2-Farthest_Point_Sampling

February 15, 2018

In [1]:
```
import pymesh
import numpy as np
from ipywidgets import FloatProgress
from IPython.display import display
from scipy.spatial import distance_matrix

object_name = "teapot" # object_name = "violin_case"
mesh = pymesh.load_mesh("%s.obj" % object_name)
```

In [2]:
```
################################################################################
# def triangle_area(v0, v1, v2):
# Returns the area of a triangle given three points (v0, v1, v2)
# Given by Area = |AB x AC| / 2 (half of the length of the cross product)
################################################################################
def triangle_area(v0, v1, v2):
    return np.linalg.norm(np.cross(np.array(v1) - np.array(v0),
                                   np.array(v2) - np.array(v0))) * 0.5
```

In [3]:
```
# Calculate the total surface area of the mesh
total_area = 0
for face in mesh.faces:
    v0 = mesh.vertices[face[0]]
    v1 = mesh.vertices[face[1]]
    v2 = mesh.vertices[face[2]]
    total_area += triangle_area(v0, v1, v2)
```

In [4]:
```
# Calculate weight per triangle
triangle_weights = []
for face in mesh.faces:
    v0 = mesh.vertices[face[0]]
    v1 = mesh.vertices[face[1]]
    v2 = mesh.vertices[face[2]]
    triangle_weights.append(triangle_area(v0, v1, v2) / total_area)
```

In [5]:
```
# Sample points along mesh surface
num_points = 10000
point_cloud = []
for face, weight in zip(mesh.faces, triangle_weights):
```

```
            num_points_in_triangle = weight * num_points
            v0 = mesh.vertices[face[0]]
            v1 = mesh.vertices[face[1]]
            v2 = mesh.vertices[face[2]]
            for _ in range(int(np.ceil(num_points_in_triangle))):
                r1 = np.random.rand()
                r2 = np.random.rand()
                D = (1 - np.sqrt(r1)) * v0 + np.sqrt(r1) * (1 - r2) * v1 + np.sqrt(r1) * r2 * v
                point_cloud.append(D)

                if len(point_cloud) >= num_points:
                    break
            if len(point_cloud) >= num_points:
                    break
```

In [6]:
```
num_samples = 1000

i = np.random.randint(0, len(point_cloud))
S = [point_cloud[i]]
del point_cloud[i]

progress = FloatProgress(min=1, max=num_samples); display(progress)

# Generate new point cloud using farthest point sampling
while len(S) < num_samples:
    D = distance_matrix(S, point_cloud)
    progress.value = len(S)
    progress.description = "%i/%i" % (len(S), num_samples)
    d_max = 0
    for i in range(len(point_cloud)):
        d_min = float("inf")
        for j in range(len(S)):
            d_min = min(D[j, i], d_min)
        if d_min > d_max:
            d_max = d_min
            q_farthest = i
    S.append(point_cloud[q_farthest])
    del point_cloud[q_farthest]
```

```
FloatProgress(value=1.0, max=1000.0, min=1.0)
```

In [7]:
```
import pickle
def save_object(obj, filename):
    with open(filename, 'wb') as output:
        pickle.dump(obj, output, pickle.HIGHEST_PROTOCOL)
save_object(S, "%s.cloud" % object_name)
```