Finding Topological Invarient

Currently, there are many algorithm has been purposed for the surface reconstruction of Manifold and Non-Manifold surfaces from the point cloud data even in high dimensions. Apart from surface reconstruction some algorithm has been also purposed for finding Topological Invariant in the surfaces like holes, connected components and cavities.Our aim here is to detect topological invariant of a figure.

1 What work has been done

Application and work done

- .name proposed K- Laplacian spectrum method to find higher topological invariant, hole based on persistent homology. They applied method to the metabolic network based on FDG-PET data of Alzheimer Disease (AD), mild cognitive impairment(MCI) and normal control (NC) groups. They were successful in finding the persistence of hole in Alzheimer Disease.

- Proposed[1] a framework for reconstructing lightweight polygonal surfcae from point cloud based on hypothesizing and selection strategy with help of binary linear programming formula. It focus on reconstructing piecewise planar objects (i.e., man-made objects such as buildings). a)2-manifold and watertight
  b)it should be able to recover sharp features of the objects
  c)the method should not closely follow surface details due to imperfections.
  Primitive Extraction
  - Researches falling in this category aim at extracting high-quality instances of basic geometric primitives (e.g., plane, cylinder) from point clouds corrupted by noise and outliers. The common practice for this particular task is the Random Sample Consensus (RANSAC)
  Primitive Regularization
  By exploiting the prior knowledge about the structure of an object, researchers further regularize the extracted primitives. Li et.al. discover global mutual relations between basic primitives and use such information as constraints to refine the initial primitives base on local fitting and constrained optimization
  Hypothesis based reconstruction
  Their strategy is generating a non-uniform grid and then selecting a subset of its cells

that have good data support and are smooth. In our work, we generalize this idea to reconstruct general piecewise planar objects, and our reconstruction is based on optimization under hard constraints that guarantee manifold and watertight polygonal surface models.

Candidate face generation

We first extract a set of planar segments from the point cloud using RANSAC .We refine these planar segments by iteratively merging plane pairs and fitting new planes

Face selection

We choose an optimal subset of the candidate faces to assemble a manifold and watertight polygonal surface model. To do so, we formulate the face selection as a binary linear programming problem.objective function combines three terms that favor data fitting, point coverage, and model complexity, respectively.

Plane extraction

We use the RANSAC-based primitive detection method proposed by Schnabel et al. [24] to detect a set of initial planar segments S = si from the point cloud P. each points can be assigned to no more than one plane.

Plane refinement

we first compute the angle of the supporting planes for each pair of planar segments. Then, starting from the pair (si, sj) with the smallest angle, we test if the following two conditions are met. First, the angle between the two planes is lower than a threshold., i.e., angle(si, sj) $<\theta$t. Second, more than a specified number (denoted as Nt) of points lie on the supporting planes of both segments. If both conditions are satisfied, we merge the two planar segments and fit a new supporting plane using PCA.θt =10° and Nt = min(|si |, |sj |)/5 Pairwise intersecting To hypothesize the object's faces, we crop the supporting planes of all planar segments by the bounding box of the point cloud.we compute pairwise intersections of the cropped planes (see Figure 1 (e)). It should be noted that pairwise intersections introduce redundant candidate faces. Since most of the redundant faces do not represent actual structures of an object, they are supported by no or very few (due to noise and outliers) point samples. The subsequent optimization based face selection is designed to favor choosing the most confident faces while satisfying certain constraints The pairwise intersections maintain incidence information of the faces and edges. Each edge of a candidate face is either connecting four neighboring candidate faces or representing a boundary. For example in Figure 2 (a), edge e connects four faces while others are boundaries. We rely on such incidence information to formulate our manifold and watertight constraints for face selection.

Face selection

Given N candidate faces F = fi |1 <i <N generated in the previous step, we select a subset of these candidate faces that can best describe the geometry of the object and ensure that the chosen faces form a manifold and watertight polygonal surface. This is achieved through optimization. We define multiple energy terms that constitute our objective function. Data-fitting,Point coverage, Model complexity With the above energy terms, the optimal set of faces can be obtained by minimizing a weighted sum of these terms under certain hard constraints enforcing the final model to be manifold

without boundary. Thus, the final formulation for face selection can be written as min

$$X = \lambda_f \cdot E_f + \lambda_m \cdot E_m + \lambda_c \cdot E_c$$

with given constraints

- . name presented an algorithm capable of reconstructing a non-manifold surface embedded as a point cloud in a high-dimensional space.They modified freeman triangulation theorem for using that for any type of non-manifold surface.
- A software named Persus developed by Dr.Videt Nanda provide information about topological invariant such as holes and tunnels.

2 Work Done By Us

With objective of finding topological invariant in shapes along with forming their point cloud, we have done some work toward constructing point cloud of some given shapes with help of their parametric coordinates. We are able to find topological invariant of shapes by having their domain's specifications.

2.1 Point Cloud Generation

We discovered that point cloud of a known surface can be generated by two methods, one by rejection method and other by parametric method.

- **Rejection Method**
  In Rejection Method, we generated a rectangular domain greater than the known surface. Since, we know the domain and geometry of the shape we can generate random points all over the rectangular suface using rand function. We need the random points inside or on the surface of the shape so we reject those points who lies outside the boundary of the surface.

- **Parametric Method**
  In parametric method, we generated a point cloud surface using variable parametrization of the known shape. By having parametric coordinates we again use Rand function to generate random point cloud.

Examples

1. **Ring**
   Let n be the number of random points to be generated inside the ring and r1 be the

inner radius and r2 be the whole radius of the ring. We can draw random point cloud in the ring by the following method in Matlab.

Parametric Method

```
r = r1+(r2-r1)*rand(1,n)
t = 2*pi*rand(1,n)
x = r.*cos(t)
y = r.*sin(t)
plot(x,y,'r.')
```

2. **Ellipse**
   Let n be the number of random points to be generated inside the ellipse and a be the major axis and b be the minor axis.
   Parametric Method

```
r=sqrt(rand(1,n))
var= 2*pi*rand(1,n)
x=r*a.*cos(var)
y=r*b.*sin(var)
plot(x,y,'r.')
```

   Rejection Method

```
j= 1
i=1
x= zeros(1,n)
y= zeros(1,n)
while i<=1000
   a1 = rand*a
    b1 = rand*b
    var = (a1^2)/a^2 + (b1^2)/b^2 -1
    var1 = (a1^2)/a + (b1^2)/b -1
    j=j+1
    if(var <=0 && var1 >0)
        x(i) = a1
        y(i) = b1
        i= i+1
    end
end
plot([x -x -x x],[y y -y -y],'r.')
```

3. **Torus**

   Let R be the bigger radius and r be the smaller radius of Torus. Then we can generate

random point cloud in Matlab by following methods

Parametric Method

```
R=100
r=35
n=10000
u=2*pi*rand(1,n)
v=2*pi*rand(1,n)
x= (R + r*cos(v)).*cos(u)
y= (R + r*cos(v)).*sin(u)
z= r*sin(v)
plot3(x,y,z,'.r')
```

Rejection Method

```
x= zeros(1,n)
y= zeros(1,n)
z= zeros(1,n)
i = 1
j=0
while i<n
    a1 = R*rand
    b1 = R*rand
    c1 = r*rand
    j=j+1
    var = c1^2 + (R- sqrt(a1^2 + b1^2))^2 -r^2
    if(var<=0)
        x(i) = a1
        y(i) = b1
        z(i) = c1
        i= i+1
    end
end

plot3([x -x -x x x -x -x x],[y y -y -y y y -y -y],[z z z z -z -z -z -z],'.r')
axis equal
```

4. **Cylinder**

Let R be the bigger radius and r be the smaller radius while h be the height of the Cylinder. Then we can generate the n random point cloud of Cylinder in Matlab by following methods

Parametric Method

```
r = r+(R-r)*rand(1,n)
t = 2*pi*rand(1,n)
x = r.*cos(t)
y = r.*sin(t)
z = 5*rand(1,n)
plot3(x,y,z,'r.')
axis equal
```

5. **Sphere**
   Let r be the radius of sphere and n be the number of random points. We can also generate random point cloud of shpere by both Reduction and Parametric Method.
   Parametric Method

```
r=5
n=1000
t=pi*rand(1,n)
T=2*pi*rand(1,n)
x=r.*sin(t).*cos(T)
y=r.*sin(t).*sin(T)
z=r.*cos(t)
plot3(x,y,z,'r.')
```

   Redcution Method

```
i=1
x=zeros(1,n)
y=zeros(1,n)
z=zeros(1,n)
while i<=n
    a = 16*rand
    b = 16*rand
    c = 16*rand

    if( (a-r)^2 + (b-r)^2 + (c-r)^2 <=r^2)
        x(i)=a
        y(i)=b
        z(i)=c
        i=i+1
    end
end
plot3(x,y,z,'r.')
```

2.2 Random Numbers Generators

2.2.1 LCG Generator

Algorithm
Balance of linear momentum requires that the rate of change of linear momentum in volume 'V'is equal to applied force, which include body forces and forces which act across boundary of V.
LCG generator is defined by recurrence relation:
Xn+1 = (aXn + c) Mod(m)
Here a is a constant multiplication factor, c is addition factor and m is modulus.
m>0, 0<a>m, 0<=c<m, 0<=x0<m
If c= 0 then it is pure multiplicative congruential generator and known as mixed congruential generator and it has full period of (m-1) and can be subjected to spectral test.
For the time period length of the sequence of random numbers = m

- c and m must be relatively prime or co-prime (their common divisor is 1) or gcd(c,m)=1
- The a^{th} phase is a viscous fluid.a-1 should be divisible by all prime factors of m
- The b^{th} phase is in the form of particle, bubble or droplets distributed throughout the region.a-1 should divisible by 4 if m is divisible by 4.

Pseudocode

```
LCG <-function(n,m,a,c,X0)
X <-c()
Xn <- X0
for (i in 1:n) {
Xn <- (a*Xn + c) % % m
X[i] <- Xn
}
X <- X/m
return(X)
```

It can pass formal test of randomness

Disadvantages

- LCG is vulnerable to the cyber attack if used to generate keys in cryptorism because it is possible to recover the parameters of LCGs in polynomial time given several observed outputs

2.2.2 Blum Blum Shub Generator

It generate the pseudo random generators via following recurence relation:

$$Xn+1=(X)^{2mod m}$$

- Output is the least significant bit of Xn+1 or parity bit of Xn+1.
- M is product of two larger prime numbers that should be congruent to 3Mod4 (If a is congruent to bModc then a-b should divisible by c).
- Seed X0 should co-prime with M and should not be 1 or 0.

Disadvantages
However, BBS is not a permutation generator as the RNGs mentioned above. A -permutation generator is a program which produces permutations of a set of distinct objects. Each of these permutations is called a -permutation. A full-period RNG with period length is also an -permutation generator since any consecutive outputs form an -permutation. The period length of BBS is much lesses than , therefore, it can not be adapted to produce uniform random variates between 1 and . For example, use use , and hence the period is much lesser than 16873. Due to this reason, BBS failed all of the statistical tests we have conducted and is therefore not suitable for stochastic simulations. There is a proof reducing its security to the computational difficulty of solving the quadratic residuosity problem.[1] When the primes are chosen appropriately, and O(log log M) lower-order bits of each xn are output, then in the limit as M grows large, distinguishing the output bits from random should be at least as difficult as solving the Quadratic residuosity problem modulo M.
Example
If p=11, q=19, X0=3 then M= 209
Sequence will 9, 81, 82, 36, 42, 92.

2.2.3 Mersenne Twister Algorithm

Its first state doesn't produce any random numbers. The next states after initial state produce long sequence of 264 random numbers. (The state is transformed into random numbers via a g function).
It has long period of $(2)^{19937-1}$

```
Here Iw-1 is the (w-1) by (w-1) identity matrix and
```

2.2.4 RC4

RC4 has 2 steps

- KSA
- PRNG To begin require a key between 40bits and 256 bits.

Strength

- The difficulty of knowing where any value is in the table.
- The difficulty of knowing which location in the table is used to select each value in the sequence.
- A particular RC4 Algorithm key can be used only once.
- Encryption is about 10 times faster than DES.

Weakness

- The algorithm is vulnerable to analytic attacks of the state table.
- One in every 256 keys can be a weak key. These keys are identified by cryptanalysis that is able to find circumstances under which one of more generated bytes are strongly correlated with a few bytes of the key.
- WEAK KEYS: these are keys identified by cryptanalysis that is able to find circumstances under which one or more generated bytes are strongly correlated with small subset of the key bytes. These keys can happen in one out of 256 keys generated.

KSA

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap values of S[i] and S[j]
endfor
```

Pseudo Random Number Generator

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```

References

[1] Liangliang Nan, Peter Wonka, 'PolyFit: Polygonal Surface Reconstruction from Point Clouds', (2017).

[2] Truesdell, C. and Toupin, 'R.Classical Field Theories in Volume III/1 of handbuch der physik', *S. Flugge, ed.,springer Verlag*, Berlin(1960).

[3] Jaunzemis, 'W. Continuum Mechanics. Macmillan', New York, (1967)

[4] Drew,D.A and Segel, 'L.A. Averaged Equations for Two Phase Flows', *Studies in Appl. Math.* Vol. L, 3, (1971)

[5] Drew,D.A and Segel, 'L.A. Averaged Equations for Two Phase Flows', *Studies in Appl. Math.* Vol. L, 2, (1971)