Pioneering 21st Century
Electromagnetics and Photonics

Module 4 –
MATLAB to CAD

Course Website:
http://emlab.utep.edu/scSVL.htm

EM Lab Website:
http://emlab.utep.edu/

Lecture Outline

- STL files
- Reading and writing STL files from MATLAB
- Generating faces and vertices using MATLAB
- Converting images and 2D objects to STL

STL Files

What is an STL File?

STL – Standard Tessellation Language

STL does not mean stereolithography file.

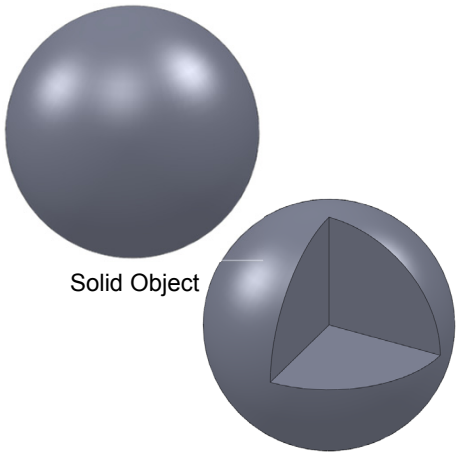
This file format is widely used in rapid prototyping. It contains only a triangular mesh of an objects surface. Color, texture, materials, and other attributes are not represented.

They can be text files or binary. Binary is more common because they are more compact. We will look at text files because that is more easily interfaced with MATLAB.

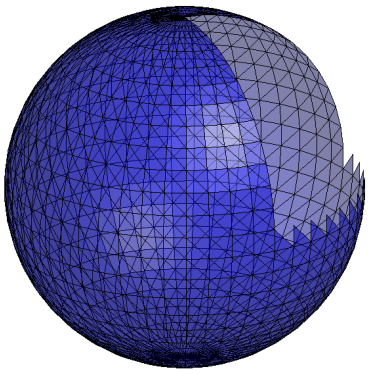
Surface Mesh

Short Course

Despite this sphere really being a solid object, it is represented in an STL file by just its surface.



Solid Object



STL Representation

STL File Format

Short Course

solid *name*

•••

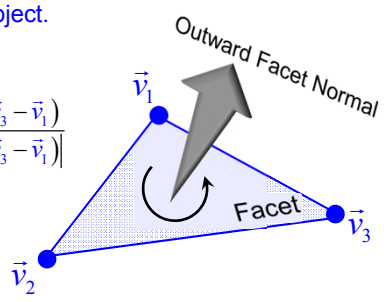
facet normal $n_x\ n_y\ n_z$
  outer loop
    vertex $v_{x,1}\ v_{y,1}\ v_{z,1}$
    vertex $v_{x,2}\ v_{y,2}\ v_{z,2}$
    vertex $v_{x,3}\ v_{y,3}\ v_{z,3}$
  endloop
endfacet

•••

endsolid *name*

This set of text is repeated for every triangle on the surface of the object.

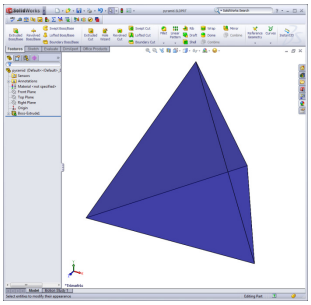
$$\hat{n} = \frac{(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)}{\left|(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)\right|}$$



Bold face indicates a keyword; these must appear in lower case. Note that there is a space in "facet normal" and in "outer loop," while there is no space in any of the keywords beginning with "end." Indentation must be with spaces; tabs are not allowed. The notation, "{…}+," means that the contents of the brace brackets can be repeated one or more times. Symbols in italics are variables which are to be replaced with user-specified values. The numerical data in the **facet normal** and **vertex** lines are single precision floats, for example, 1.23456E+789. A **facet normal** coordinate may have a leading minus sign; a **vertex** coordinate may not.

Example STL File

```
solid pyramid
   facet normal -8.281842e-001 2.923717e-001 -4.781524e-001
      outer loop
         vertex 4.323172e-018 1.632799e-017 6.495190e-001
         vertex 3.750000e-001 7.081604e-001 4.330127e-001
         vertex 3.750000e-001 0.000000e+000 0.000000e+000
      endloop
   endfacet
   facet normal 0.000000e+000 2.923717e-001 9.563048e-001
      outer loop
         vertex 7.500000e-001 0.000000e+000 6.495190e-001
         vertex 3.750000e-001 7.081604e-001 4.330127e-001
         vertex 0.000000e+000 0.000000e+000 6.495190e-001
      endloop
   endfacet
   facet normal 8.281842e-001 2.923717e-001 -4.781524e-001
      outer loop
         vertex 3.750000e-001 -1.632799e-017 0.000000e+000
         vertex 3.750000e-001 7.081604e-001 4.330127e-001
         vertex 7.500000e-001 0.000000e+000 6.495190e-001
      endloop
   endfacet
   facet normal 0.000000e+000 -1.000000e+000 0.000000e+000
      outer loop
         vertex 3.750000e-001 0.000000e+000 0.000000e+000
         vertex 7.500000e-001 0.000000e+000 6.495190e-001
         vertex 0.000000e+000 0.000000e+000 6.495190e-001
      endloop
   endfacet
endsolid pyramid
```



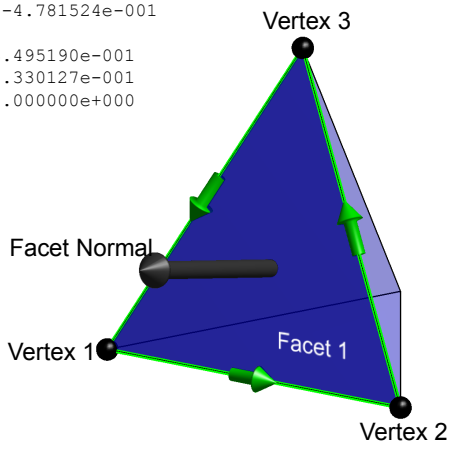
An STL file is essentially just a list of all the triangles on the surface of an object.

Each triangle is defined with a surface normal and the position of the three vertices.

A Single Triangle

```
facet normal -8.281842e-001 2.923717e-001 -4.781524e-001
   outer loop
      vertex 4.323172e-018 1.632799e-017 6.495190e-001
      vertex 3.750000e-001 7.081604e-001 4.330127e-001
      vertex 3.750000e-001 0.000000e+000 0.000000e+000
   endloop
endfacet
```

1. Vertices must be given in the order which follows the right-hand rule.
2. Facet normal must follow right-hand rule and point outward from object.
   a) Some programs set this to [0;0;0] or convey shading information.
   b) Don't depend on it*!*
3. Adjacent triangles must have two common vertices.
4. STL files appear to be setup to handle arbitrary polygons. Don't do this.

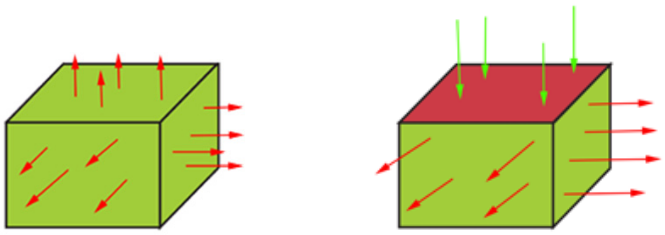


Warnings About Facet Normals

- Since the facet normal can be calculated from the vertices using the right-hand rule, sometimes the facet normal in the STL file contains other information like shading, color, etc.
- Don't depend on the right-hand rule being followed.
- Basically, don't depend on anything!

STL Problems:
Inverted Normals

All surface normals should point outwards.



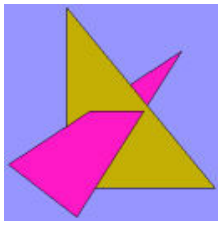
Good Bad

http://admproductdesign.com/workshop/3d-printing/definition-of-stl-errors.html

STL Problems:
Intersecting Triangles

Short Course

No faces should cut through each other. Intersections should be removed.

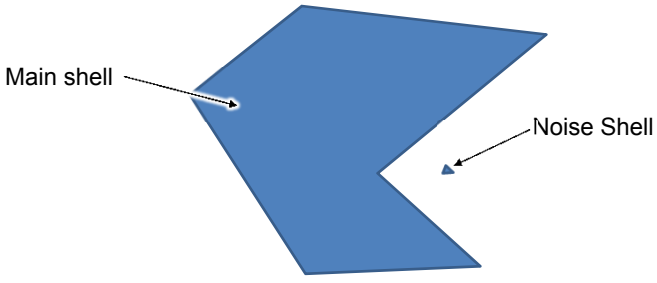


http://admproductdesign.com/workshop/3d-printing/definition-of-stl-errors.html

STL Problems:
Noise Shells

Short Course

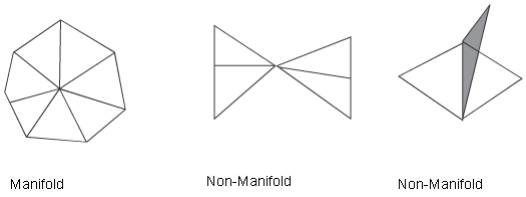
A shell is a collection of triangles that form a single independent object. Some STL files may contain small shells that are just noise. These should be eliminated.



STL Problems:
Nonmanifold Mesh

Short Course

A manifold (i.e. watertight) mesh has no holes and is described by a single continuous surface.



http://http://www.autodesk.com/

Mesh Repair Software

Short Course

- Commercial Software
  - Magics
  - NetFabb
  - SpaceClaim
  - Autodesk
- Open Source Alternatives
  - MeshLab
  - NetFabb Basic
  - Blender
  - Microsoft Model Repair

Reading/Writing STL Files in MATLAB

How to Store a Surface Mesh in MATLAB

We have N facets and $\leq 3N$ vertices to store in arrays.

`F(`N`,3)` Array of triangle facets
`V(?,3)` Array of triangle vertices

Many times, the number of vertices is exactly $3N$. Observing that many of the triangle facets share vertices, there will be redundant vertices.

STL files can be compressed to eliminate redundant vertices, but many times they are not.

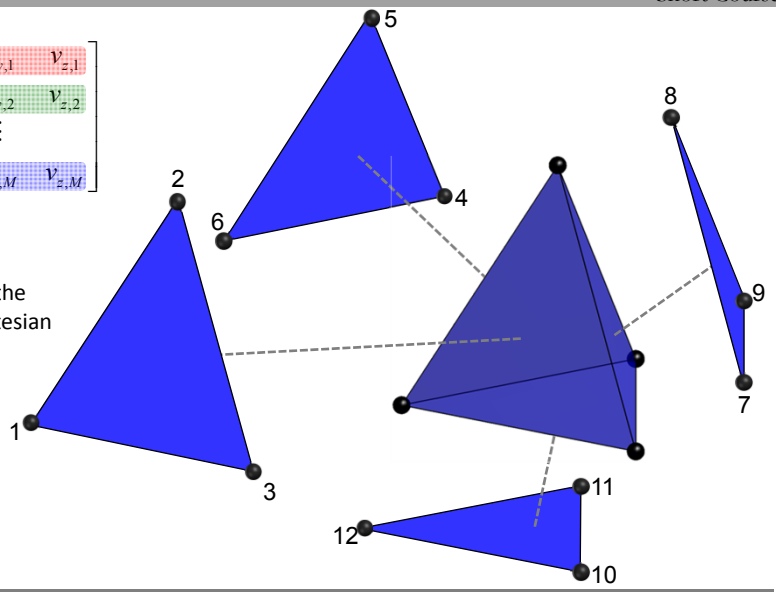
Lazy Array of Vertices (1 of 2)

Short Course

$$V = \begin{bmatrix} v_{x,1} & v_{y,1} & v_{z,1} \\ v_{x,2} & v_{y,2} & v_{z,2} \\ \vdots & & \\ v_{x,M} & v_{y,M} & v_{z,M} \end{bmatrix}$$

V is an array containing the position of all the vertices in Cartesian coordinates.

M is the total number of vertices.



Lazy Array of Vertices (2 of 2)

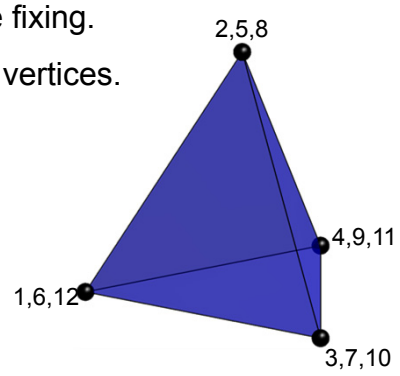
Short Course

There is redundancy here. Twelve vertices are stored, but the device is really only composed of four.

While an inefficient representation, this is probably not worth your time fixing.

SolidWorks exports a lazy array of vertices.

$$V = \begin{bmatrix} v_{x,1} & v_{y,1} & v_{z,1} \\ v_{x,2} & v_{y,2} & v_{z,2} \\ \vdots & & \\ v_{x,M} & v_{y,M} & v_{z,M} \end{bmatrix}$$

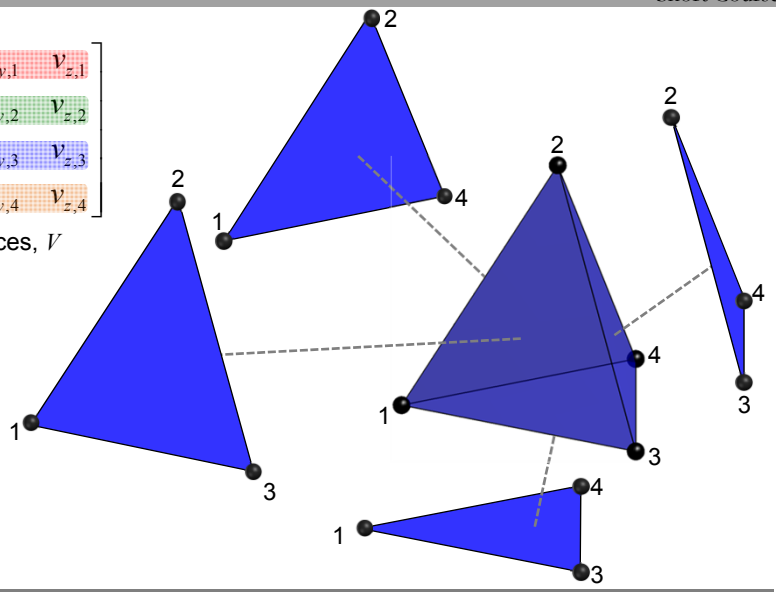


Compact Array of Vertices

Short Course

$$V = \begin{bmatrix} v_{x,1} & v_{y,1} & v_{z,1} \\ v_{x,2} & v_{y,2} & v_{z,2} \\ v_{x,3} & v_{y,3} & v_{z,3} \\ v_{x,4} & v_{y,4} & v_{z,4} \end{bmatrix}$$

Array of Vertices, V



Array of Faces

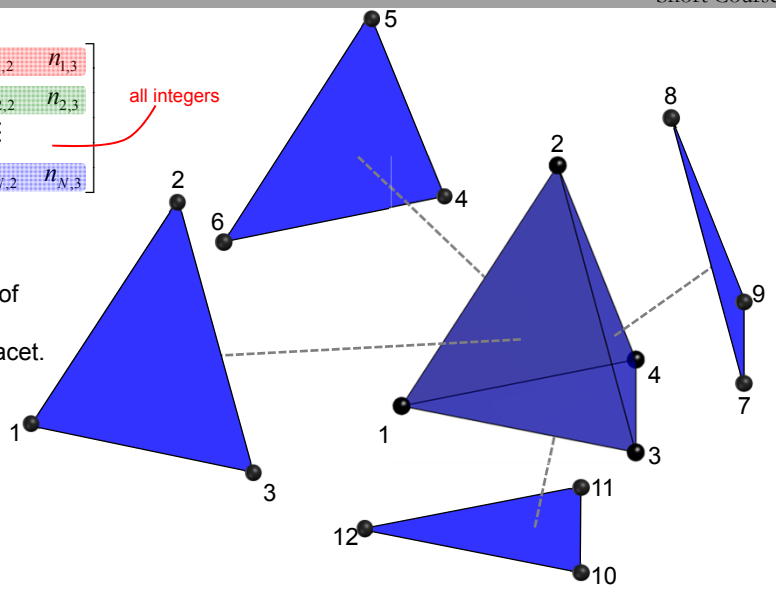
Short Course

$$F = \begin{bmatrix} n_{1,1} & n_{1,2} & n_{1,3} \\ n_{2,1} & n_{2,2} & n_{2,3} \\ \vdots & & \\ n_{N,1} & n_{N,2} & n_{N,3} \end{bmatrix}$$

all integers

F is an array indicating the array indices of the vertices defining the facet.

N is the total number of faces.



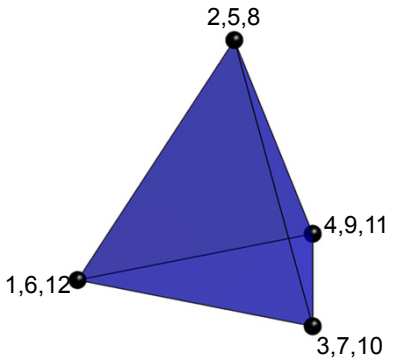
Example of Lazy Arrays

```
F =

     1     2     3
     4     5     6
     7     8     9
    10    11    12


V =

    0.0000    0.6495    0.0000
    0.3750    0.4330    0.7082
    0.3750         0         0
    0.7500    0.6495         0
    0.3750    0.4330    0.7082
         0    0.6495         0
    0.3750         0   -0.0000
    0.3750    0.4330    0.7082
    0.7500    0.6495         0
    0.3750         0         0
    0.7500    0.6495         0
         0    0.6495         0
```



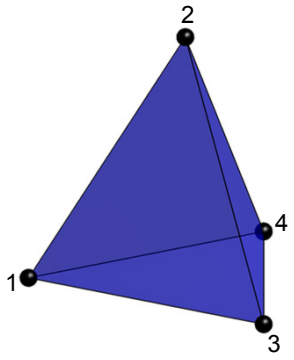
Example of Compact Arrays

```
F =

     1     2     3
     4     2     1
     3     2     4
     3     4     1


V =

    0.0000    0.6495    0.0000
    0.3750    0.4330    0.7082
    0.3750         0         0
    0.7500    0.6495         0
```



This can make a very large difference for large and complex objects.

How to Read an STL File Into MATLAB

1. Open file as read-only.
2. Read first line and store object name if needed.
3. Read facet data
   a. Read the **facet normal**. You probably don't need this, and you can ignore it.
   b. Skip **outer loop** line
   c. Read **vertex** 1
   d. Read **vertex** 2
   e. Read **vertex** 3
   f. Skip **endloop** line
   g. Skip **endfacet** line
   h. Add vertices to *V* array.
   i. Add facet to *F* array.
4. If another facet exists, go back to Step 3.
5. Skip **endsolid** line
6. Close the file.

The MathWorks website has several good STL readers and other tools.

How to Write an STL File From MATLAB

1. Open a new text file.
2. Add first line, **solid** *name*.
3. For each triangle facet, repeat the following steps
   a. Determine the vertices of the facet $\vec{v}_1, \vec{v}_2, \vec{v}_3$
   b. Calculate the facet normal (n_x, n_y, n_z) using the right-hand rule.
   c. Write line **facet normal** $n_x\ n_y\ n_z$
   d. Write line **outer loop**
   e. Write **vertex** $v_{x,1}\ v_{y,1}\ v_{z,1}$
   f. Write **vertex** $v_{x,2}\ v_{y,2}\ v_{z,2}$
   g. Write **vertex** $v_{x,3}\ v_{y,3}\ v_{z,3}$
   h. Write line **endloop**
   i. Write line **endfacet**
4. If another facet exists, go back to Step 3.
5. Write line **endsolid** *name*
6. Close the file.

See next slide.

Be sure to put these in the order that follows the right-hand rule!

For this Short Course, we will use the supplied `svlcad()` function, which creates a binary STL file.

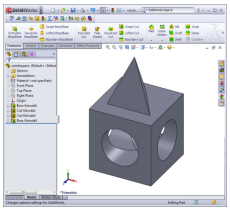
STL Files Generated by SolidWorks

For some reason, SolidWorks does not use the z-axis as the vertical axis.

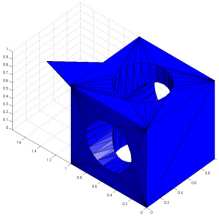
For convenience, STL files can be easily reoriented.

```
% REORIENT SOLIDWORKS AXES TO MATLAB AXES
Y       = V(:,2);
V(:,2)  = V(:,3);
V(:,3)  = Y;
```

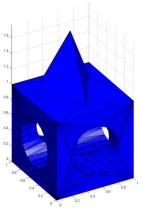
SolidWorks Axes



Orientation in SolidWorks



Imported Orientation



Adjusted Orientation

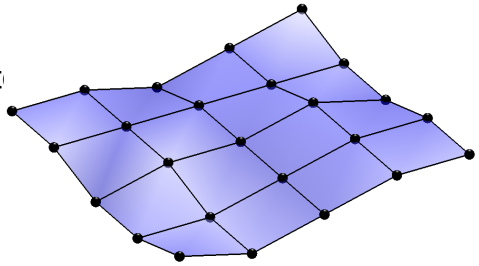
Generating Faces and Vertices Using MATLAB

MATLAB Surfaces

Short Course

Surfaces composed of square facets are stored in `X`, `Y`, and `Z` arrays.

The surface shown is construct
of arrays that are all 5×5.



```
X =

   -1.0000   -0.6000   -0.2000    0.2000    0.6000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000
```

```
Y =

   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -0.6000   -0.6000   -0.6000   -0.6000   -0.6000
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
    0.2000    0.2000    0.2000    0.2000    0.2000
    0.6000    0.6000    0.6000    0.6000    0.6000
```

```
Z =

    0.1000         0    0.1000    0.2000    0.2000
         0         0    0.1000    0.2000    0.2000
         0    0.1000    0.1000    0.2000    0.1000
    0.1000    0.1000    0.1000    0.1000    0.1000
    0.1000    0.1000         0    0.1000    0.2000
```


Direct Construction of the Surface Mesh

Short Course

MATLAB has a number of built-in commands for generating surfaces. Some of these are `cylinder()`, `sphere()` and `ellipsoid()`.

```
% CREATE A UNIT SPHERE
[X,Y,Z] = sphere(41);
```

Surfaces can be converted to triangular patches (facets and vertices) using the `surf2patch()` function.

```
% CONVERT TO PATCH
[F,V] = surf2patch(X,Y,Z,'triangles');
```

The faces and vertices can be directly visualized using the `patch()` function.

```
% VISUALIZE FACES AND VERTICES
h = patch('faces',F,'vertices',V);
set(h,'FaceColor',[0.5 0.5 0.8],'EdgeColor','k');
```

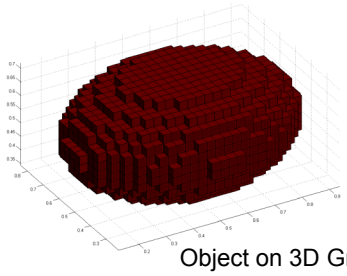
Grid → Surface Mesh

Short Course

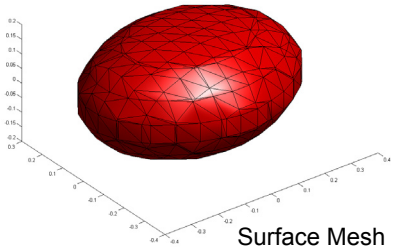
3D objects on a grid can be converted to a surface mesh using the command `isosurface()`.

```
% CREATE ELLIPTICAL OBJECT
OBJ = (X/rx).^2 + (Y/ry).^2 + (Z/rz).^2;
OBJ = (OBJ < 1);

% CREATE SURFACE MESH
[F,V] = isosurface(X,Y,Z,OBJ,0.5);
```



Object on 3D Grid



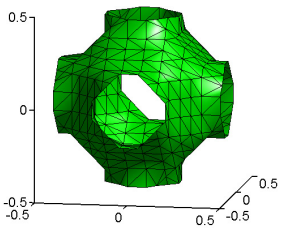
Surface Mesh

May Need `isocaps()`

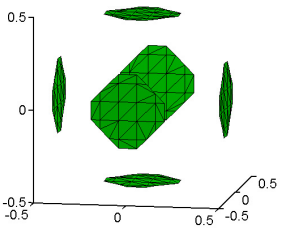
Short Course

When 3D objects extend to the edge of the grid, you may need to use `isocaps()`.

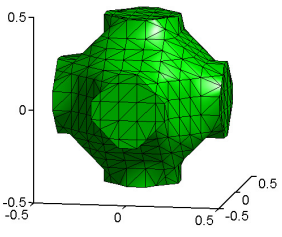
`isosurface()`



`isocaps()`



`isosurface()`
`+ isocaps()`



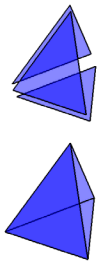

```
% CREATE SURFACE MESH
[F,V]   = isosurface(X,Y,Z,OBJ,0.5);
[F2,V2] = isocaps(X,Y,Z,OBJ,0.5);
```

Combining Faces and Vertices from Two Objects

SVL Short Course

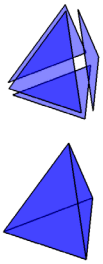
There are no functions in MATLAB to perform Boolean operations on multiple meshes. We can, however, combine the faces and vertices from two objects. Be careful this does not result in overlaps or gaps between objects.

F1 and V1

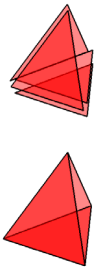


F2 and V2

Correctly Combined



Incorrectly Combined

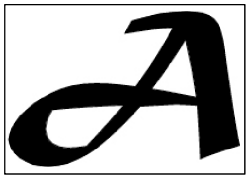


```
% COMBINE FACES AND VERTICES
F3 = [ F1 ; F2+length(V1(:,1)) ]
V3 = [ V1 ; V2 ]
```

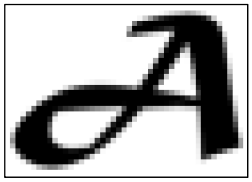
```
% WRONG WAY TO
% COMBINE FACES AND VERTICES
F3 = [ F1 ; F2 ]
V3 = [ V1 ; V2 ]
```


Converting Images and 2D Objects to STL

Load and Resize the Image



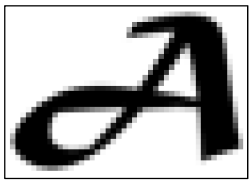
```
% LOAD IMAGE
B = imread('letter.jpg');
```



```
% RESIZE IMAGE
B = imresize(B,0.2);
[Nx,Ny,Nc] = size(B);
```

This will give us a coarser mesh to be faster and more memory efficient.

Flatten the Image

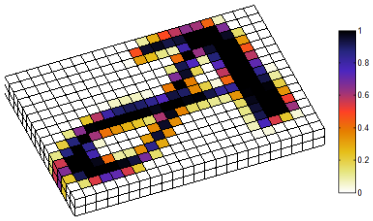


Images loaded from file usually contain RGB information making them 3D arrays. These arrays must be converted to flat 2D arrays before meshing.


```
% FLATTEN COLOR IMAGE
B = double(B);
B = B(:,:,1) + B(:,:,2) + B(:,:,3);
B = 1 - B/765;
```

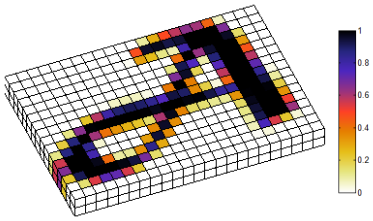
Stack the Image

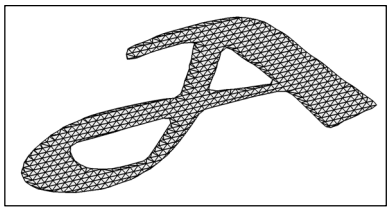
We will mesh the image using `isocaps()`, but that function requires a 3D array. So, we will stack this image to be two layers thick.



```
% STACK IMAGE
B(:,:,2) = B;
```


Mesh the Image Using `isocaps()`





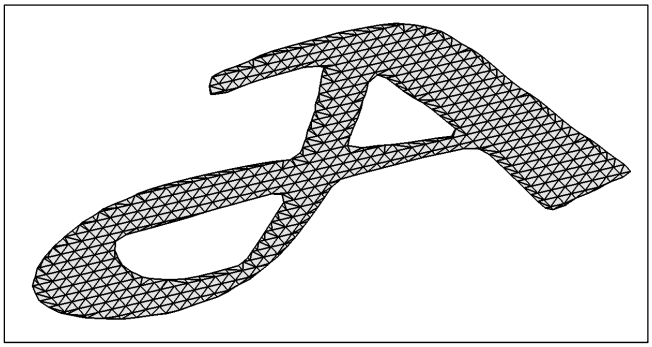
We only wish to mesh a single surface so we give `isocaps()`, the additional parameter '`zmin`' to do this.

```
% CREATE 2D MESH
[F,V] = isocaps(ya,xa,[0 1],B,0.5,'zmin');
```

Save the Mesh as an STL File

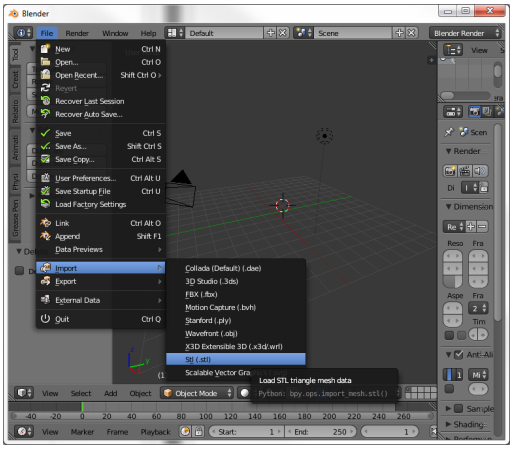
We can save this as an STL file.

```
% SAVE AS A STL FILE
svlcad('letter.stl',F,V);
```



Extrude Using Blender (1 of 2)

1. Open Blender.exe.
2. File → Import → Stl (.stl)
3. Open the STL file you just created.



Extrude Using Blender (2 of 2)

Short Course

1. Select the object with right mouse click.
2. Press TAB to enter Edit mode.
3. Press ‘e’ to extrude mesh.
4. Press TAB again to exit Edit mode.
5. You can now edit your object or export as a new 3D STL file.