Tutorial 62: Design of Monolithic Snake-like Manipulators BY YANNIK KRIEGER

Yannick Krieger/Korbinian Rzepka, MIMED, Prof. Lueth - Technische Universität München, Germany (URL: http://www.SG-Lib.org) - Last Change: 2021-02-25

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

Motivation for this tutorial: (Originally SolidGeometry 5.0 required)

This tutorial describes the basic function for the design of monolithic manipulator structures, which can be produced by additive manufacturing. Part of the code that was developed in the context of the thesis of Korbinian Rzepka and Simon Schiele was included in the present design function for monolithic manipulator structures.

Introduced Function:

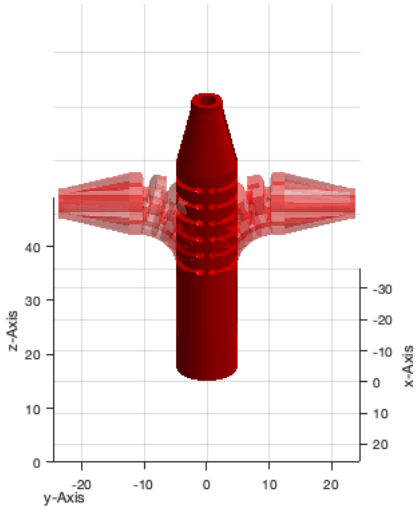
- SGmanipulator - Generates STL-modells monolithic manipulators based on flexure hinge structures to be 3D-printed using selective lasersintering.

Generating a simple manipulator arm

Example of a simple manipulator structure consisting of one single section

```
SGmanipulator({PLcircle(5)},...     CPL to define contour of arm
                          3,...     Diameter for tool channel at [0 0]
                          0,...     Orientation of hinge
                         15,...     Section length
                    'tip');         ...Flag to not generate shaft
view(90,30);
```

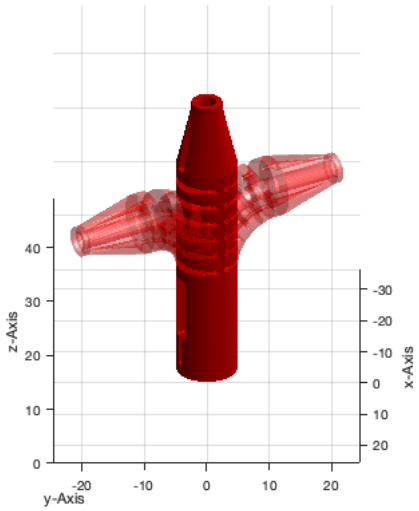
```
Warning: File:
/Applications/MATLAB_R2023a.app/toolbox/shared/statslib/+statslib/+internal/pdist2.m
Line: 184 Column: 1
Unrecognized pragma "%#exclude statslib.gpu.pdist2".
```



Orientation angle

Orientation angle changes the orientation of the rotation axis of the hinge

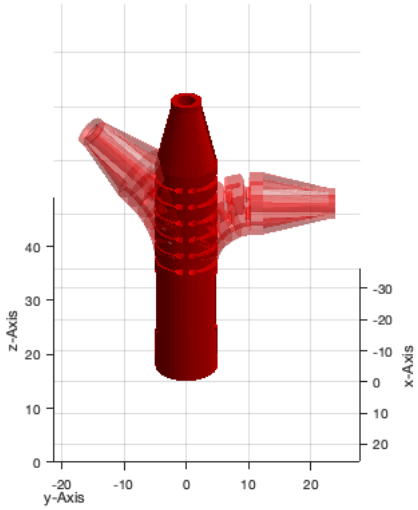
```
SGmanipulator({PLcircle(5)},...
                          3,...
                         30,...     Previous Example with 30 degree orientation angle. Value range: -180 to 180.
                         15,...
                      'tip');
view(90,30);
```



Deflection angles

Besides orientation angle the deflection angles respectively bending angles of a section can be specified.

```
SGmanipulator({PLcircle(5)},...
                          3,...
                  [0 45 90],...     [Deflection angle 1, Deflection angle 2]
                         15,...
                    'tip');
view(90,30);
```

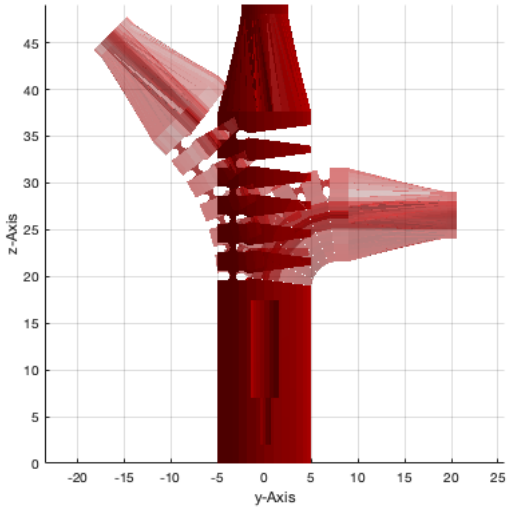


Optimizing for one bending direction

The hinge can be optimized for one bending direction.

```
SGmanipulator({PLcircle(5)},...
                        3,...
                [0 45 90 1],...     Angle flag set to 1 optimizes hinge position for second deflection angle. First angle gets ignored. -1 for the firs
                       15,...
                    'tip');
view(90,0);
```

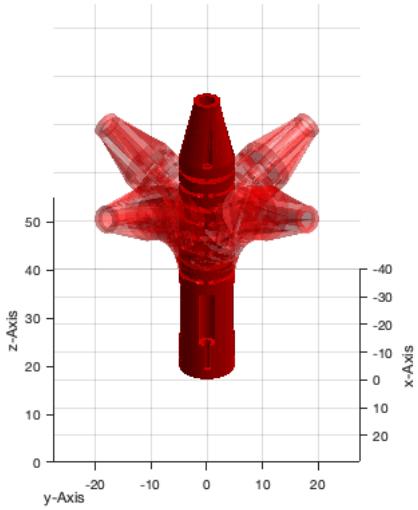
```
Warning: Width of flexure hinge needs to be specified, when hinge should be
optimized. Minimal hinge width set to 2
```



Alternating sections.

The fourth angle parameter set to 2 generates alternating sections. Flexure hinges are arranged alternating around +/-90 deg. Therefore, the section has two bending directions.

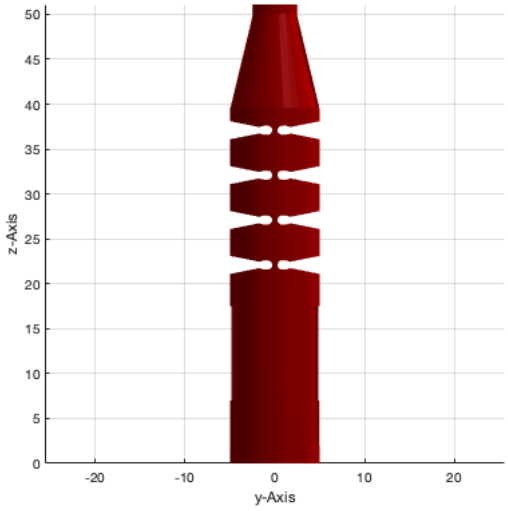
```
SGmanipulator({PLcircle(5)},...
                        3,...
                [0 90 90 2],...     Angle flag set to 2 generates an alternating section. Second angle is offset 90 degrees to the original angle.
                       15,...
                    'tip');
view(90,30);
```



Element Height

With the argument after section length the element height can be altered.

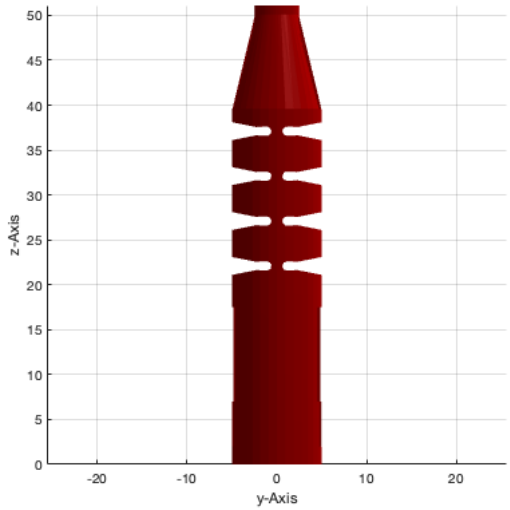
```
SGmanipulator({PLcircle(5)},...
                          3,...
                          0,...
                     [15 4],...     Element height set to 4. Default is 2.
                     'tip');
view(90,0); delete(findobj('type', 'patch','FaceAlpha',0.3));
```



Hinge thickness

The second argument behind section lengths specifies the hinge thickness.

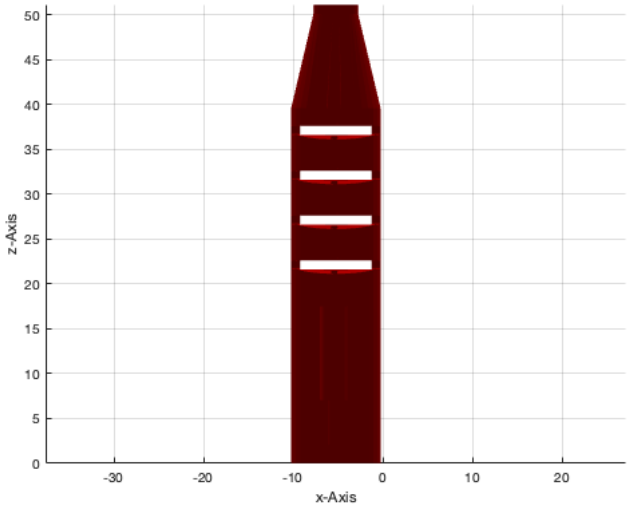
```
SGmanipulator({PLcircle(5)},...
                          3,...
                          0,...
                   [15 4 1.25],...     Hinge thickness set to 1.25. Default is 0.6
                     'tip');
view(90,0); delete(findobj('type', 'patch','FaceAlpha',0.3));
```



Hinge width

The third argument behind section lengths specifies the hinge width.

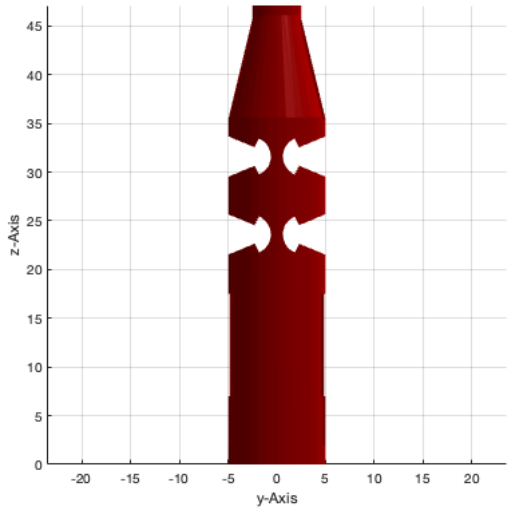
```
SGmanipulator({PLcircle(5)},...
                            3,...
                            0,...
                  [15 4 1.25 1],...     Hinge length set to 1. Default is 2. Setting this value to 0 projects the hinge over the whole element.
                      'tip');
view(0,0); delete(findobj('type', 'patch','FaceAlpha',0.3));
```



Hinge radius

The fourth argument behind section lengths specifies the hinge radius and therefor its height. Default is 0.5 mm. A radius of 1 mm is also a good standard value to use.

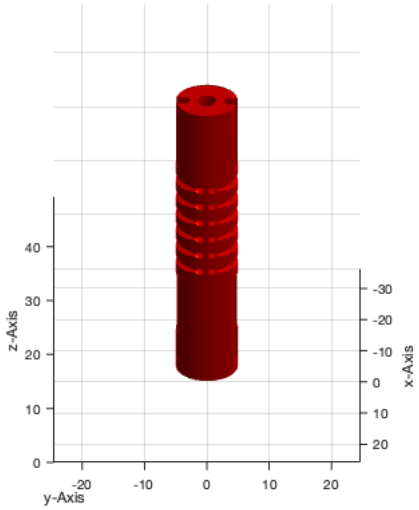
```
SGmanipulator({PLcircle(5)},...
                            3,...
                            0,...
                [15 4 1.25 1 2],...     Hinge radius set to 2. Default is 0.5.
                      'tip');
view(90,0); delete(findobj('type', 'patch','FaceAlpha',0.3));
```



Flat Tip

To generate a straigth flat tip instead of a pointy one the flat_tip flag can be used.

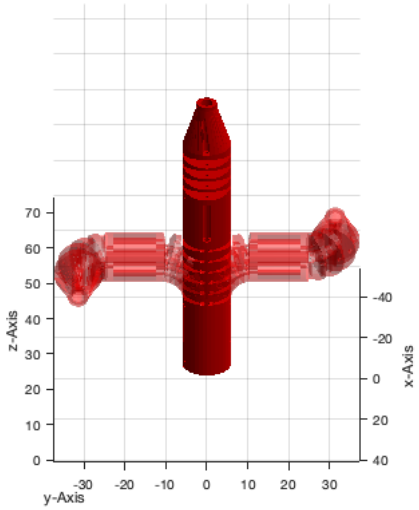
```
SGmanipulator({PLcircle(5)},...
                           3,...
                           0,...
                          15,...
                 'flat_tip',...
                     'tip');
view(90,30); delete(findobj('type', 'patch','FaceAlpha',0.3));
```



Multiple sections

To generate multiple sections per arm an angle and length value need to be set for each section.

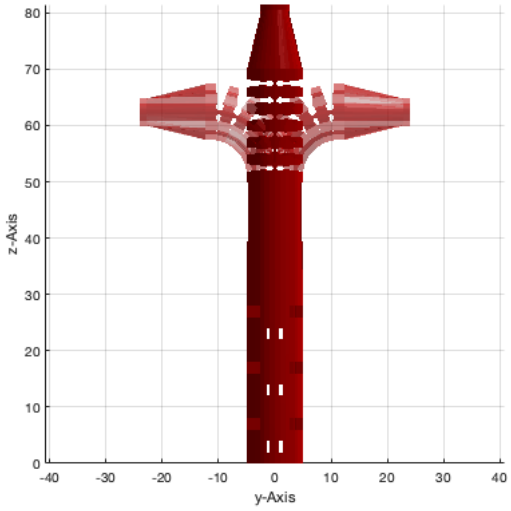
```
SGmanipulator({PLcircle(6)},...
                           3,...
                      [0;90],...    Second angle value.
                     [15;10],...    Second length value.
                       'tip');
view(90,30);
```



Shaft length

Specify the length of a rigid shaft.

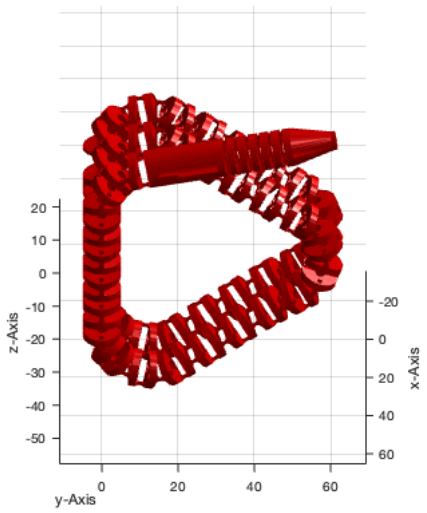
```
SGmanipulator({PLcircle(5)},...
                           3,...
                           0,...
                          15,...
              'length',50);        ...Parameter to set shaft length. Value is in mm. Can not be shorter than crimp connector which is around 16mm
view(90,0);
```



Flexible shaft

Specify the length of a passive flexible.

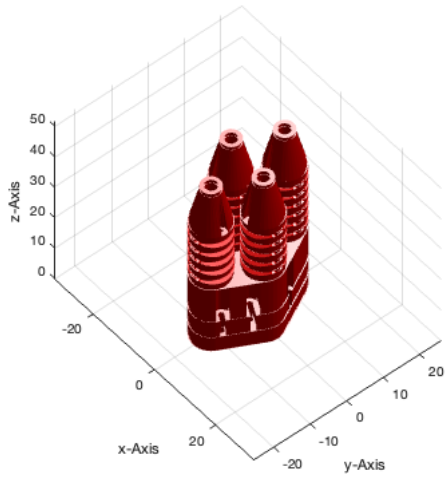
```
SGmanipulator({PLcircle(5)},...
                           3,...
                           0,...
                          15,...
             'flex',[40,10],...        Two angle values need to be provided for fitting into printer workspace.
              'length',250);        ...Parameter to set shaft length
view(90,30);delete(findobj('type', 'patch','FaceAlpha',0.3));
```



Four-arm manipulator

Up to four arms the arms will be positioned in a predefined way for optimal space usage.

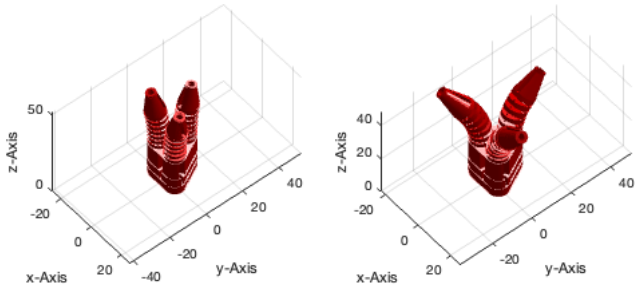
```
CPL = PLcircle(5);
SGmanipulator({{CPL},{CPL},{CPL},{CPL}},...
                              [3;3;3;3],...
                              {0,0,0,0},...
                            {15,12,15,12},...
                    'length',20);         ...Parameter to set shaft length
view(50,50); delete(findobj('type', 'patch','FaceAlpha',0.3));
```



SGTchain of the manipulator and deflection angles.

It is possible to return the framechain and max angles used to generate the manipulator. With those a new configuration of the SGTchain of the manipulator can be generated. The phis vector includes current defelction angles as well as min/max angles for each flexure hinge element of the manipulator's SGTchain.

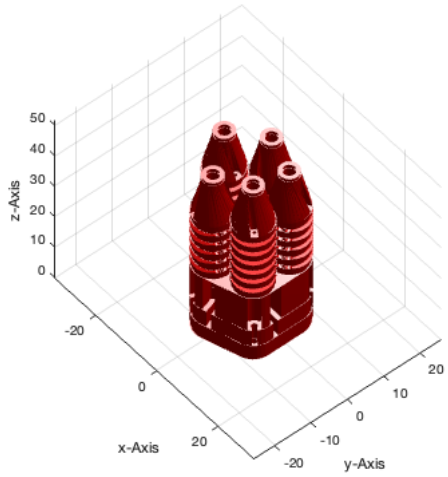
```
CPL = PLcircle(5);cla;subplot(1,2,1);
[SG,SGc,framechain,phis] =SGmanipulator({{CPL},{CPL},{CPL}},...
                                                [3;3;3],...
                                              {-45,-90,0},...
                                               {15,12,15},...
                                  'deflection',[0.1;0.1;0.1],...       % Generates a manipultor with 10% deflection in each section.
                                               'length',20);
SGplot(SG);VLFLplotlight;view(50,50); subplot(1,2,2);
SGplot(SGTchain(SGc,phis(2,:)*0.5,'',framechain));VLFLplotlight;      % Generates same manipulator with 50% deflection in each section
view(50,50); delete(findobj('type', 'patch','FaceAlpha',0.3));
```



Multiple arms

For more than 4 arms the arms will be placed in a circular pattern.

```
CPL = PLcircle(5);
SGmanipulator({{CPL},{CPL},{CPL},{CPL},{CPL}},...
                                [3;3;3;3;3],...
                              {90,90,90,90,90},...
                              {15,12,15,12,12},...
                                    'radial',...     Radial flag must not be used, if number of arms is greater than 4.
                                  'length',20);       ...Parameter to set shaft length
view(50,50); delete(findobj('type', 'patch','FaceAlpha',0.3));
```



Example for endoscopic manipulator

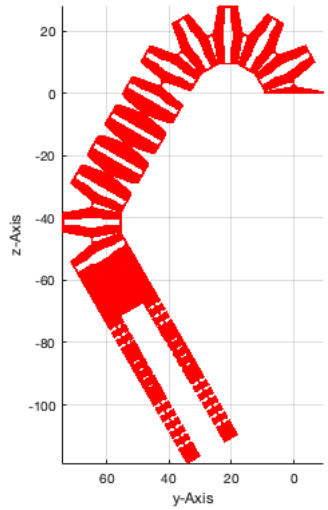
Design of a dual-arm endoscopic manipulator with passive flexible shaft

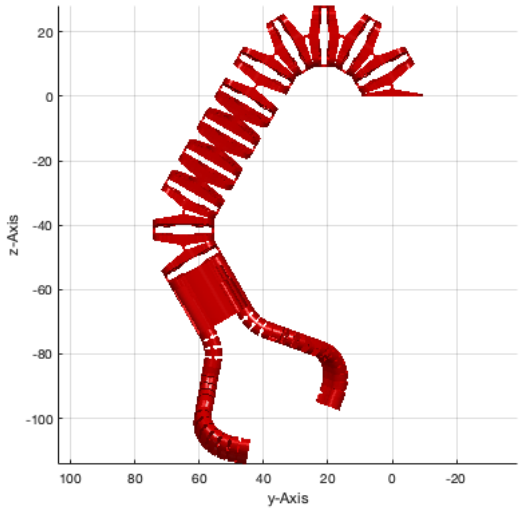
```
CPL = PLtrans(PLkidney(9.5,14.5,pi/5.5),[-12 0]);
CPL2 = PLtrans(PLkidney(9.5,14.5,pi/6.5),[-12 0]);
CPL3 = PLtrans(PLkidney(9.5,14.5,pi/8.5),[-12 0]);

[SG,SGc,framechain] = SGmanipulator({{CPL;CPL2;CPL3},{CPL;CPL2;CPL3}},...
                                        [2;2],...
                                            ...
             {[90 10 50 0;0 60 60 0;90 90 90 0],...
             [90 10 50 0;0 60 60 0;90 90 90 0]},...
                                            ...
                                    {[8;10;15],...
                                    [8;10;15]},...
                                            ...
                                    'single',...
                                 'optic_mid',...     Adds a channel between the manipulator arms.
                               'optic_radius',4,...     Defines the radius of the optic channel between the arms.
                                 'flex',[30,10],...
```

```
                                'length',100,...
        'deflection',[-0.8,-0.7,1;-0.8,-0.7,1]);        ...Flag to pre deflect the section in percent respective to their maximum deflection.


cla;SGud=SGTchain(SGc,'','',framechain); %undeflected manipulator arms
SGplot(SGud);view(-90,0);VLFLplotlight(1,0.9);
figure;SGplot(SG);view(-90,0);VLFLplotlight(1,0.9);
```





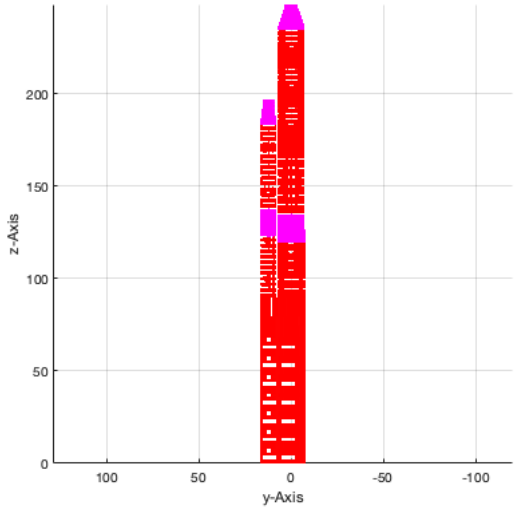
Example for laparoscopic manipulator

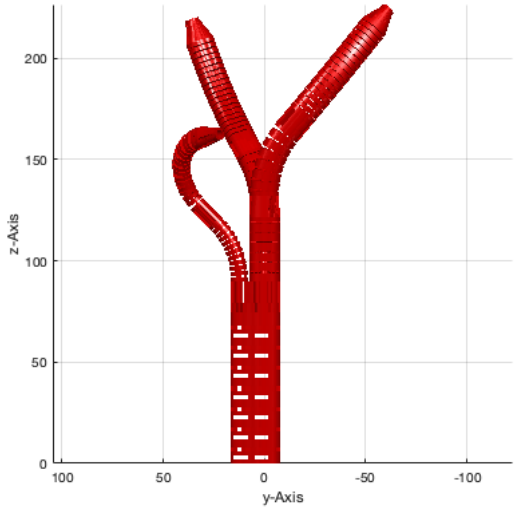
Design of a laparoscopic manipulator with rigid shaft with two manipulator arms and one additional actuated camera arm

```
CPL_camera = PLtrans(PLkidney(8,16.5,0.2),[-12 0])*rot(-pi/2);
CPL = PLtrans(PLkidney(7,16.5,0.5),[-12 0]);
CPL2 = PLtrans(PLkidney(6,15.5,0.45),[-12 0]);

[SG,SGc,framechain] = SGmanipulator({{CPL;CPL2},{CPL;CPL2},{CPL_camera}},...
                                                        [5;5;4],...
                                                          ...
                                {[90 40 40 1;0 80 70 0;90 160 160 0],...
                                 [90 40 40 1;0 45 45 0;90 160 160 0],...
                                           [0 90 45 -1;0 250 250 2]},...
                                                          ...
                        {[27 4 1.2 3 0.5;30 4 1.0 0 0.5;55 2 0.8 4 0.5],...
                         [27 4 1.2 3 0.5;30 4 1.0 0 0.5;55 2 0.6 4 0.5],...
                                   [30 2 1 2 0.5;40 2 0.6 1 0.5]},...
                                                          ...
                        'deflection',[-0.1 0.5 0.2;-1 0.5 0.2;1 -0.6 0],...
                                                    'first_single',...      Flag to generate the first section as a section with only one bowdencable
                                                'hole_radius',0.75,...      This flag is used to alter the radius for the bowdencable channels. Defau
                                                     'length',90);

cla; SGud=SGTchain(SGc,'','',framechain);  %undeflected manipulator arms
SGplot(SGud);view(-90,0);VLFLplotlight(1,0.9);
figure;SGplot(SG);view(-90,0);VLFLplotlight(1,0.9);
```





Bowden wires

For use in manipulators that use a push rod to transmit push and pull forces (e.g. endoscopic manipulator), Microlumen (Oldsmar, USA) sleeves have proven to be advantageous: PTFE coated wire braid made of 1.4301 stainless steel wire (Microlumen Code 215-VI Pure PTFE ID/BRD)

For double pull systems (e.g. laparoscopic manipulator) with pure tensile load for the bowden wire, simple coiled tension spring strands showed to be the most advantageous. Especially with higher applied tensile forces in case of electical robotic actuation. Tension spring strands of 1.4310 stainless steel with a 0.3 mm thick single coiled spring wire (LUPA Präzisionsfedern, Hardt, Germany) were used.

As wire a 0.35mm 1.4310 draw-hard stainless steel wire is used.

Published with MATLAB® R2023a