

# week 2

## mesh refinement - Lochana

process of improving the quality and detail of a 3D model's surface by increasing the number of triangles that make it up

There are two main types of mesh refinement you can use:

- **Uniform Subdivision:** This is the simplest method. It takes every triangle in the mesh and splits it into four smaller triangles. This increases the total number of vertices and faces, creating a more detailed mesh.
- **Subdivision Surfaces:** These are more advanced algorithms that not only add new triangles but also smooth the surface by adjusting the positions of the new vertices. The most common algorithms are **Loop subdivision** for meshes made of triangles and **Catmull-Clark** for meshes made of quadrilaterals.

PyVista supports three different subdivision algorithms: `linear`, `loop`, and `butterfly`.

- `linear`: This is the fastest method. It simply splits each triangle into four, without any smoothing.
- `loop`: This algorithm is specifically designed for triangular meshes and provides a smooth result. It's the best option for getting a more accurate curvature reading.
- `butterfly`: This is an interpolating scheme that preserves the original vertices and also results in a smoothed mesh.

```
import pyvista as pv

# Load your original mesh
mesh = pv.read(r"C:\Users\user\Documents\1 code\Research\data set\free for m\32770.stl")
print("Original Mesh Info:")
print(f"Number of Vertices: {mesh.n_points}")
print(f"Number of Faces: {mesh.n_cells}")
```

```

# Perform mesh refinement using the Loop subdivision algorithm
# nsub=1 means each face is split into 4 smaller faces.
refined_mesh = mesh.subdivide(nsub=1, subfilter='loop')

# Print the new, refined mesh's information
print("\nRefined Mesh Info:")
print(f"Number of Vertices: {refined_mesh.n_points}")
print(f"Number of Faces: {refined_mesh.n_cells}")

# Visualize the refined mesh to see the result
# This will open a new window showing the more detailed mesh.
refined_mesh.plot(show_edges=True, show_bounds=True)

```

- -- Curvature Statistics for the REFINED Mesh ---  
 Minimum curvature: -412.8250  
 Maximum curvature: 1236.1607  
 Average curvature: 0.1443  
 Number of values: 75770
- -- Curvature Statistics without refined mesh---  
 Minimum curvature: -83.4959  
 Maximum curvature: 197.3721  
 Average curvature: 0.0236  
 Number of values: 18944

The number of faces has increased by four times

## multi-body mesh

we tried to identify that are there more than one 3d model in a one stl file.

```

import pyvista as pv

# Load the single STL file containing multiple models
multi_body_mesh = pv.read(r"C:\Users\user\Documents\1 code\Research\data
set\43852.stl")

# Split the mesh into a list of individual bodies
bodies = multi_body_mesh.split_bodies()

print(f"The file contains {len(bodies)} separate models.")

# You can now loop through each model and analyze its curvature individually
for i, body in enumerate(bodies):
    # Check if the body is a PolyData object before processing
    if isinstance(body, pv.PolyData):
        print(f"\n--- Analyzing Body #{i + 1} ---")

        # Calculate curvature on this individual body
        body_curvature = body.curvature(curv_type='gaussian')

        print(f"Number of Vertices: {body.n_points}")
        print(f"Average Curvature: {body_curvature.mean():.4f}")

    else:
        # Skip this body if it's not a PolyData object
        print(f"\n--- Skipping Body #{i + 1} (not a valid mesh for curvature analys
is) ---")

```

## Mesh Refinement & Curvature - presha

### 1.Smaller Scale

- Refine mesh: `refined_mesh = mesh.subdivide(nsub=1, subfilter='loop')` -  
Smooth carefully: `smoothed = refined_mesh.smooth(n_iter=20, relaxation_factor=0.05)`

- Curvature: ``gaussian_curvatures_refined = refined_mesh.curvature('gaussian')``

## 2 .Constraints

- High-curvature points:

```
selected_points = refined_mesh.points[gaussian_curvatures_refined > 0.1]
```

- Split bodies: `bodies = refined_mesh.split_bodies()`

- Bounding box: `subset = refined_mesh.clip_box([xmin, xmax, ymin, ymax, zmin, zmax])`

When working with 3D meshes, it is important to capture small details and focus only on the parts we want to study. Small features can be missed if the mesh is too coarse. To fix this, we can refine the mesh using subdivision, which adds more vertices and faces. Using more subdivisions helps capture smaller features better, but too much refinement can make the mesh very big and slow to work with. After refining, you can apply gentle smoothing to reduce noise while keeping the small details. Calculating Gaussian or Mean curvature on the refined mesh helps highlight small bumps, ridges, or dents, giving a clearer picture of the surface shape.

To focus on specific areas, we can select vertices based on curvature, so only high-curvature regions, like sharp edges, are analyzed. For STL files with multiple disconnected parts, we can split each part and analyze them separately, avoiding interference between objects. we can also use a bounding box to limit analysis to a certain 3D region. In summary, refining the mesh, smoothing carefully, and using constraints helps analyze complex meshes accurately and focus on the important details.

```
import pyvista as pv
import numpy as np

# 1. Load mesh
mesh = pv.read(r"C:\Users\user\Documents\1 code\Research\data set\43852.stl")
print("Original Mesh Info:")
print(f"Vertices: {mesh.n_points}, Faces: {mesh.n_cells}")
```

```

# 2. Refine mesh to capture small-scale features
refined_mesh = mesh.subdivide(nsub=2, subfilter='loop') # increase nsub if
needed
print("\nRefined Mesh Info:")
print(f"Vertices: {refined_mesh.n_points}, Faces: {refined_mesh.n_cells}")

# Optional: gentle smoothing (preserve small details)
smoothed_mesh = refined_mesh.smooth(n_iter=20, relaxation_factor=0.05)

# 3. Compute Gaussian curvature
gaussian_curvature = smoothed_mesh.curvature(curv_type='gaussian')
smoothed_mesh['gaussian_curvature'] = gaussian_curvature

# 4. Apply constraint: select only high-curvature regions
threshold = 0.1 # adjust depending on your model
high_curv_mask = gaussian_curvature > threshold

# Create a new mesh containing only high-curvature points
high_curv_mesh = smoothed_mesh.extract_points(high_curv_mask)

print(f"\nHigh-Curvature Region Info:")
print(f"Vertices: {high_curv_mesh.n_points}, Faces: {high_curv_mesh.n_cells}")
print(f"Average curvature in constrained region: {gaussian_curvature[high_curv_mask].mean():.4f}")

# 5. Visualization
plotter = pv.Plotter(shape=(1,2))

# Full refined mesh with curvature
plotter.add_mesh(smoothed_mesh, scalars='gaussian_curvature', cmap='viridis', show_scalar_bar=True)
plotter.subplot(0,1)

```

```
# Only high-curvature constrained region
plotter.add_mesh(high_curv_mesh, scalars='gaussian_curvature', cmap='plasma', show_scalar_bar=True)
plotter.show()
```

## Curvature classification - Imasha

```
# batch_curvature_classification.py

import os
import trimesh
import numpy as np

1. Set folder path containing STL files

folder_path = r"I:\Thing10K\raw_meshes" # change to your folder
Threshold for flat vs free-form (adjust if needed) curvature_threshold = 1e-3

# Optional: PyVista visualization

use_visualization = False

try:
    import pyvista as pv
    use_visualization = True
except ModuleNotFoundError:
    print("PyVista not installed. Visualization will be skipped.")

# -----

# 2. Function to compute curvature
```

```

# -----

def compute_gaussian_curvature(mesh):
    curvatures = np.zeros(len(mesh.vertices))
    for i, vertex in enumerate(mesh.vertices):
        faces_idx = np.where(mesh.faces == i)[0]
        angle_sum = 0.0
        for f in faces_idx:
            face = mesh.faces[f]
            other_idx = [idx for idx in face if idx != i]
            v0 = mesh.vertices[other_idx[0]] - vertex
            v1 = mesh.vertices[other_idx[1]] - vertex
            cos_angle = np.dot(v0, v1) / (np.linalg.norm(v0) * np.linalg.norm(v1))
            cos_angle = np.clip(cos_angle, -1.0, 1.0)
            angle = np.arccos(cos_angle)
            angle_sum += angle
        curvatures[i] = 2 * np.pi - angle_sum
    return curvatures

# 3. Process all STL files in folder

for filename in os.listdir(folder_path):
    if filename.lower().endswith(".stl"):
        file_path = os.path.join(folder_path, filename)
        mesh = trimesh.load_mesh(file_path)
        curvatures = compute_gaussian_curvature(mesh)
        max_curv = np.max(curvatures)

# Classify model
model_type = "Flat Model" if max_curv < curvature_threshold else "Free-form Model"
print(f"{filename}: {model_type}")

# Optional visualization of the first file

if use_visualization and filename == os.listdir(folder_path)[0]:
    faces_pv = np.hstack((np.full((len(mesh.faces), 1), 3), mesh.faces)).astype

```

```
(np.int64) mesh_pv = pv.PolyData(mesh.vertices, faces_pv)
curv_norm = (curvatures - np.min(curvatures)) / (np.max(curvatures) -
np.min(curvatures) + 1e-12)
```

```
p = pv.Plotter()
```

```
p.add_mesh(mesh_pv, scalars=curv_norm, show_scalar_bar=True)
p.add_axes()
p.show()
```

```
import os
```

```
import trimesh
```

```
import numpy as np
```

```
1. Set folder path containing STL files
```

```
folder_path = r"!:\Thing10K\raw_meshes" # change to your folder
```

```
Threshold for flat vs free-form (adjust if needed) curvature_threshold = 1e-3
```

```
# Optional: PyVista visualization
```

```
use_visualization = False
```

```
try:
```

```
import pyvista as pv
```

```
use_visualization = True
```

```
except ModuleNotFoundError:
```

```
print("PyVista not installed. Visualization will be skipped.")
```

```
# -----
```

```
# 2. Function to compute curvature
```

```
# -----
```

```
def compute_gaussian_curvature(mesh):
```

```
curvatures = np.zeros(len(mesh.vertices))
```

```
for i, vertex in enumerate(mesh.vertices):
```

```
faces_idx = np.where(mesh.faces == i)[0]
```

```
angle_sum = 0.0
```

```
for f in faces_idx:
```

```
face = mesh.faces[f]
```

```
other_idx = [idx for idx in face if idx != i]
```

```
v0 = mesh.vertices[other_idx[0]] - vertex
```



```

v1 = mesh.vertices[other_idx[1]] - vertex
cos_angle = np.dot(v0, v1) / (np.linalg.norm(v0) * np.linalg.norm(v1)) cos_angle = np.clip(cos_angle, -1.0, 1.0)
angle = np.arccos(cos_angle)
angle_sum += angle
curvatures[i] = 2 * np.pi - angle_sum
return curvatures

# 3. Process all STL files in folder
for filename in os.listdir(folder_path):
    if filename.lower().endswith(".stl"):
        file_path = os.path.join(folder_path, filename) mesh = trimesh.load_mesh(file_path)
        curvatures = compute_gaussian_curvature(mesh) max_curv = np.max(curvatures)

# Classify model
model_type = "Flat Model" if max_curv < curvature_threshold else "Free-form Model" print(f"{filename}: {model_type}")

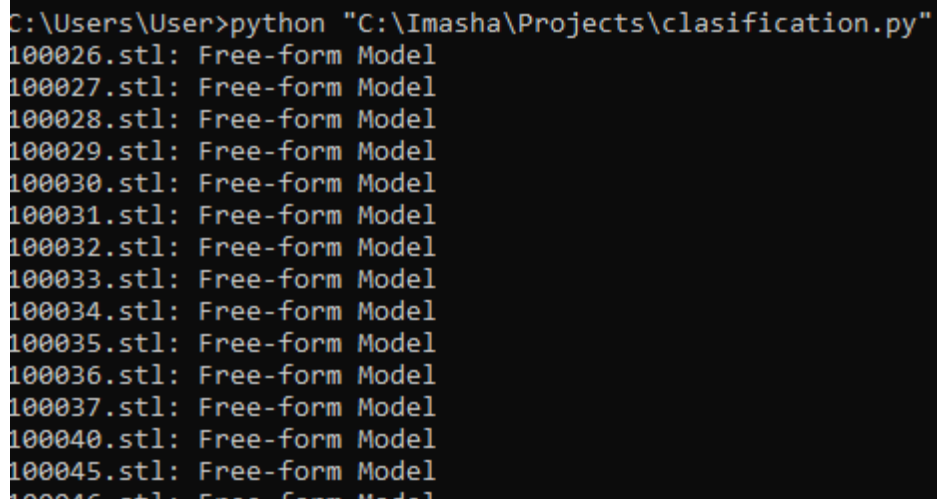
# Optional visualization of the first file
if use_visualization and filename == os.listdir(folder_path)[0]:
    faces_pv = np.hstack((np.full((len(mesh.faces),1),3), mesh.faces)).astype(np.int64) mesh_pv =
pv.PolyData(mesh.vertices, faces_pv)

    curv_norm = (curvatures - np.min(curvatures)) / (np.max(curvatures) -
np.min(curvatures) + 1e-12)

    p = pv.Plotter()

    p.add_mesh(mesh_pv, scalars=curv_norm, show_scalar_bar=True)
    p.add_axes()

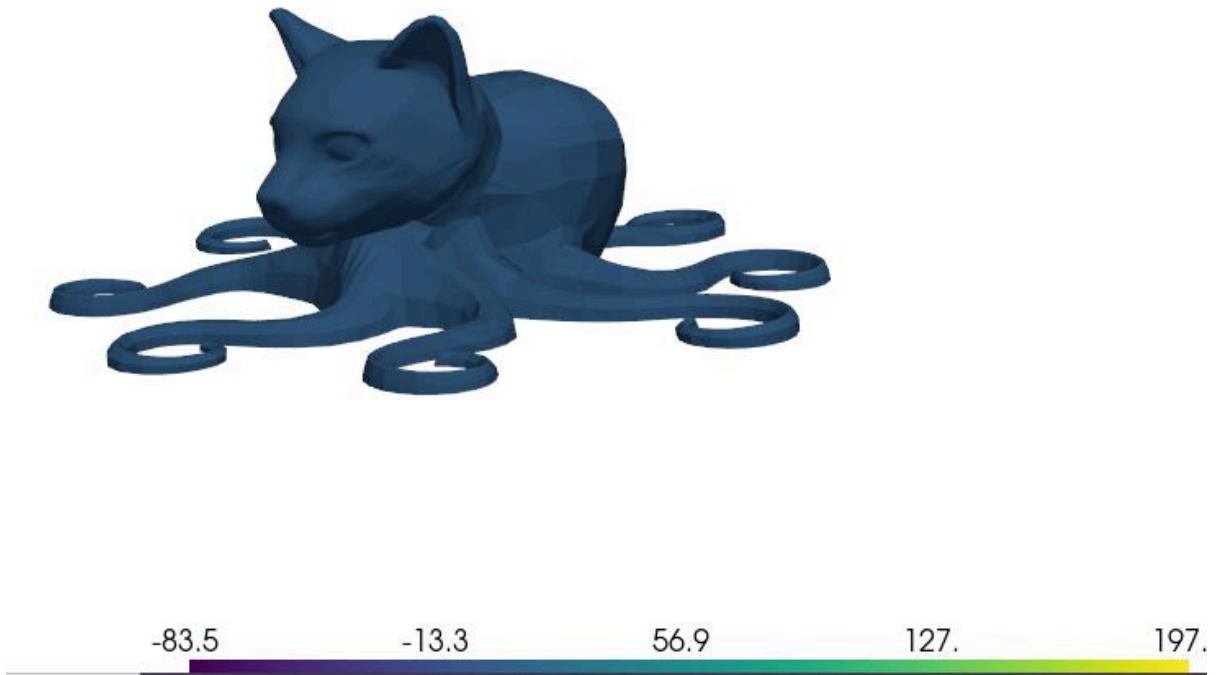
```



```

C:\Users\User>python "C:\Imasha\Projects\clasification.py"
100026.stl: Free-form Model
100027.stl: Free-form Model
100028.stl: Free-form Model
100029.stl: Free-form Model
100030.stl: Free-form Model
100031.stl: Free-form Model
100032.stl: Free-form Model
100033.stl: Free-form Model
100034.stl: Free-form Model
100035.stl: Free-form Model
100036.stl: Free-form Model
100037.stl: Free-form Model
100040.stl: Free-form Model
100045.stl: Free-form Model
100046.stl: Free-form Model

```



```
import pyvista as pv
import numpy as np

# 1. Load the mesh (use your correct path)
mesh = pv.read(r"!:\Thing10K\raw_meshes\32770.stl")

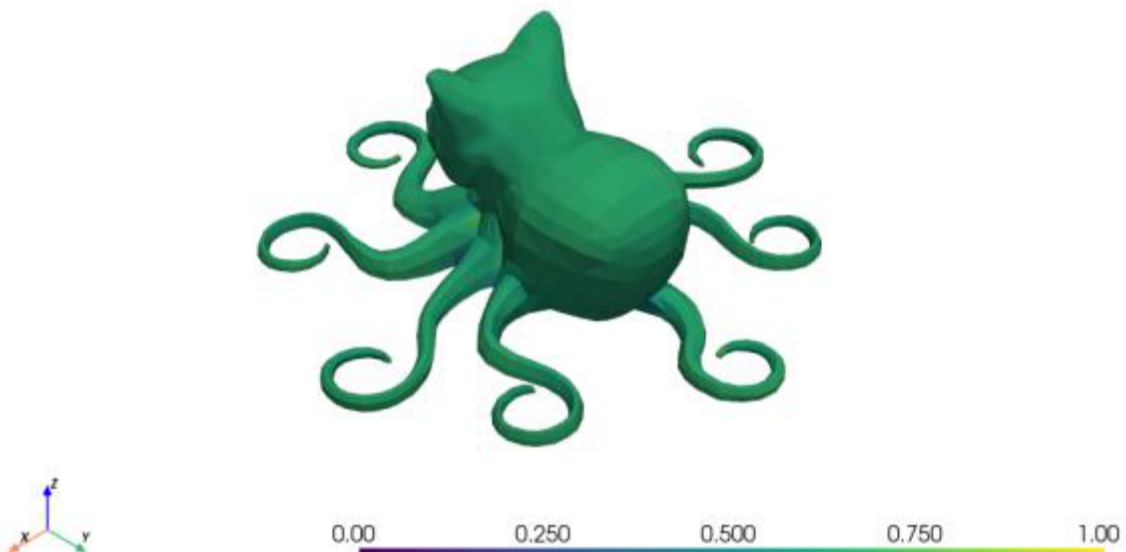
# 2. Compute curvature
gaussian_curvatures = mesh.curvature(curv_type='gaussian')

# 3. Print and analyze the values
print("The first 10 curvature values are:")
print(gaussian_curvatures[:10])
print("\n--- Curvature Statistics ---")
```

```
print(f"Minimum curvature: {np.min(gaussian_curvatures):.4f}") print(f"Maximum curvature: {np.max(gaussian_curvatures):.4f}") print(f"Average curvature: {np.mean(gaussian_curvatures):.4f}") print(f"Number of values: {len(gaussian_curvatures)}")
```

# 4. (Optional) visualize

```
p = pv.Plotter()
p.add_mesh(mesh, scalars=gaussian_curvatures, show_scalar_bar=True)
p.show()
```



```
import trimesh
import numpy as np
```

```
# ----- # 1. Load the STL mesh # -----
# curvature_model_analysis.py
mesh_path = r"I:\Thingi10K\raw_meshes\32770.stl" # change to your STL
path mesh = trimesh.load_mesh(mesh_path)
```

```

print(f"Mesh loaded: {mesh_path}")
print(f"Number of vertices: {len(mesh.vertices)}, faces: {len(mesh.faces)}")

# -----
# 2. Compute Gaussian curvature (angle deficit) # -----
----

curvatures = np.zeros(len(mesh.vertices))
for i, vertex in enumerate(mesh.vertices):
    faces_idx = np.where(mesh.faces == i)[0]
    angle_sum = 0.0
    for f in faces_idx:
        face = mesh.faces[f]
        other_idx = [idx for idx in face if idx != i]
        v0 = mesh.vertices[other_idx[0]] - vertex
        v1 = mesh.vertices[other_idx[1]] - vertex
        cos_angle = np.dot(v0, v1) / (np.linalg.norm(v0) * np.linalg.norm(v1))
        cos_angle = np.clip(cos_angle, -1.0, 1.0)
        angle = np.arccos(cos_angle)
        angle_sum += angle
    curvatures[i] = 2 * np.pi - angle_sum

# -----
# 3. Print curvature stats
# -----

min_curv = np.min(curvatures)
max_curv = np.max(curvatures)
mean_curv = np.mean(curvatures)
print("\n--- Curvature Statistics ---")
print("First 10 curvature values:", curvatures[:10])
print(f"Minimum curvature: {min_curv:.6f}")
print(f"Maximum curvature: {max_curv:.6f}")
print(f"Average curvature: {mean_curv:.6f}")

# -----

```

```

# 4. Decide model type
# -----
# Define threshold for flat vs free-form
threshold = 1e-3 # adjust if needed
if max_curv < threshold:
    model_type = "Flat Model"
else:
    model_type = "Free-form Model"
print(f"\n--- Model Type ---\nThis model is classified as: {model_type}")

# -----
# 5. Optional: Visualize with PyVista
# -----
try:
    import pyvista as pv

    # Convert trimesh faces to PyVista format
    faces_pv = np.hstack((np.full((len(mesh.faces),1),3), mesh.faces)).astype
    (np.int64) mesh_pv = pv.PolyData(mesh.vertices, faces_pv)
    # Normalize curvature for coloring

    curv_norm = (curvatures - np.min(curvatures)) / (np.max(curvatures) - np.
    min(curvatures) + 1e-12)
    p = pv.Plotter()
    p.add_mesh(mesh_pv, scalars=curv_norm, show_scalar_bar=True)
    p.add_axes()
    p.show()

except ModuleNotFoundError:
    print("\nPyVista not installed. Skipping visualization. To see 3D plot, run: pi
    p install pyvista pyvistaqt")

```

```
C:\Users\User>python "C:\Imasha\Projects\curvature_model_analysis.py"
Mesh loaded: I:\Thingii10K\raw_meshes\32770.stl
Number of vertices: 18944, faces: 37884

--- Curvature Statistics ---
First 10 curvature values: [-0.04568433 -0.16935457 -0.21727065 -0.13307396 -0.05595086 -0.14037562
-0.07343839 -0.02218619 -0.0495411 -0.00435835]
Minimum curvature: -3.814224
Maximum curvature: 2.164721
Average curvature: 0.000663

--- Model Type ---
This model is classified as: Free-form Model
```

clasification.py: Best when you have lots of models and want to quickly sort them into Flat or Free-form.

curvature.py: Best when you want a quick preview of one model.

curvature\_model\_analysis.py: Best when you want detailed information and a clear decision on one model.