double]}
     TFoL: {[]}
```



Now position solid B that its Frame 'B' aligns with Frame 'F' of Solid A Afterwards, both only axis Y overlap completely. Z and X have opposite orienations.

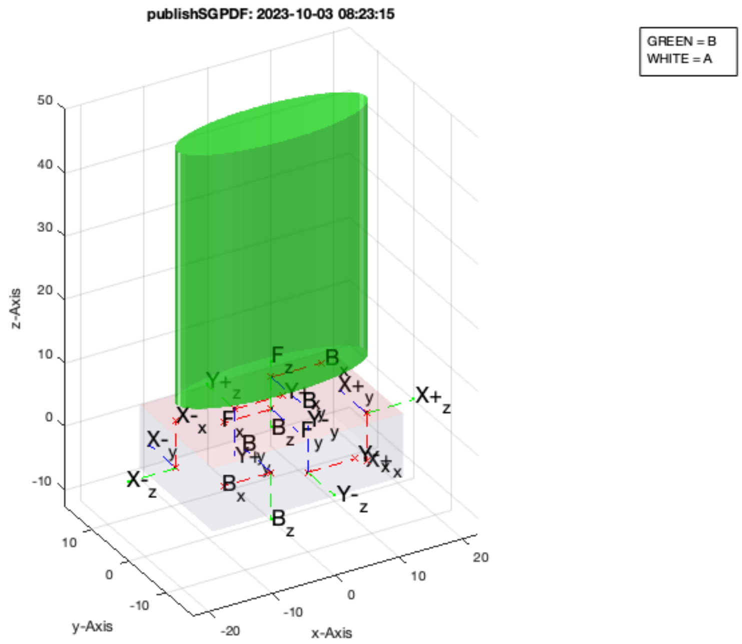
```
SGtransrelSG(B,A,'alignT',{'B','F'})
```

```
ans =
  struct with fields:

       VL: [110×3 double]
       FL: [216×3 double]
      col: 'r'
    alpha: 0.9000
    Tname: {'B'}
        T: {[4×4 double]}
     TFiL: {[53×1 double]}
     TFoL: {[]}
```



Now position solid B that its Frame 'B' aligns with Frame 'F' of Solid A Afterwards, both only axis Y overlap completely. Z and X have opposite orienations. IN ADDITION create a distance of 5 mm

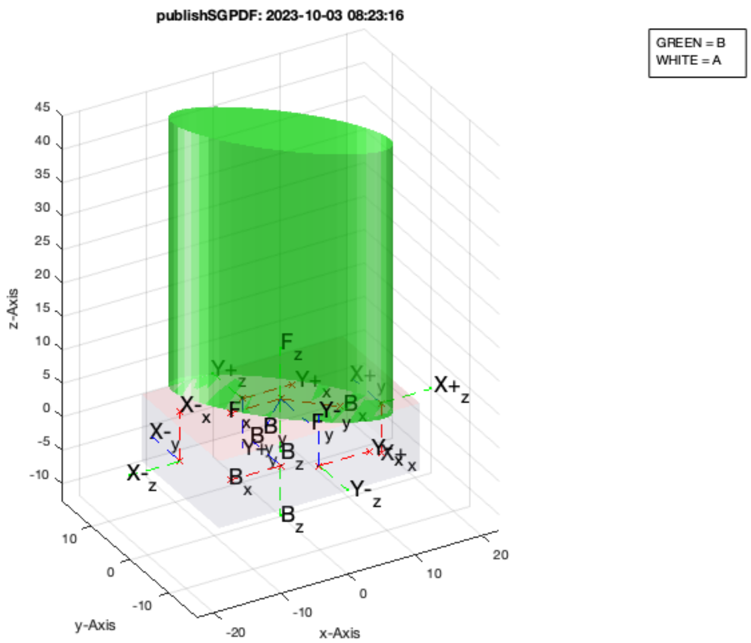
```
SGtransrelSG(B,A,'alignT',{'B','F',TofP([0 0 -5])});
```



Now position solid B that its Frame 'B' aligns with Frame 'F' of Solid A Afterwards, both only axis Y overlap completely. Z and X have opposite orienations. IN ADDITION TURN 45 degrees

```
SGtransrelSG(B,A,'alignT',{'B','F',TofR(rot(0,0,pi/4))})
```

```
ans =
  struct with fields:

       VL: [110×3 double]
       FL: [216×3 double]
      col: 'r'
    alpha: 0.9000
    Tname: {'B'}
        T: {[4×4 double]}
     TFiL: {[53×1 double]}
     TFoL: {[]}
```

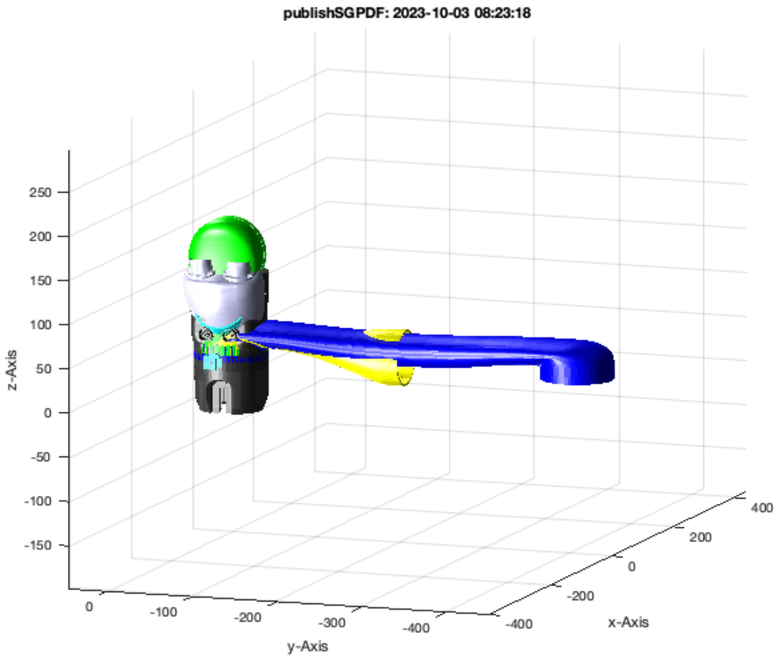


4. Simple Sequential Kinematic Chains

As soon as all solids have a base frame and a follower frame, it is possible to consider them als kinematic chain with some degrees of freedom between the frame. Such as rotation around the z-axis of the follower frame. The easist case is to define a cell list of all involved solids. To explain this feature, the origins of all solids are changed to their base frames. This is done just to avoid misunderstandings.

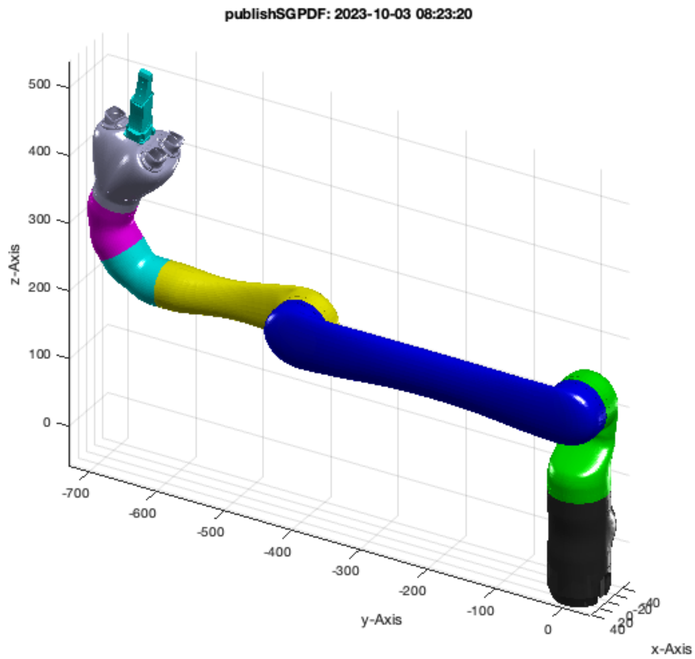
```
JC0=SGTsetorigin(JC0,'B'); % change the origin of Solid to Frame 'B'
JC1=SGTsetorigin(JC1,'B'); % change the origin of Solid to Frame 'B'
JC2=SGTsetorigin(JC2,'B'); % change the origin of Solid to Frame 'B'
JC3=SGTsetorigin(JC3,'B'); % change the origin of Solid to Frame 'B'
JC4=SGTsetorigin(JC4,'B'); % change the origin of Solid to Frame 'B'
JC5=SGTsetorigin(JC5,'B'); % change the origin of Solid to Frame 'B'
JC6=SGTsetorigin(JC6,'B'); % change the origin of Solid to Frame 'B'
JACO={JC0,JC1,JC2,JC3,JC4,JC5,JC6,JCF}
SGfigure; SGplot(JACO);   view(-70,10);
```

```
JACO =
  1×8 cell array
  Columns 1 through 4
    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}
  Columns 5 through 8
    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}
```



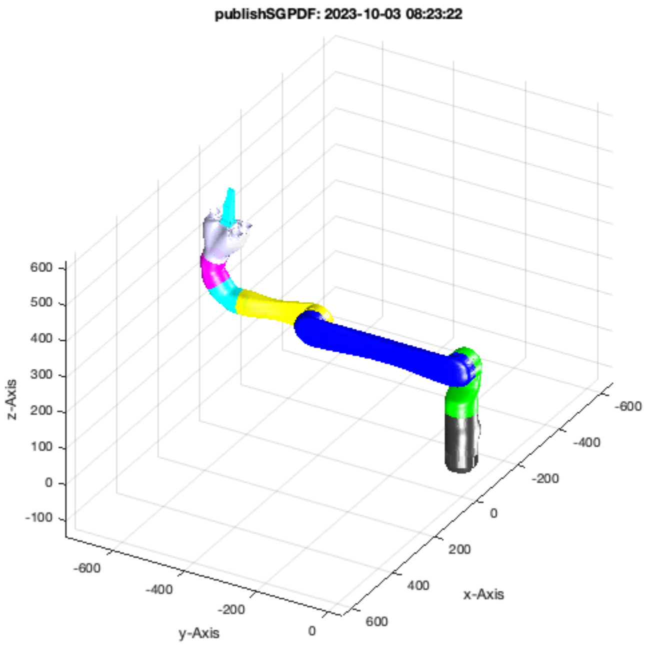
There is a function that aligns automatically base and follower frame AND modifies the vertex list for all of the solids but the first one.

```
SGfigure; SGchain(JACO); view(120,30);
```



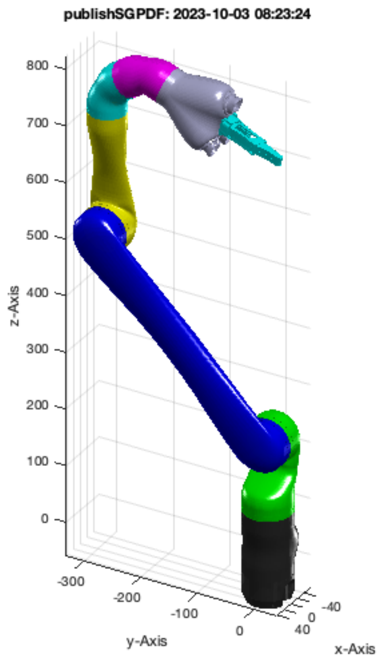
The function SGTchain changes the all vertex coordinates, therefor afterwards the parts seem to stay is space as the kinematic chain. In this example X is a pose of the robot if all frames are aligned. If X is plotted as a solid it looks like a robot in a specific pose.

```
X=SGTchain(JACO);
SGfigure; SGplot(X); view(120,30);
```



SGTchain also allow to deliver additional rotatorial parameters. For each joint a rotating angle can be specified. NEvertehelss, currently the first value is ignored, since there is no base frame. The nth rotation is relative to the base frame of the nth element.

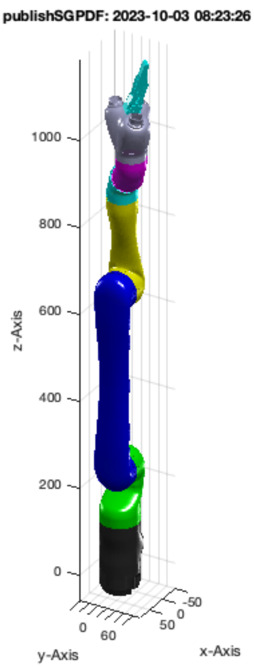
```
SGTchain(JACO,[nan 0 +pi/4 -pi/4]); view(120,30);

% again, the output value is the same surface cell list but describing
% exactly this position.
```



5. Calibration of a Sequential Kinematic Chain

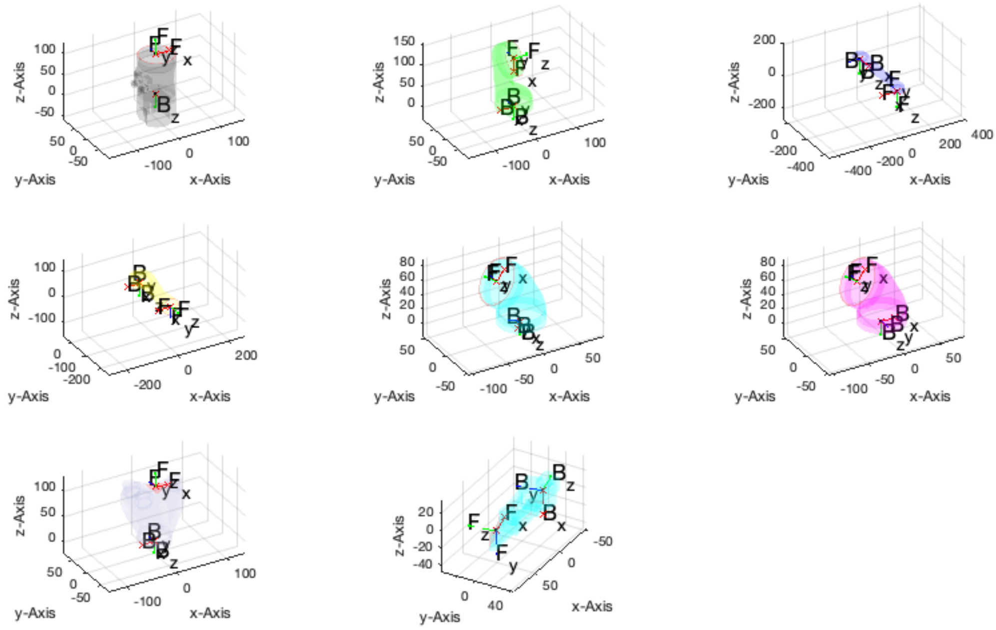
Often it is not possible to specify the frames using SGTui exactly as the real motor configuration is. Therefor it is necessary to calibrate the zero position. In case try to bring the robot by a set of rotating angles into the desired zero position or use an additional angle vector as offest. As soon as the offest is know call SGTcalibchain using the offset values. For example

```
SGTchain(JACO,[nan 0 pi/2 0 pi/4 -pi 0]); view(120,30);
```



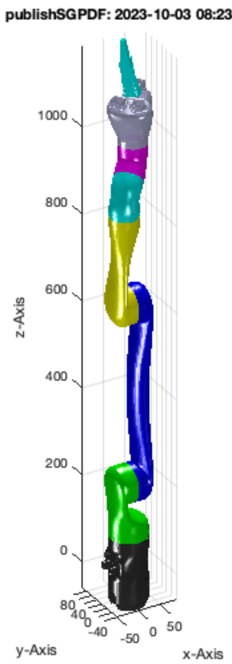
now change all frames of the chain to create a new zero position. In this case ALL elements need a value. Even the finger element.

```
SGTcalibchain(JACO,[nan 0 pi/2 0 pi/4 -pi 0 0]); view(120,30);
JACO_cal=ans;
```



Now the robot has a new zero position The position shown here has nothing to do with the real zero position of KINOVA's JACO robot.

```
SGTchain(JACO_cal);
```



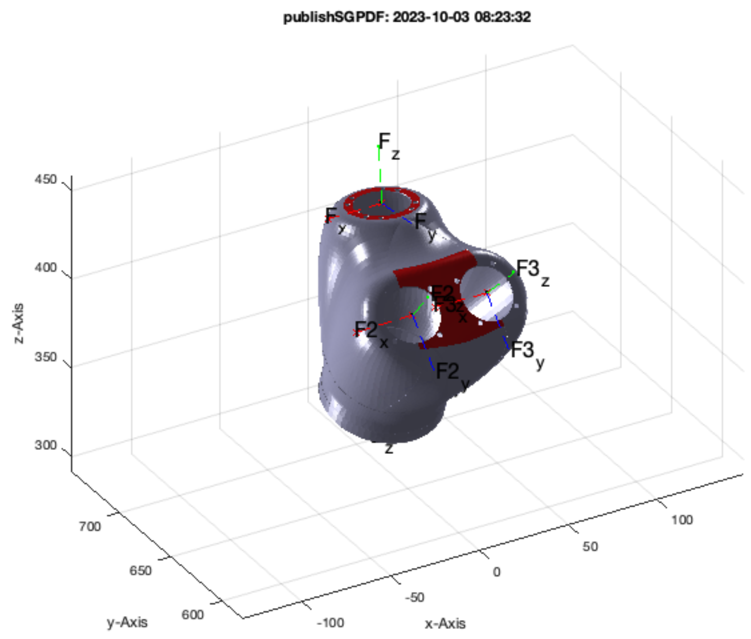
6. Creating Kinematic Trees

It is easy to see that the real JACO has three fingers and a simple chain is not enough. Therefor there is an additional format for SGTchain to explain the kinematic structure and the order of motors/angles. At first we need three follower frames. THis is part of solid JC61. Beside "F" tehre is also "F1" and "F2"

```
SGfigure; SGTplot(JC61); view(-30,30)
JACO={JC0,JC1,JC2,JC3,JC4,JC5,JC61,JCF}
```

```
JACO =
  1×8 cell array
  Columns 1 through 4
    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}
  Columns 5 through 8
```

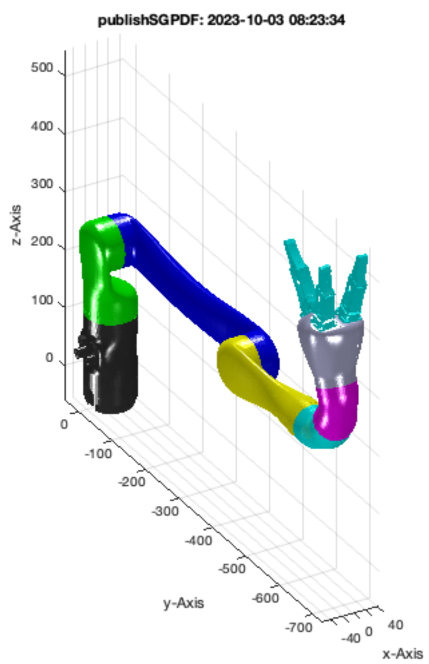
```
    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}
```



Next is to specify two additional degrees of freedom between Part 7 and Frame "F2" and Part 8 Frame "B" and Part 7 and Frame "F3" and Part 8 Frame "B". Automatically, there are two additional rotations or motors introduced. In case of the real JACO robot, the joints 7, 8, 9 are not rotational but linear for the fingers.

```
SGTchain(JACO,'','',1:8,[7 'F2' 8 'B', 7 'F3' 8 'B'])
```

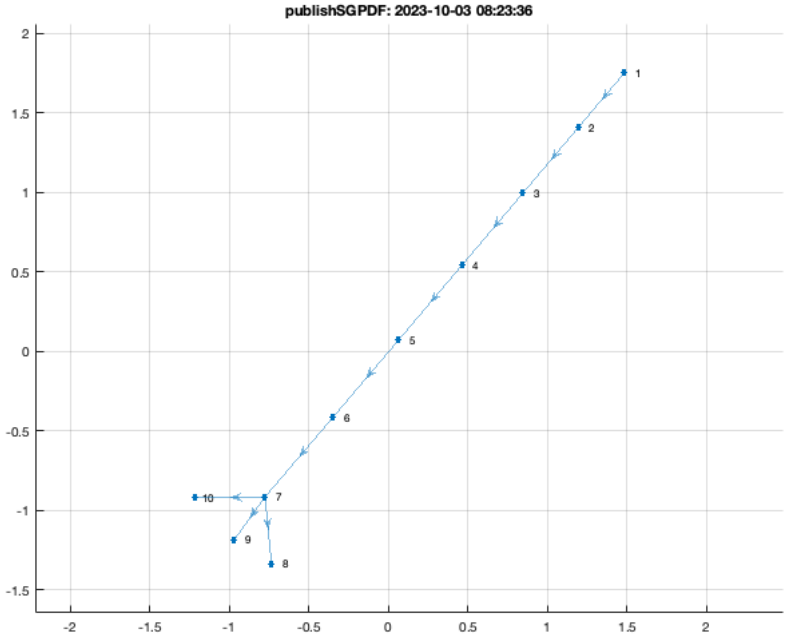
```
ans =
  1×10 cell array
  Columns 1 through 4
    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}
  Columns 5 through 8
    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}
  Columns 9 through 10
    {1×1 struct}    {1×1 struct}
```



To understand better the kinematic chains, it is also possible to call a auxiliary function to create a kinematic chain table. This function returns the number/order of the DoF and which frames are connected and which solid was used for the connection. In Future also the type of DoF will be added to this list

```
SGTframeChain(1:8,[7 'F2' 8 'B', 7 'F3' 8 'B'])
```

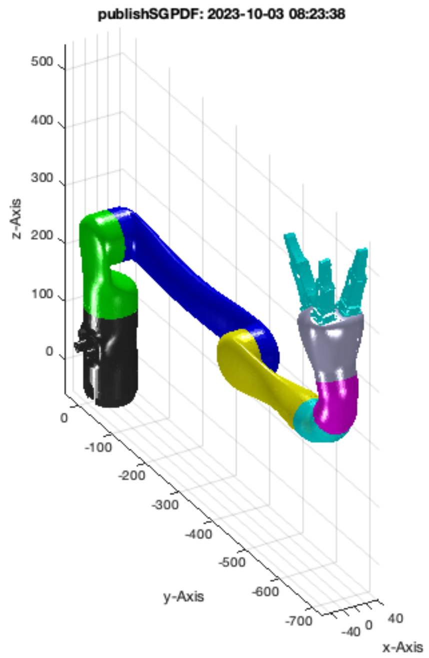
```
ans =
  10×5 cell array
    {[ 1]}    {'_' }    {'B'}    {[0]}    {[1]}
    {[ 2]}    {'F' }    {'B'}    {[1]}    {[2]}
    {[ 3]}    {'F' }    {'B'}    {[2]}    {[3]}
    {[ 4]}    {'F' }    {'B'}    {[3]}    {[4]}
    {[ 5]}    {'F' }    {'B'}    {[4]}    {[5]}
    {[ 6]}    {'F' }    {'B'}    {[5]}    {[6]}
    {[ 7]}    {'F' }    {'B'}    {[6]}    {[7]}
    {[ 8]}    {'F' }    {'B'}    {[7]}    {[8]}
    {[ 9]}    {'F2'}    {'B'}    {[7]}    {[8]}
    {[10]}    {'F3'}    {'B'}    {[7]}    {[8]}
```



It is also possible to call SGT directly using this table:

```
FC=SGTframeChain(1:8,[7 'F2' 8 'B', 7 'F3' 8 'B'])
SGTchain(JACO,'','',FC);
```

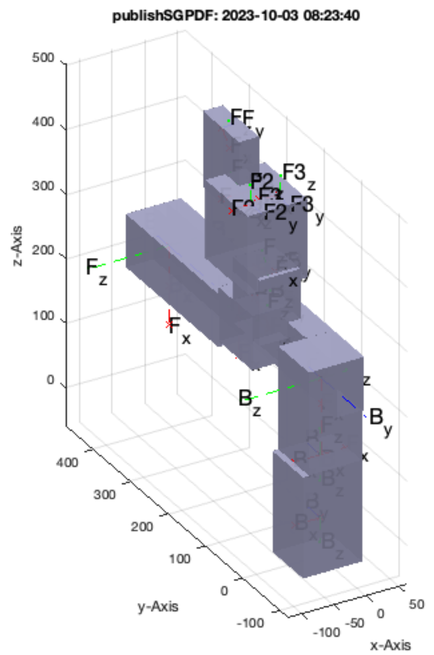
```
FC =
  10×5 cell array
    {[ 1]}    {'_' }    {'B'}    {[0]}    {[1]}
    {[ 2]}    {'F' }    {'B'}    {[1]}    {[2]}
    {[ 3]}    {'F' }    {'B'}    {[2]}    {[3]}
    {[ 4]}    {'F' }    {'B'}    {[3]}    {[4]}
    {[ 5]}    {'F' }    {'B'}    {[4]}    {[5]}
    {[ 6]}    {'F' }    {'B'}    {[5]}    {[6]}
    {[ 7]}    {'F' }    {'B'}    {[6]}    {[7]}
    {[ 8]}    {'F' }    {'B'}    {[7]}    {[8]}
    {[ 9]}    {'F2'}    {'B'}    {[7]}    {[8]}
    {[10]}    {'F3'}    {'B'}    {[7]}    {[8]}
```



7. Calculating Boxes for Quick Collision Checks

The algorithms for collision check are very time consuming since there is a need for testing all traingles for collision/penetration. This makes sensor for bollean operations but is not suitable for gast follision checks during a movement of a kinematic chain. Therefor there is a wish to perform these steps with a simplified kineamtic model, consisting of bounding boxes

```
J=SGTchain(JACO,[nan,0 pi pi]);

SGTBB(J); JB=ans; view(-30,30);
```



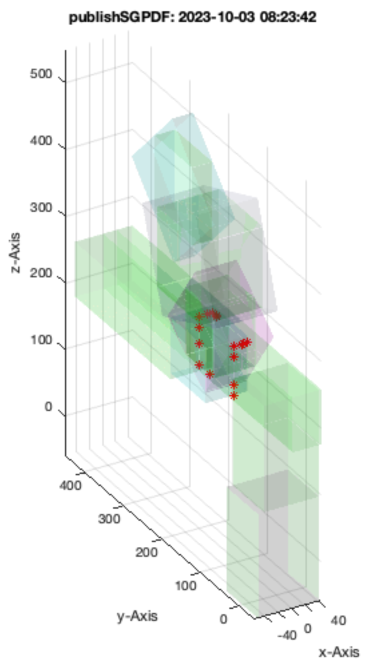
8. Collision Check

There are two functions:

- **iscollofVLBB** for testing of Vertices are inside of a bounding box
- **iscollofSG** for face testing of two solids or selftest of one solid Please read the documentation for both functions to see what is possible

```
% Self collision test in a safe configuration
iscollofSG(SGTchain(JB,[nan 0 pi pi]))
```

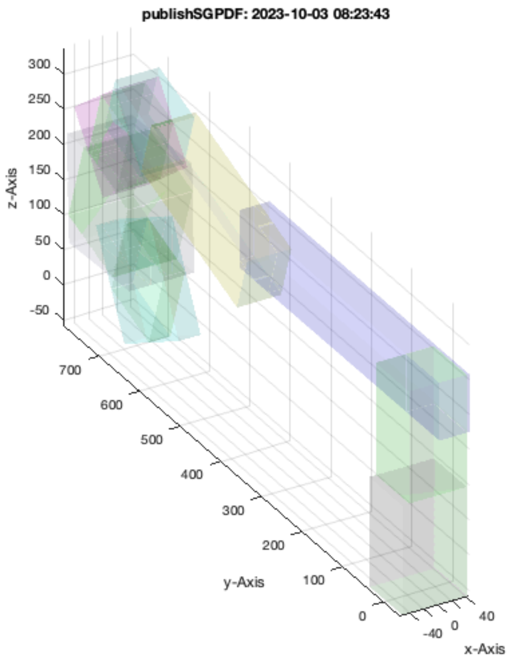
```
ans =
   13.5351  111.7080  257.2501
   13.5351  111.7080  257.2501
   21.7502  111.7080  257.2501
   21.7502  111.7080  257.2501
    1.4999  111.7080  257.2501
    1.4999  111.7080  257.2501
    1.4999  111.7080  184.4999
    1.4999  111.7080  241.2684
    1.4999  111.7080  184.4999
    1.4999  111.7080  241.2684
   14.7772  111.7080  257.2501
    1.4999  111.7080  200.3724
   14.7772  111.7080  257.2501
    1.4999  111.7080  200.3724
    1.4999  173.6560  184.4999
    1.4999  173.6560  184.4999
   21.7502  191.0959  257.2501
   21.7502  191.0959  257.2501
   21.7502  192.0389  257.2501
   21.7502  192.0389  257.2501
    1.4999  203.0001  241.2684
   13.5351  203.0001  257.2501
    1.4999  203.0001  241.2684
   13.5351  203.0001  257.2501
   21.7502  203.0001  257.2501
   21.7502  203.0001  257.2501
    1.4999  203.0001  215.6650
    1.4999  203.0001  257.2501
    1.4999  203.0001  184.4999
    1.4999  203.0001  184.4999
    1.4999  203.0001  215.6650
    1.4999  203.0001  257.2501
```



Self collision test in a problematic configuration

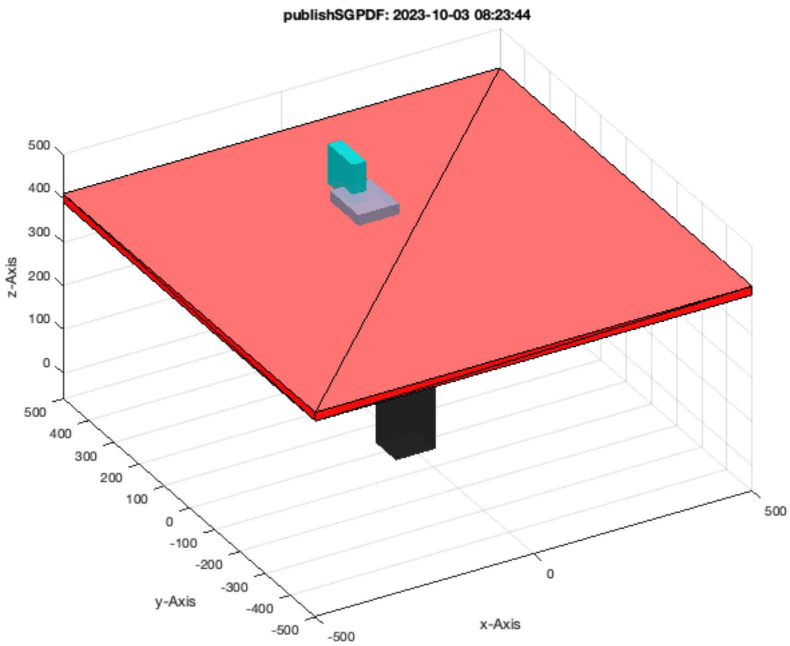
```
iscollofSG(SGTchain(JB,[nan 0 pi pi/10]))
```

```
ans =
  0×3 empty double matrix
```



Collsion collision test in a problematic configuration

```
A=SGbox([1000,1000,20]); A=SGtransP(A,[0 0 400]);
SGTchain(JB,[nan 0 pi pi]); SGplot(A);
```

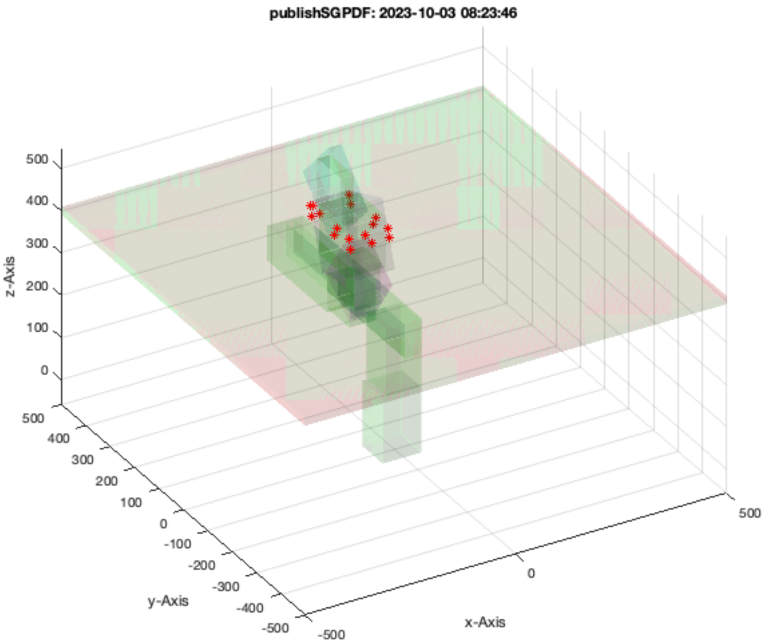


Now make a test for crossing robot and solid

```
iscollofSG(SGTchain(JB,[nan 0 pi pi]),A)
```

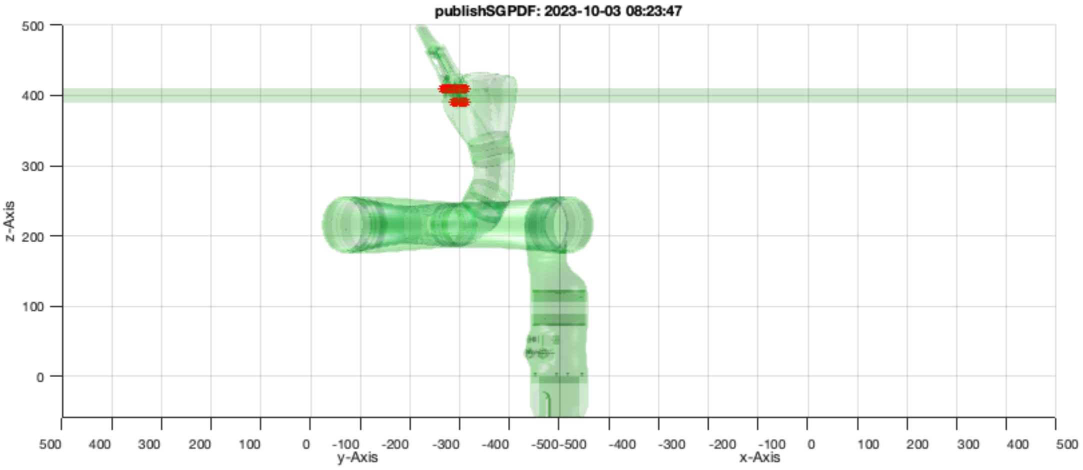
```
ans =
   36.9130   82.2649  389.9999
   36.9130   82.2649  389.9999
   -3.9110   82.2649  389.9999
   -3.9110   82.2649  389.9999
  -55.9248   82.2650  389.9999
  -55.9248   82.2650  389.9999
   36.9130   89.5443  410.0001
```

```
  36.9130   89.5443  410.0001
 -16.8148   89.5444  410.0001
 -16.8148   89.5444  410.0001
 -55.9248   89.5444  410.0001
 -55.9248   89.5444  410.0001
  36.9130  138.7881  410.0001
  36.9130  138.7881  410.0001
 -55.9248  138.7881  410.0001
 -55.9248  138.7881  410.0001
  36.9130  147.7560  389.9999
  36.9130  147.7560  389.9999
 -55.9248  147.7560  389.9999
 -55.9248  147.7560  389.9999
  36.9131  240.4606  389.9999
  36.9131  240.4606  389.9999
 -36.7140  240.4606  389.9999
 -36.7140  240.4606  389.9999
 -55.9247  240.4607  389.9999
 -55.9247  240.4607  389.9999
  36.9131  247.7400  410.0001
  36.9131  247.7400  410.0001
 -49.6179  247.7401  410.0001
 -49.6179  247.7401  410.0001
 -55.9247  247.7401  410.0001
 -55.9247  247.7401  410.0001
```



*The full test with the original geometry is much slower if the collision objects have more facets than those 12 of the simple box!

```
iscollofSG(SGTchain(JACO,[nan 0 pi pi]),A,true); view(-45,0)
```



Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2023-Oct-03), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 06-Jul-2078 08:23:49!
Executed 03-Oct-2023 08:23:51 by 'timlueth' on a MACI64 using Mac OSX 13.6 | R2023a Update 5 | SG-Lib 5.4
======================================== Used Matlab products: =====================================
database_toolbox
distrib_computing_toolbox
fixed_point_toolbox
image_toolbox
map_toolbox
matlab
simmechanics
simscape
simulink
====================================================================================================
```

Published with MATLAB® R2023a