CSC293 CS Improves 3D Printed Manufacturing
CAD and Graphics Background

Sreepathi Pai

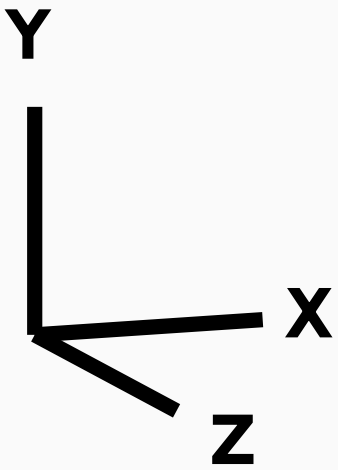
Oct 4, 2023

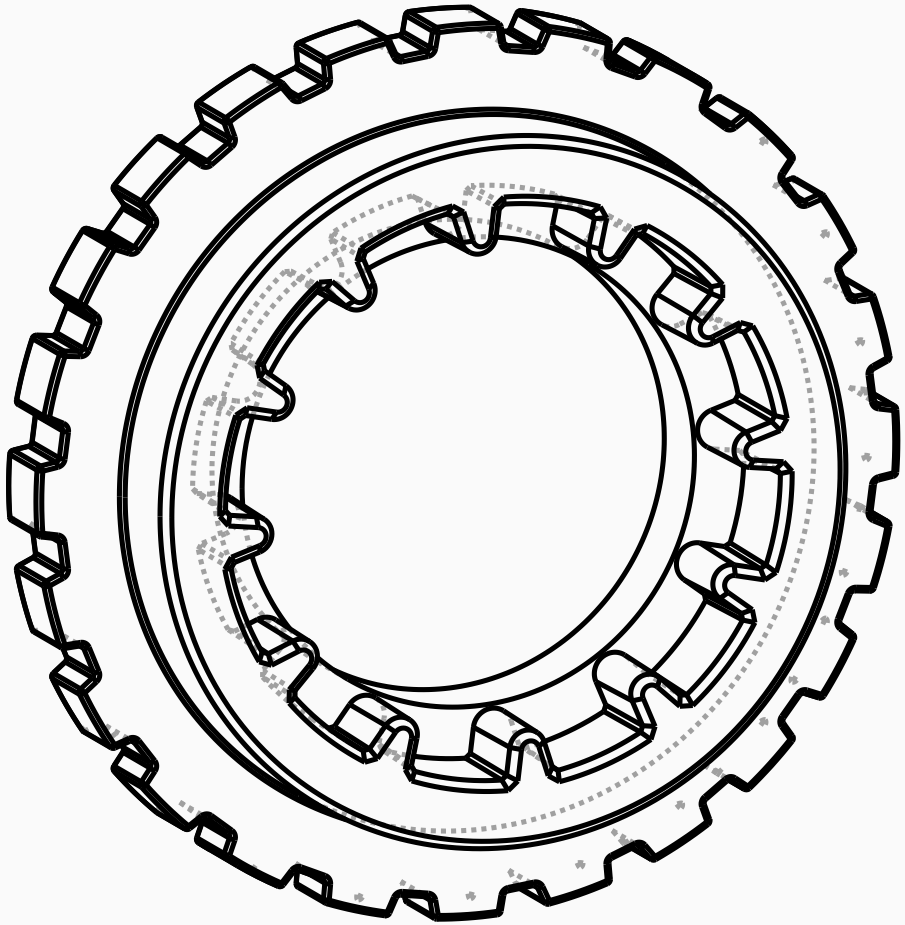
URCS

Outline

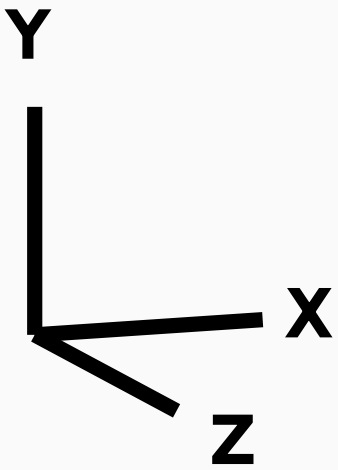
Outline

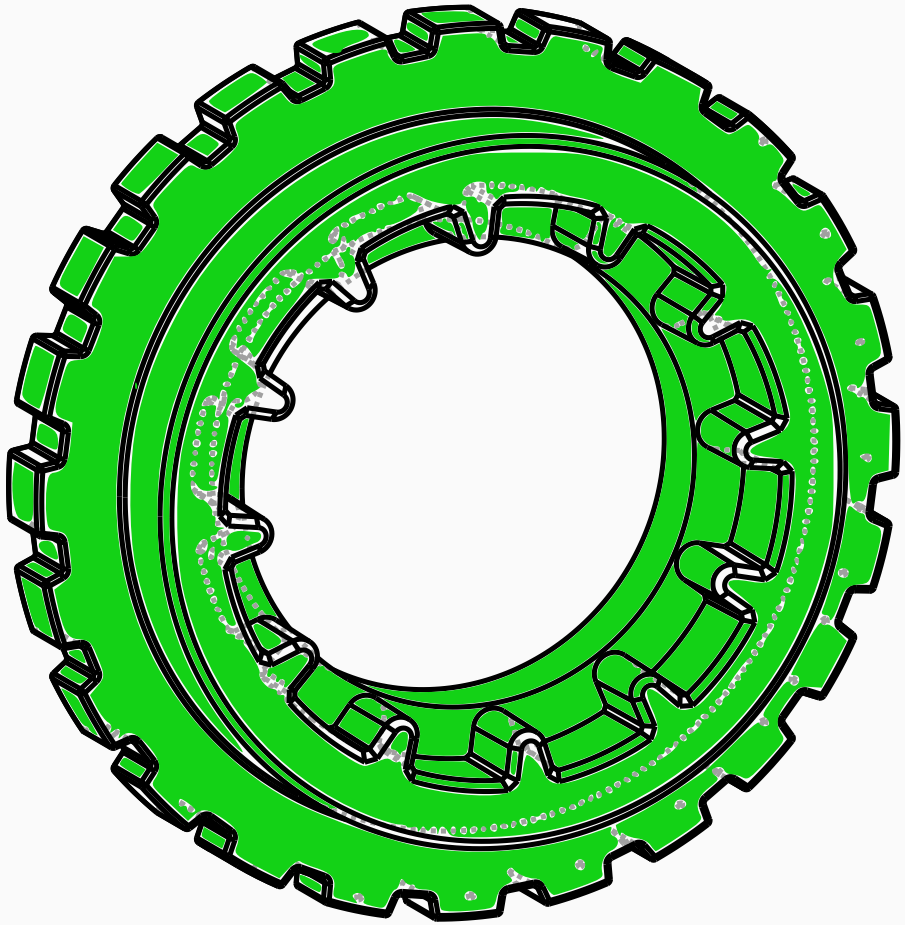
Generating a Wireframe





Shading a 3D part





Outline

SVG

- Scalable *Vector* Graphics
- W3C Standard, Embeddable in HTML
- Supported by most browsers
- 2D
- Mozilla has a great tutorial
- CadQuery can export wireframe to SVG
    - but has no way to specify shading

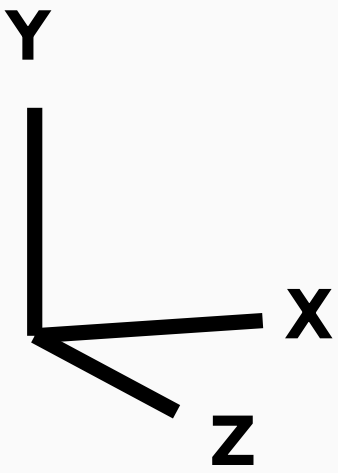
Examining the Generated SVG

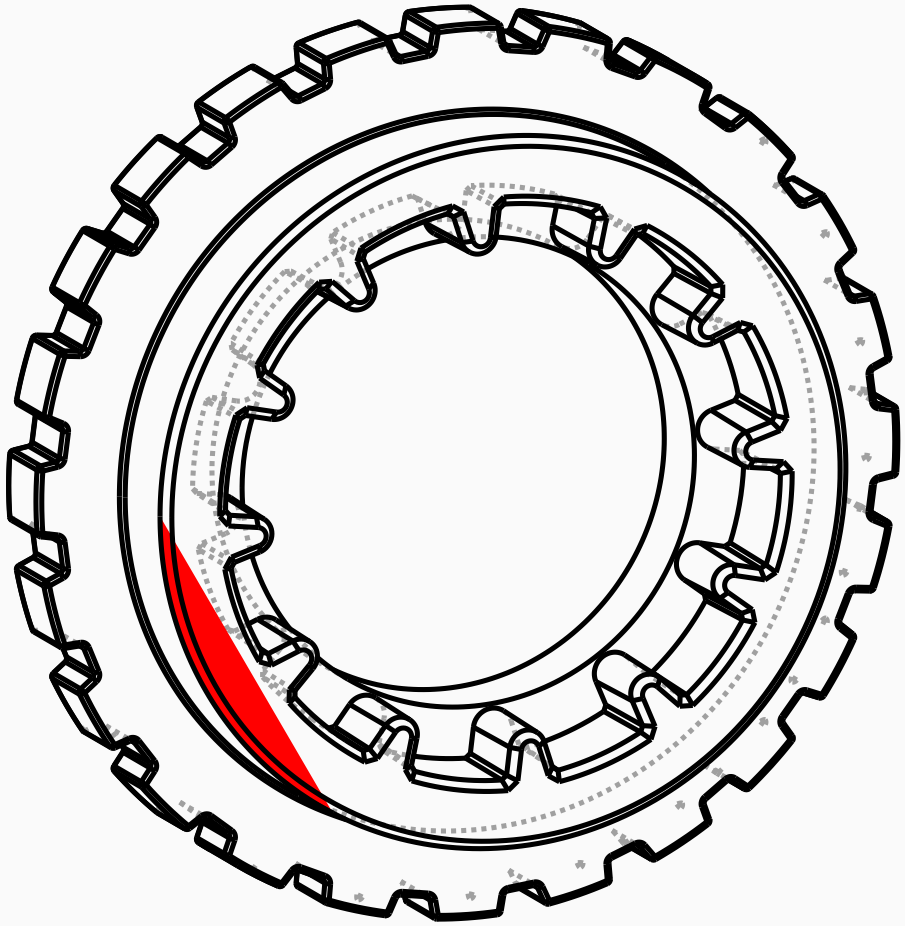
```
 ...
<g transform="scale(9.010167114828722, -9.010167114828722)
   translate(32.26901019751 4014,-61.3560295213 0987)" stroke-width="
   fill="none">
 ...
       <!-- solid lines -->
       <g  stroke="rgb(0,0,0)" fill="none">
            <path d="M-6.753157558956 9215,47.979713997476026 L-6.75
  L-6.749099056728342,47.723866214426415 L-6.744026956694709,47.596
L-6.736927934841 8125,47.46887244813687 L-6.727803908742059,47.34178
L-6.716657342957623,47.21500819345776 L-6.7034912483747

    ...
```

Manual fill?

```
<path fill="rgb(255,0,0)" ...
```



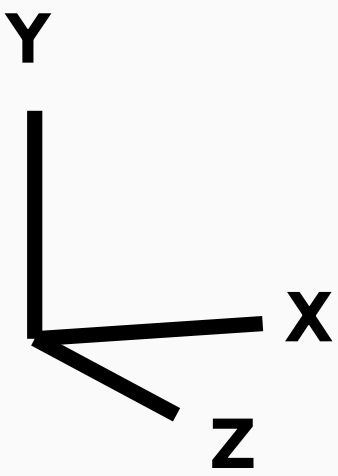


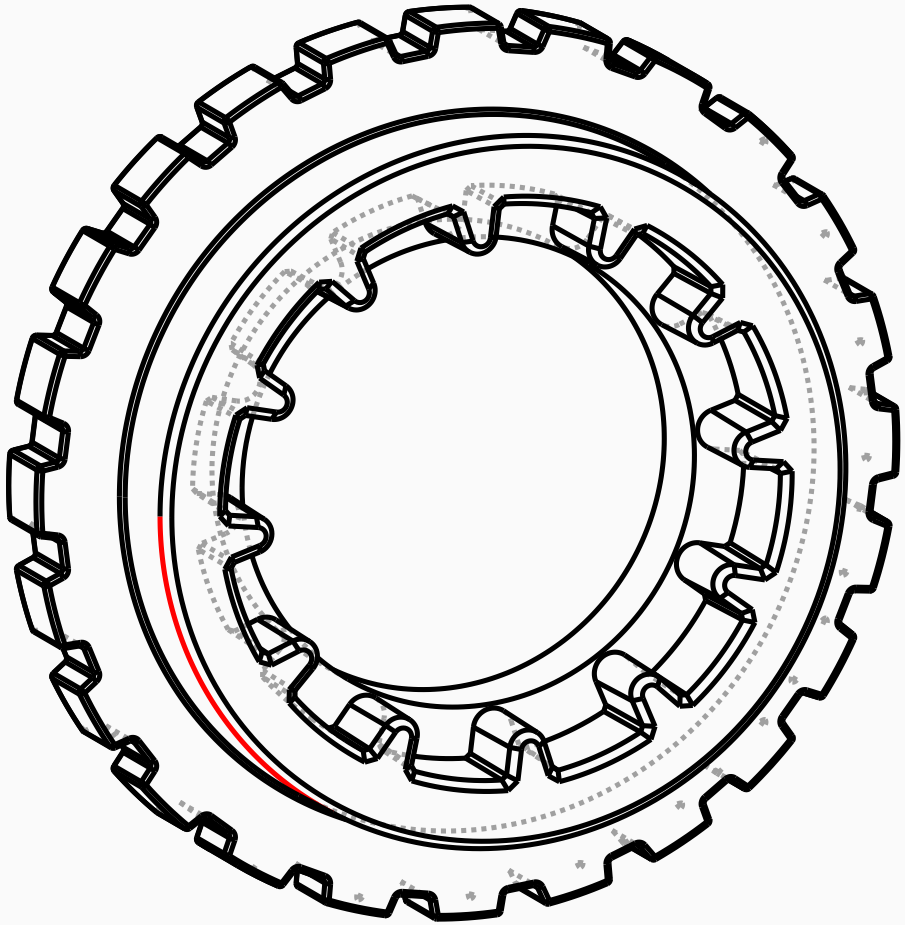
Fills

- A fill must be a closed path
  - if the path is not closed, the last point is connected to the first
- There are rules for determining what is the inside and outside of a path
  - relevant when a path intersects itself

Highlighting the path

```
<path stroke="rgb(255,0,0)" ...
```

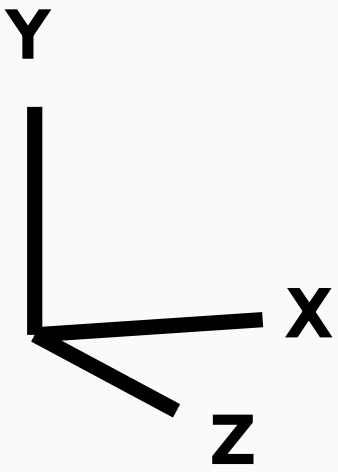


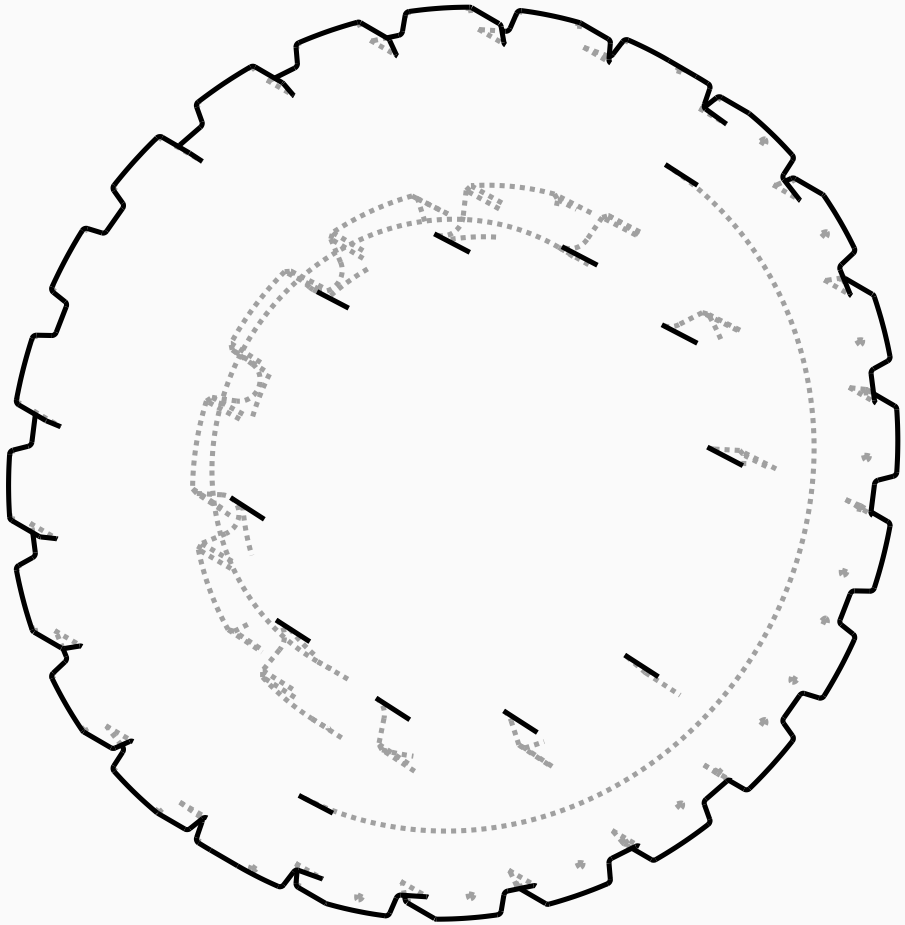


The SVG Export Code

- Link to CADQuery SVG Export Code
- OpenCascade Documentation for HLR

Outline edges?





Outline

Geometry Representations

- STEP files contain (among other things) geometry represented using *boundary representations*
  - B-reps for short
  - More details in the OpenCascade documentation
- These are made up of abstract parametric objects (lines, points, circles, etc.)
- These need to be discretized to be rendered on screen
  - This is what `makeSVGEdge` in the SVG export code is doing
- The HLR algorithms:
  - Extracts visible edges
  - Make shapes 2D (essentially, rendering)

Reframing the Problem

- Can't we just render the 3D object?
- CADQuery (and OpenCascade) have a visualization API
  - It's used in CQ-editor
- Unfortunately, produces raster images only
  - There used to be a vector output using `gl2ps`, but it was removed a few versions ago
- Also, unclear how to produce the images non-interactively
  - Should be possible, though
- Would be ideal if we had a 3D renderer that produced SVG output

Outline

Converting to a mesh and rendering it

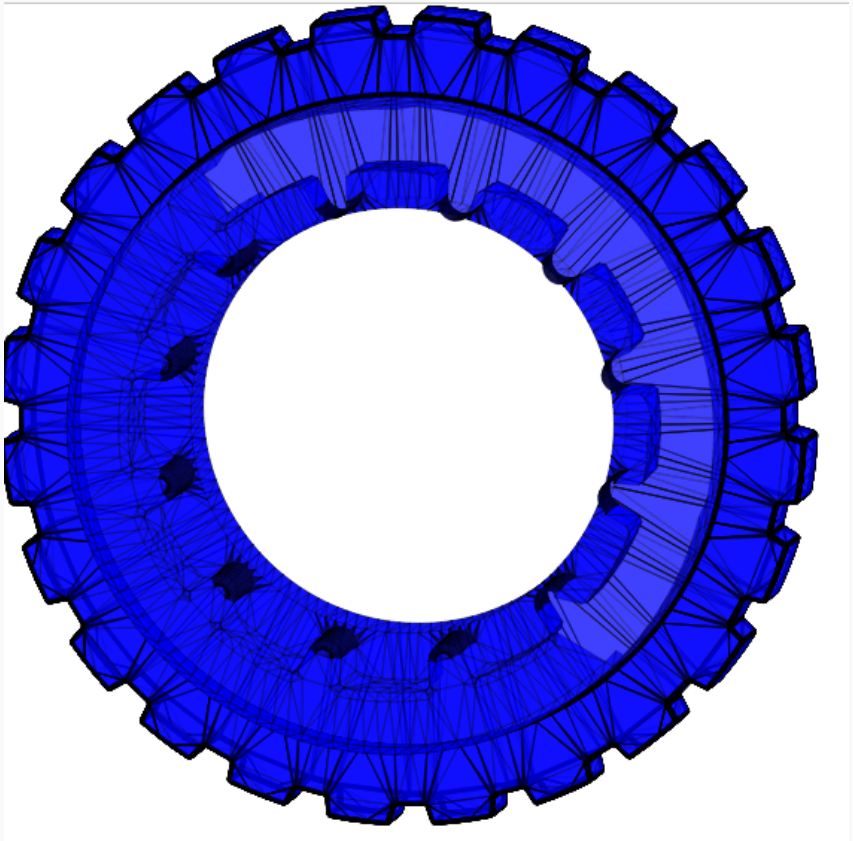
- Most 3D renderers work with meshes, not B-reps
  - Meshes are made of triangles and sometimes quads.
- Once a mesh is obtained
  - svg3d can be used to render?
  - Even matplotlib, maybe?

Obtaining a mesh

- OpenCascade can produce meshes
  - Used for STL export, for example
- Calling the `tesselate` method works
  - See the ThreeMF exporter for detailed example
- Decided to export it to `TJS`, or ThreeJS format
  - Outputs a list of vertices (points) and triangles

Rendering the Focus Wheel

- TJS file is around 16MB
  - 480K vertices
  - 1.1M triangles
- SVG file is around 19MB
  - takes around 24 seconds to produce on my laptop
  - around 2 minutes to render!
- PDF conversion (from SVG) takes 1minute
  - rendering takes around the same time



Issues

- Too slow!
  - Recall the video
- Still need to figure out camera placement
  - SVG3D uses OpenGL camera (documentation)
- Task is "accomplished"
  - But how can we improve?

Discussion

Possible Solutions

- Discretize the objects in 3D space to obtain edges as paths
- Post-process the mesh to extract edges
    - Internal triangles vs External triangles