Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import

2014-11-18: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.sg-lib.org) - Last Change: 2019-06-11

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
- Tutorial 39: HEBO Modules robot design
- Tutorial 40: JACO Robot Simulation and Control
- Tutorial 41: Inserting Blades, Cuts and Joints into Solid Geometries
- Tutorial 42: Performing FEM Stress and Displacement Analysis and Structural Optimization of Solids
- Tutorial 43: Performing FEM Structural Optimization (CAO) and Topological Optimization (SKO) of Solids
- Tutorial 44: Creation of solids and kinematics from 3D curves and transformation matrices
- Tutorial 45: Creation of Solids using the SG-Coder - SGofCPLcommand
- Tutorial 46: Creating Fischertechnik compatible gear boxes using SGofCPLcommand
- Tutorial 47: Creating four-joints by 3 pose synthesis

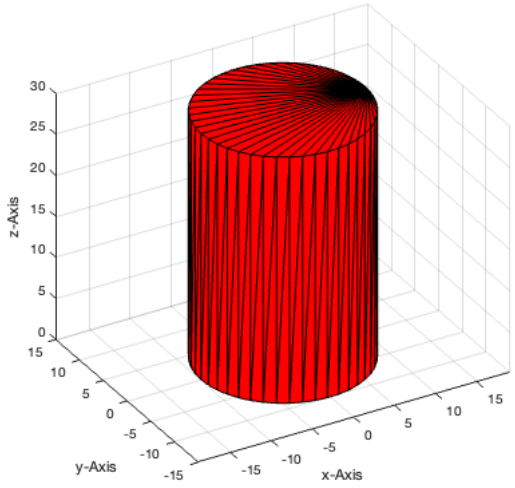
Motivation for this tutorial: (Originally SolidGeometry 1.1 required)

2. Import and export of STL-files in ASCII format and binary format

Often it is useful to import surface data fo solid volumes from STL-Files generated by other programs such as CATIA, ProEngineer, Solidworks etc. On the other hand we want to export our data for documentation, 3D-printing or the exchange with other users. The STL-File format is the most common file format for surface models. It supports ascii-text-format and binary formatted files. For export and import we need a couple of functions:

- **VLFLwriteSTL** for writing STL-files in ascii file format.
- **VLFLwriteSTLb** for writing STL-files in binary file format.

```
close all;
PL=PLcircle(10);
[VL,FL]=VLFLofPLz (PL,30);
VLFLplot(VL,FL); view (-30,30); grid on;
```




```
VLFLwriteSTL(VL,FL,'STL-ASCII','by My Name');
VLFLwriteSTLb(VL,FL,'STL-BINAR','by My Name');
```

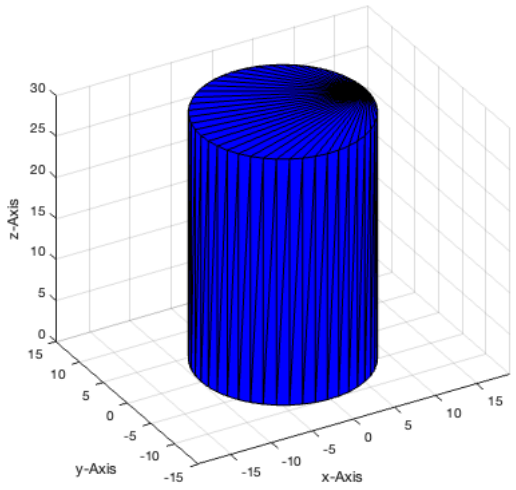
```
WRITING STL FILE /Users/timlueth/Desktop/STL-ASCII.STL in ASCII MODE
completed.
WRITING STL (90 vertices, 176 facets) FILE /Users/timlueth/Desktop/STL-BINAR.STL in BINARY MODE
completed.
```

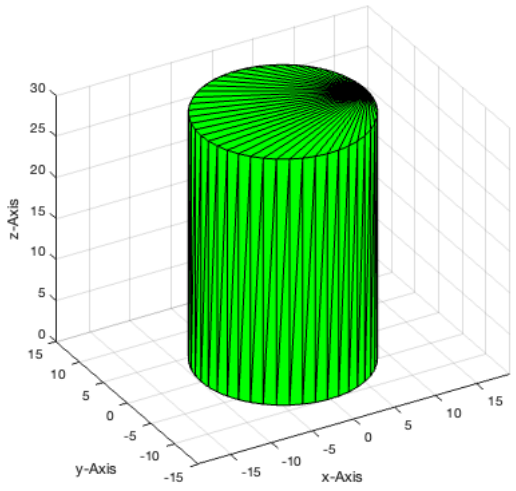
Similar it is possible to read the files in again

- **VLFLreadSTL** for importing STL-files in ascii file format.
- **VLFLreadSTLb** for importing STL-files in binary file format.

```
close all;
[VL,FL]=VLFLreadSTL ('STL-ASCII');
figure(1); VLFLplot(VL,FL,'b'); view (-30,30); grid on;
[VL,FL]=VLFLreadSTLb ('STL-BINAR');
figure(2); VLFLplot(VL,FL,'g'); view (-30,30); grid on;
```

```
LOADING ASCII STL-File: /Users/timlueth/Desktop/STL-ASCII.STL scaling factor: 1
Processing 1234 lines:
Finishing solid AOI-LIB:"STL-ASCII by My Name" 03-Oct-2023 07:09:56 03-Oct-2023 07:09:56 LOADING BINARY STL-File: /Users/timlueth/Desktop/STL-BINAR.STI
Binary Header: COLOR=RGBA,MATERIAL=AAAABBBBCCCCDDDD;SOLID "STL-BINAR by My Name" 03-Oct-2023 07
Color of solid defined as: "k"
Alpha of solid defined as: 65.00
Number of facets: 176
Number of vertices: 90
```





3. Checking surface volume data and STL-files

Especially, when reading in STL-Files that are generated by other programs and libraries it makes sense to check the data quality. For that purpose there is a function that will be explained later in more detail. This function is called at the end of each STL import.

- **VLFLchecker** is used to analyze vertex list (VL) and facet list (FL)

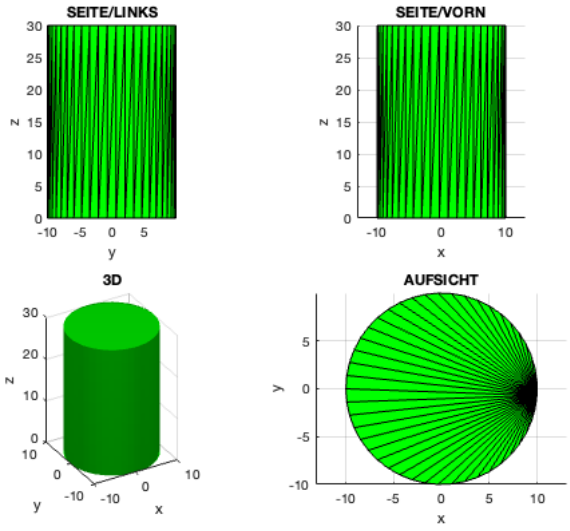
```
VLFLchecker(VL,FL);     % Check the data structure

% There are some more procedures to view and analyze solid volumee data
```

```
VLFLchecker: 90 vertices and 176 facets.
    0 FACET PROBLEMS DETECTED (ERRORS)
    0 VERTEX PROBLEMS DETECTED (OBSOLETE WARNING)
    0 EDGE PROBLEMS DETECTED (NON MANIFOLD WARNING)
    0 SOLID/EDGE PROBLEMS DETECTED (OPEN SOLID WARNING)
```

- **BBofVL** generates the bounding box dimensions of the solid
- **VLFLui** is simple user interface to open an STL file
- **VLFLminimize** eliminates doubles in VL and FL
- **VLFLnormf** calucates norm vector direction and size
- **VLFLplot4** figure with 4 subplots
- **VLFLselect** selected vertex list for a given facet list
- **VLFLseparate** find different independen objects in VL and FL
- **VLFLshort** remove unused vertices from VL
- **VLFLsurface** returns only vertex list and facet list for one surface
- **VLFLvertexfusion** shrinks vertex list by merging extremy near vertices

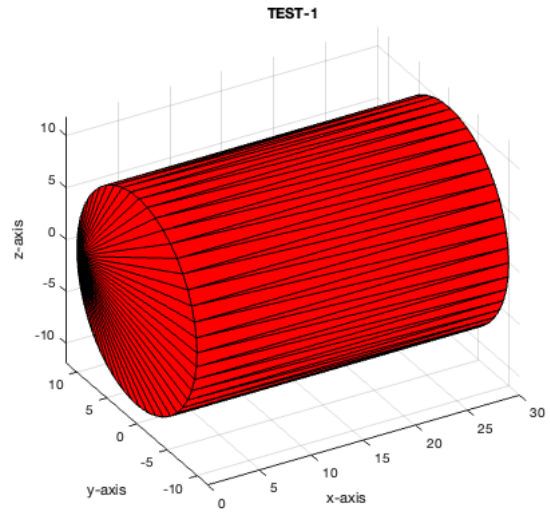
```
BBofVL(VL)
close all; VLFLplots4 (VL,FL,'g');
```

```
ans =
  -10.0000    9.9756   -9.9939    9.9939         0   30.0000
```




```
close all; VLFLseparate(VL,FL);
```

```
Analyzing 90 facets for separation z=[0.0mm|30.0mm]
Object TEST-1 with 176 facets
MVL =
     0   -10     0
```



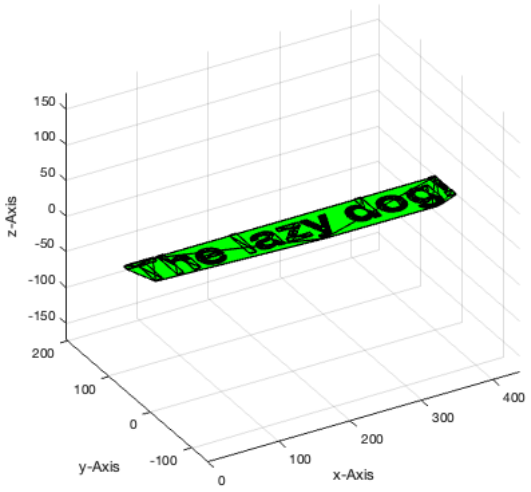
4. Generation of text, numbers, characters and formulas as solid volume

Often you want to write some numbers or code on top of a solid object. For that purpose there is a currently slow function that is able to convert a Matlab-string (even with LaTex-code) into a solid object.

- **VLFLtextimage** writes a line using the text command and converts it into a solid volume
- **VLFLtext** does the same for a very limited number of characters

```
close all;
[VL,FL]=VLFLtextimage('The lazy dog!');
VLFLplot (VL,FL,'g'); view (-30,30);

[VL,FL,d]=VLFLtext('TL-MMXI-XII-XVII');
VLFLwriteSTL (VL,FL,'exp_2011_12_17', 'by Tim C Lueth');
```

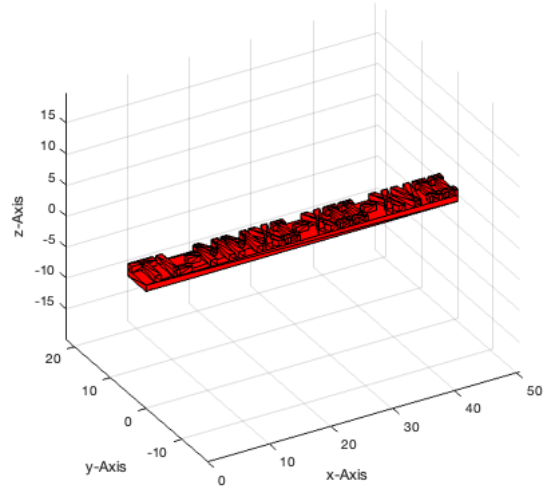
```
WRITING STL FILE /Users/timlueth/Desktop/exp_2011_12_17.STL in ASCII MODE
completed.
```



5. Turning and mirroring of solids by manipulating the vertex lists (VL)

Turning an object and mirroring is quite simple by exchanging a column of the vertex list to change the sign of a column. To show the use of the functions we generate first a simple roman date string as solid volume.

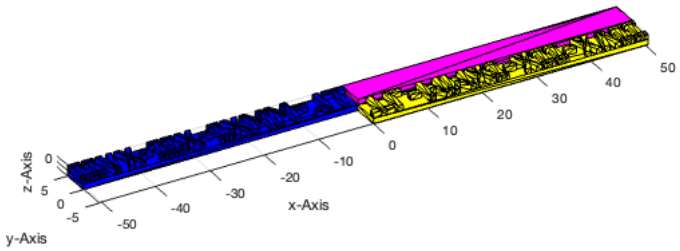
```
close all;
[VL,FL,d]=VLFLtext('TL-MMXI-XII-XVII'); VLFLplot(VL,FL,'r'); view(-30,30);
```



The functions for mirroring solid objects by manipulating the vertex list are the following:

- **VLswapX** mirrors the solid at the x-axis (y/z-plane).
- **VLswapY** mirrors the solid at the y-axis (x/z-plane).
- **VLswapZ** mirrors the solid at the z-axis (x/y-plane).

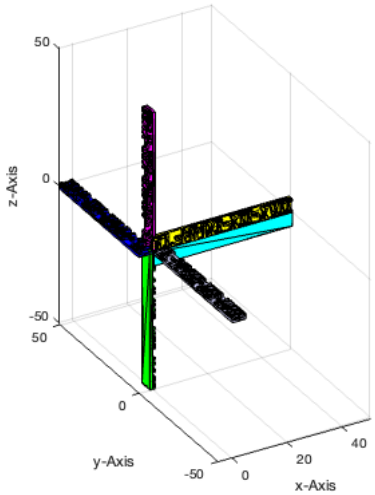
```
close all, view (-30,30); grid on;
VLFLplot(VLswapX(VL),FL,'b');       % mirror at x-axis
VLFLplot(VLswapY(VL),FL,'y');       % mirror at y-axis
VLFLplot(VLswapZ(VL),FL,'m');       % mirror at z-axis
```



The functions for turning solid objects by manipulating the vertex list are the following:

- **VLswapXY** turn the x-axis to the y-axis.
- **VLswapXZ** turn the x-axis to the z-axis.
- **VLswapYX** turn the y-axis to the x-axis.
- **VLswapYZ** turn the y-axis to the z-axis.
- **VLswapZX** turn the z-axis to the x-axis.
- **VLswapZY** turn the z-axis to the y-axis.

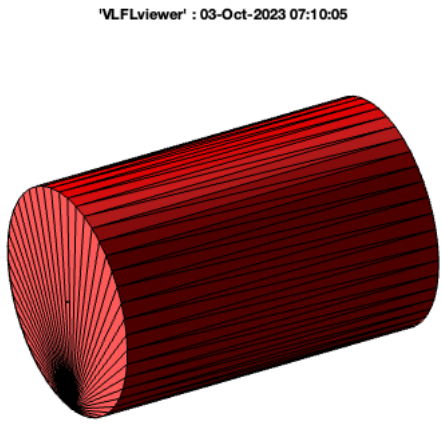
```
close all, view (-30,30); grid on
VLFLplot(VL,FL,'r');                % original solid
VLFLplot(VLswapXY(VL),FL,'b');      % turn the x-axis to the y-axis
VLFLplot(VLswapXZ(VL),FL,'m');      % turn the x-axis to the z-axis
VLFLplot(VLswapYZ(VL),FL,'y');      % turn the y-axis to the z-axis
VLFLplot(VLswapZY(VL),FL,'c');      % turn the z-axis to the y-axis
VLFLplot(VLswapZX(VL),FL,'g');      % turn the z-axis to the x-axis
VLFLplot(VLswapYX(VL),FL,'w');      % turn the y-axis to the x-axis
```



6. Spatial transformation of solids by manipulating the vertex lists (VL)

All solid objects consisting of vertices and facets can be moved and rotated by only manipulating the vertex list (VL). Since the facet list is an index list, the facet list (FL) is not affected by a transformation of the vertex list. The following example generates a cylinder and perform different postion and orientation transformations.

```
closeall;
VLFLviewer([]);
PL=PLcircle(10);                        % define a base-contour
[VL,FL]=VLFLofPLz (PL,30);              % extrude to a solid volume
VL=VLswapZX (VL);                       % swap X and X axis
VLFLplot(VL,FL); view (-30,30); grid on;    % plot as red cylinder
```



In detail, there are five basic transformation functions for manipulation a vertex list (VL)

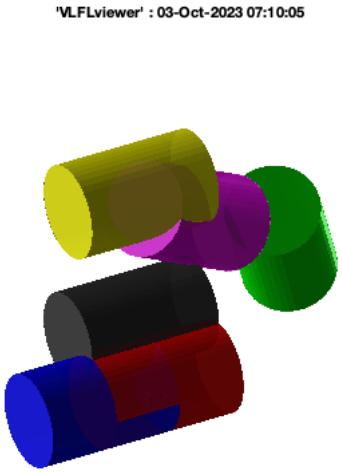
- **VLtrans0** for translating the solid in the coordiante system origin.
- **VLtrans1** for translating the solid into quadrant 1.
- **VLtransP** for translating the solid using a translation vector.
- **VLtransR** for rotation the solid using a rotation matrix.
- **VLtransT** for transformating using an homogenous transformation matrix.

In addition to the already existing matlab functions rotx, roty, and rotz, two new functions are useful.

- **rot** for generating a 3x3 rotation matrix for x y z given in rad.
- **rotdeg** for generating a 3x3 rotation matrix for x y z given in degree.

```
VL=VLtrans0 (VL);                   % Transformation into the origin (blue)
VLFLplot(VL,FL,'b'); view (-30,30);

VL=VLtrans1 (VL);                    % Transformation into quadrant 1 (black)
VLFLplot(VL,FL,'k'); view (-30,30);

VL=VLtransP (VL,[0 ;0; 30]);         % Transformation upwards 30 mm (yellow)
VLFLplot(VL,FL,'y'); view (-30,30);

VL=VLtransR (VL,rotdeg(0,30,15));    % Rotate 30 degree around y and 15 around z (magenta)
VLFLplot(VL,FL,'m'); view (-30,30);

T=[rotdeg(0,30,15), [20;0;0];[0 0 0 1]] % define a homogenous transformation matrix (green)
VL=VLtransT (VL,T);                 % Transformation using an HT matrix
VLFLplot(VL,FL,'g'); view (-30,30); grid on;
VLFLplotlight (1,0.9); grid on;
```

```
T =
    0.8365   -0.2241    0.5000   20.0000
    0.2588    0.9659         0         0
   -0.4830    0.1294    0.8660         0
         0         0         0    1.0000
```



Final remarks on toolbox version and execution date

```
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 07:10:06!
Executed 03-Oct-2023 07:10:08 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
optimization_toolbox
pde_toolbox
phased_array_system_toolbox
signal_blocks
signal_toolbox
simmechanics
simscape
simulink
statistics_toolbox
=================================================================================================
```

- Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2014-11-19
- Tim Lueth, executed and published on 64 Bit PC using Windows with Matlab 2014b on 2014-11-19

Published with MATLAB® R2023a