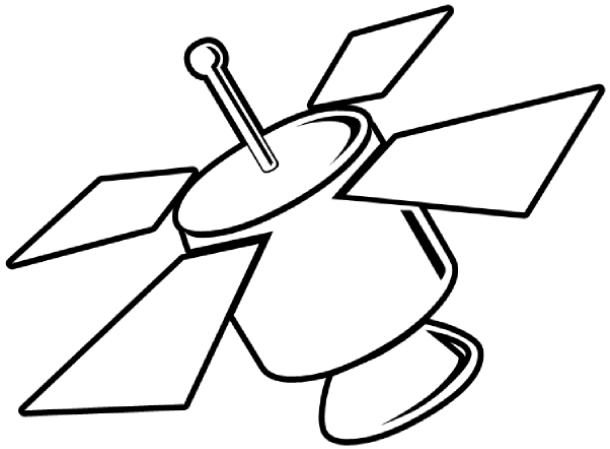
# Data Reduction for Modelling Satellite Radar Cross Sections

**Daniel Topa**
**daniel.topa@hii.com**

**Huntington Ingalls Industries**
**Mission Technologies**

**December 19, 2024**

# Models of Radar Cross Sections for Satellites



## Spherical Harmonics Expansion

$$f(r,\theta,\phi) \approx a_{0,0}Y_0^0 + a_{1,-1}Y_1^{-1} + a_{1,0}Y_1^0 + a_{1,1}Y_1^1 + \dots$$

**where**

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(2n+1)(n-m)!}{4\pi(n+m)!}}P_n^m(\cos\theta)e^{im\phi}$$

# Overview

# Input and Final Output

## Input: *.obj File

```
# Created with the Wolfram Language : www.wolfram.com

mtllib sp-006.mtl

# 6 vertex positions
v  0 0 -1
v  0 -1 0
v  -1 0 0
```

## Output: Amplitude Vector

$$a_{0,0} = 1.345 \pm 0.015$$
$$a_{1,-1} = 1.098 \pm 0.017$$
$$a_{1,0} = 1.210 \pm 0.017$$
$$a_{1,1} = 0.945 \pm 0.017$$
$$a_{2,-2} = 0.512 \pm 0.018$$
$$a_{2,-1} = 0.732 \pm 0.017$$
$$a_{2,0} = 1.110 \pm 0.017$$
$$a_{2,1} = 0.885 \pm 0.016$$
$$a_{2,2} = 0.658 \pm 0.017$$

# Big Picture: CAD to RCS Table

**We discuss Step 1 $\Rightarrow$ Step 2**

1. **Start with CAD model:** `*.stl`
2. **Finish with table** `*.rcs`
3. **Resolved to approximate** $f(\theta, \phi)$

# Software Components

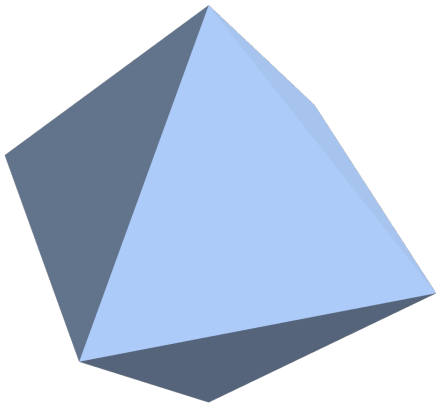
1. **converter:** `*.obj` ⇒ `*.facet`
2. **mesh analysis & repair:** `*.obj` ⇒ `*.facet`
3. **extractor: pull backscatter from** `*.4112.txt`
4. **converter: backscatter to** `*.rcs`
5. **calculator:** `*.rcs` **to spherical harmonic amplitudes**

# CAD file (*.stl) to Mesh Structure File (*.obj)

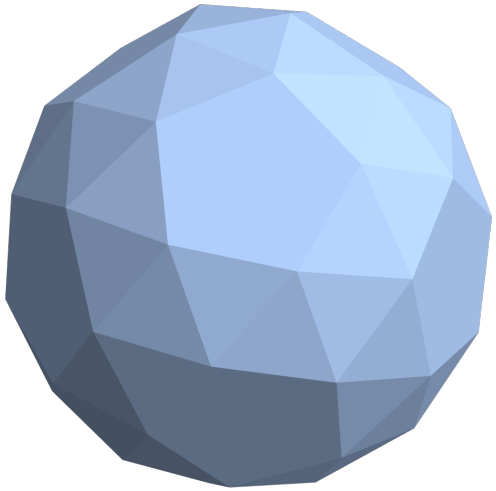
**Many Tools For Converting *.stl to *.obj**

1. **Blender**
2. **FreeCAD**
3. **OpenSCAD**
4. **SolidWorks**
5. **Tinkercad**
6. **MeshConvert.com**
7. **Online 3D Model Converter**
8. **others**

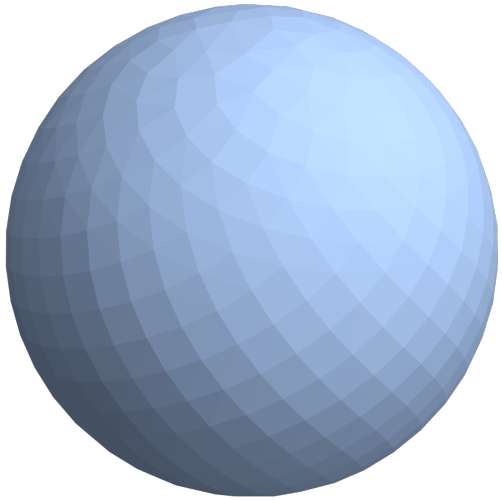
# Seeing the *.obj File



6 vertices



60 vertices



600 vertices

**Decadal Improvement in Resolution:**
**Number of vertices increases** $\times 10$

# sp-006.obj

```
1  # Created with the
       Wolfram Language
       : www.wolfram.com
2
3  mtllib sp-006.mtl
4
5  # 6 vertex positions
6  v  0 0 -1
7  v  0 -1 0
8  v  -1 0 0
9  v  1 0 0
10 v  0 0 1
11 v  0 1 0
12
13 # 0 UV coordinates
14
15 # 0 vertex normals
```

```
17 # Mesh '' with 8
       faces
18 usemtl Material_1
19 f  1/ 2/ 3/
20 f  2/ 1/ 4/
21 f  2/ 5/ 3/
22 f  5/ 2/ 4/
23 f  1/ 6/ 4/
24 f  6/ 1/ 3/
25 f  6/ 5/ 4/
26 f  5/ 6/ 3/
```

# Components of the *.obj

1. **Headers and Comments (#):**
   - **Used for metadata or human-readable information.**
   - **Example:** `# Created with Wolfram Language.`
2. **Vertex Positions (v):**
   - **Specifies 3D coordinates for vertices.**
   - **Example:** `v 0 0 -1.`
3. **Faces (f):**
   - **Defines polygons by referencing vertex indices.**
   - **Example:** `f 1/2/3.`

# Components of the *.obj

4. **Material Library Reference (mtllib):**
   - **External *.mtl file that specifies visual materials for rendering (e.g., color, shading)**
   - **Example: sp-006.mtl.**
   - **Important Note: This *.mtl file is not related to the electromagnetic materials library in CAD models, which defines physical properties like permittivity, permeability, or conductivity.**

# Python Tool for *.obj to *.facet I

```
1  from datetime import datetime
2  from Facet import Facet
3  from Vertex import Vertex
4
5  import io
6  import os
7  import sys
8
9  DEFAULT_ELEMENT_DESCRIPTION    = '3 {} 0 0 0 0 0'
10 DEFAULT_FILE_EXTENSION_OUTPUT = '.facet'
11 DEFAULT_PART_COUNT             = '1'
12 DEFAULT_PART_MIRROR            = '0'
13 DEFAULT_PART_NAME              = '<PTW MeshModel>'
14 DEFAULT_SUBPART_COUNT          = '1'
15 DEFAULT_SUBPART_NAME           = '<PTW MeshSheet>'
16
17 argumentCount = len(sys.argv)
18
19 # output argument-wise
20 if argumentCount == 2:
21     objectFileName = sys.argv[1]
22     outputFileName = os.path.splitext(objectFileName)[0] +
           DEFAULT_FILE_EXTENSION_OUTPUT
23 elif argumentCount == 3:
24     objectFileName = sys.argv[1]
```

# Python Tool for *.obj to *.facet II

```
25      outputFileName = sys.argv[2]
26  else:
27      sys.stderr.write('Usage: python Obj2Facet.py <input-obj-file-name>
            [<output-facet-file-name>]\n')
28      sys.exit()
29
30  facetCount  = 0
31  facetLines  = ""
32  vertexCount = 0
33  vertexLines = ""
34  with io.open(objectFileName, 'r', encoding='utf-8') as objectFile:
35      line = objectFile.readline()
36      lineNumber = 1
37      while line:
38          tokens = line.strip().split(' ')
39          if len(tokens) == 4:
40              type = tokens[0]
41
42              if type.lower() == 'f':
43                  facetLines += ' '.join(tokens[1:4])
44                  facetLines += ' 0'
45                  facetLines += '\n'
46                  facetCount += 1
47
48              elif type.lower() == 'v':
```

# Python Tool for *.obj to *.facet III

```
49                  vertexLines += ' '.join(tokens[1:4])
50                  vertexLines += '\n'
51                  vertexCount += 1
52
53          line         = objectFile.readline()
54          lineNumber += 1
55
56      objectFile.close()
57
58  with io.open(outputFileName, 'w', encoding='utf-8') as outputFile:
59      outputFile.write('FACET FILE V3.4 ')
60      outputFile.write(datetime.today().strftime('%d-%b-%Y %H:%M:%S'))
61      outputFile.write('\n')
62
63      outputFile.write(DEFAULT_PART_COUNT)
64      outputFile.write('\n')
65      outputFile.write(DEFAULT_PART_NAME)
66      outputFile.write('\n')
67      outputFile.write(DEFAULT_PART_MIRROR)
68      outputFile.write('\n')
69
70      outputFile.write(str(vertexCount))
71      outputFile.write('\n')
72      outputFile.write(vertexLines)
73
```

# Python Tool for *.obj to *.facet IV

```
74      outputFile.write(DEFAULT_SUBPART_COUNT)
75      outputFile.write('\n')
76      outputFile.write(DEFAULT_SUBPART_NAME)
77      outputFile.write('\n')
78
79      outputFile.write(DEFAULT_ELEMENT_DESCRIPTION.format(facetCount))
80      outputFile.write('\n')
81      outputFile.write(facetLines)
82
83      outputFile.close()
```

# Mathematica Commands I

```
(* Generate a mesh for a convex hull of points on a sphere *)
mesh = ConvexHullMesh[SpherePoints[60]] // Region;

(* Extract vertex coordinates from the mesh *)
vlist = MeshCoordinates[mesh];

(* Label each point in the mesh *)
labeledPoints = MapIndexed[
    Text[Style[ToString[#2[[1]]], Bold, Black], #1] &, vlist
];
```

# *.obj: Vertices and Plaquettes

**Boo**

# *.geo: Material Properties

**Boo**

# Bibliography I

# Data Reduction for Modelling Satellite Radar Cross Sections

**Daniel Topa**
**daniel.topa@hii.com**

**Huntington Ingalls Industries**
**Mission Technologies**

**December 19, 2024**