Point Cloud-Based Target-Oriented 3D Path Planning for UAVs

ICUAS 2020

Zhaoliang Zheng
Electrical and Computer Engineering
LEMUR
UC Los Angeles

Motivation

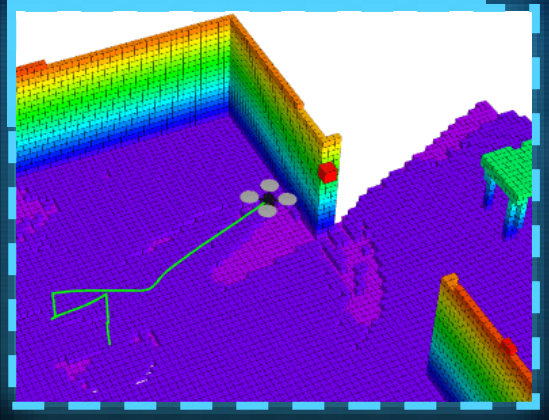
- UAVs play a very important in many areas.
- Path planning is a key technique in better performance of those tasks.
- Most of the work implement their algorithm mainly in Lab or grid maps.
- In the real world, people use ptcloud map to record and represent our world.
- Is it 3D path planning algoritpossible to develop a hm in ptcloud map?

Point Cloud Map Representation

01 Grid Map

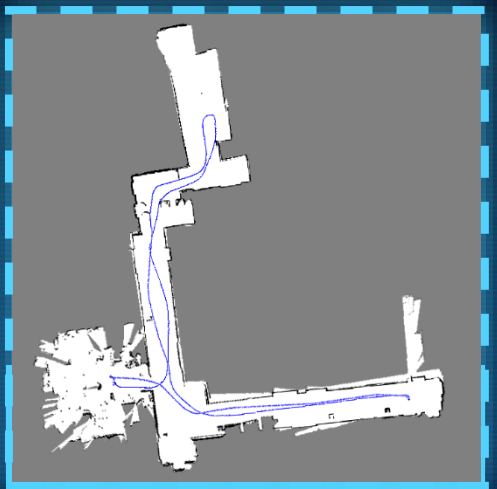
[1] PythonRobotics Github

02 OctoMap



[2] Wang, Chaoqun & Meng, Lili & Li, Teng & W. De Silva, Clarence & Q.-H. Meng, Max. (2017). Towards autonomous exploration with information potential field in 3D environments.

03 Occupancy map



04 Point Cloud Map

Courtesy of Vid and Eric at Drone Lab@UCSD

Point Cloud Generation

01 Monocular Camera

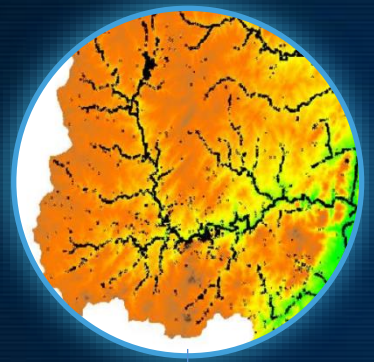
02 Stereo Camera

03 Lidar System

Point Cloud Applications

1

Digital elevation model of the terrain

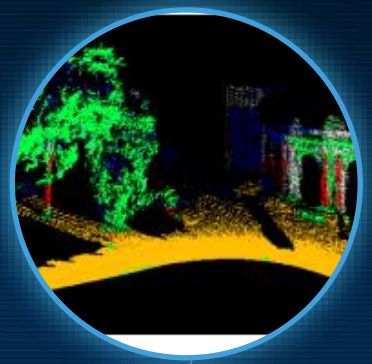


2

Feature Extraction

3

Contextual classifications



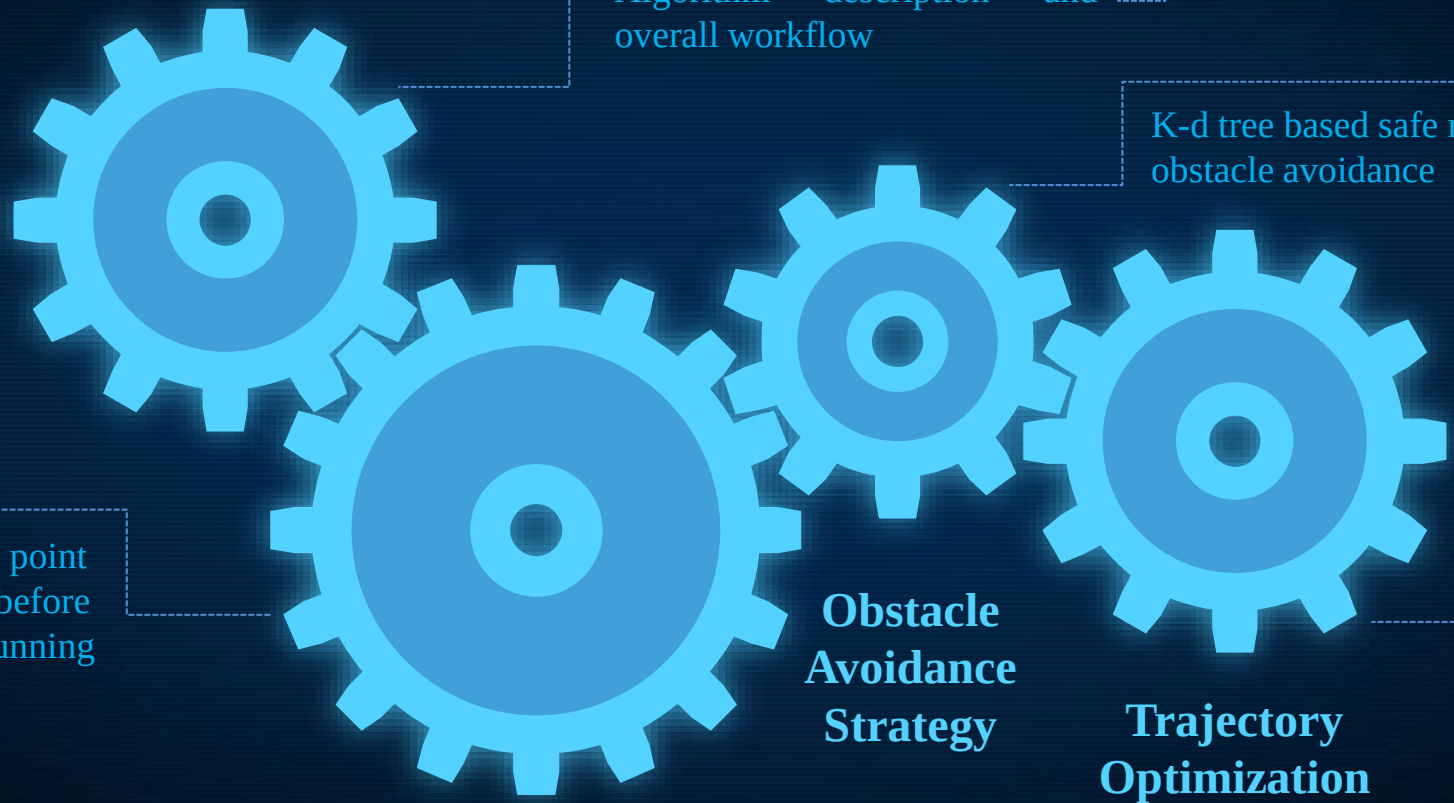
4

3D re-construct cultural heritages

Target-Oriented 3D RRT Algorithm

Algorithm Design & Workflow

Algorithm description and overall workflow



K-d tree based safe region obstacle avoidance

Analysis to point cloud map before algorithm running

Point Cloud Map Analysis

Obstacle Avoidance Strategy

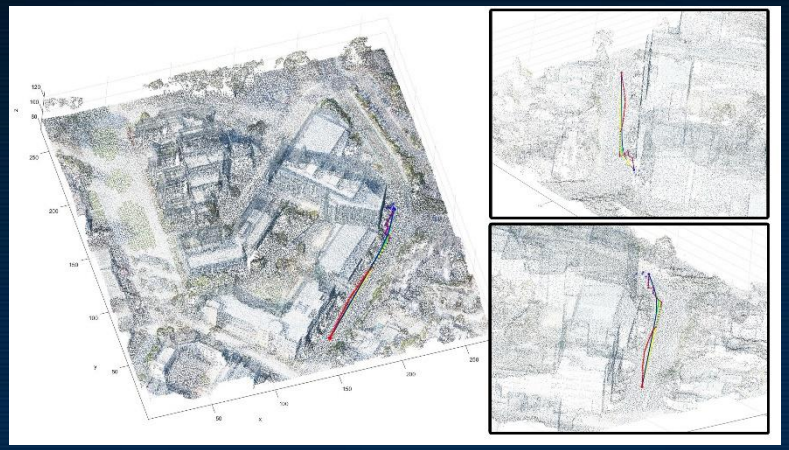
Trajectory Optimization

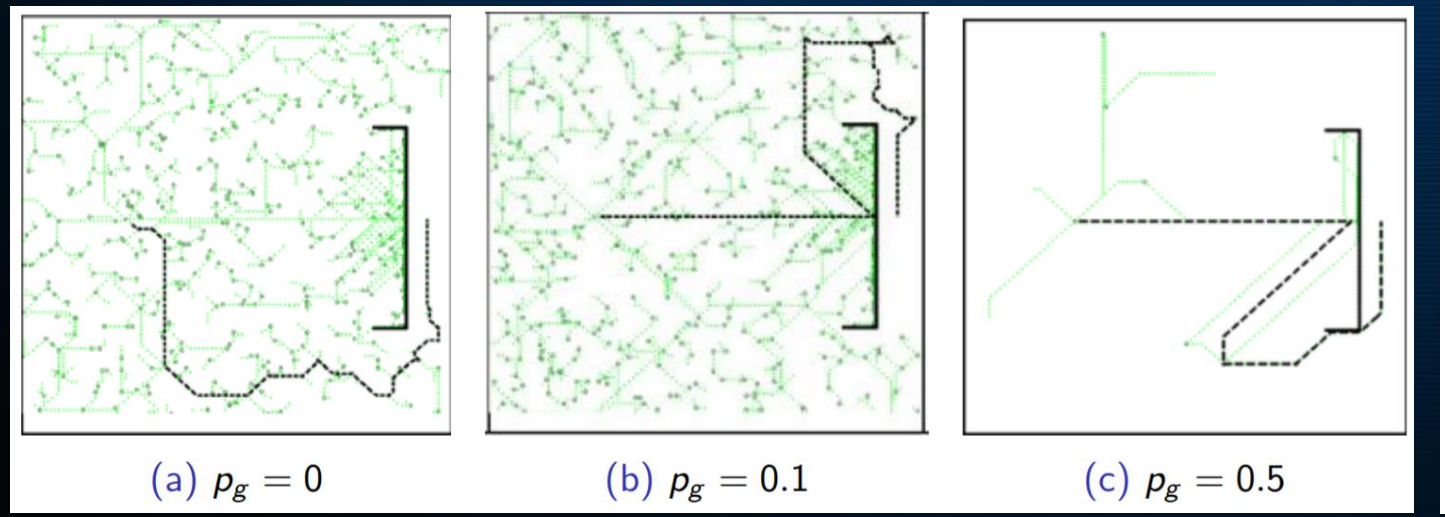
Down sample optimization
Up sample optimization
B-spline Curve Optimization

Algorithm Design & Workflow

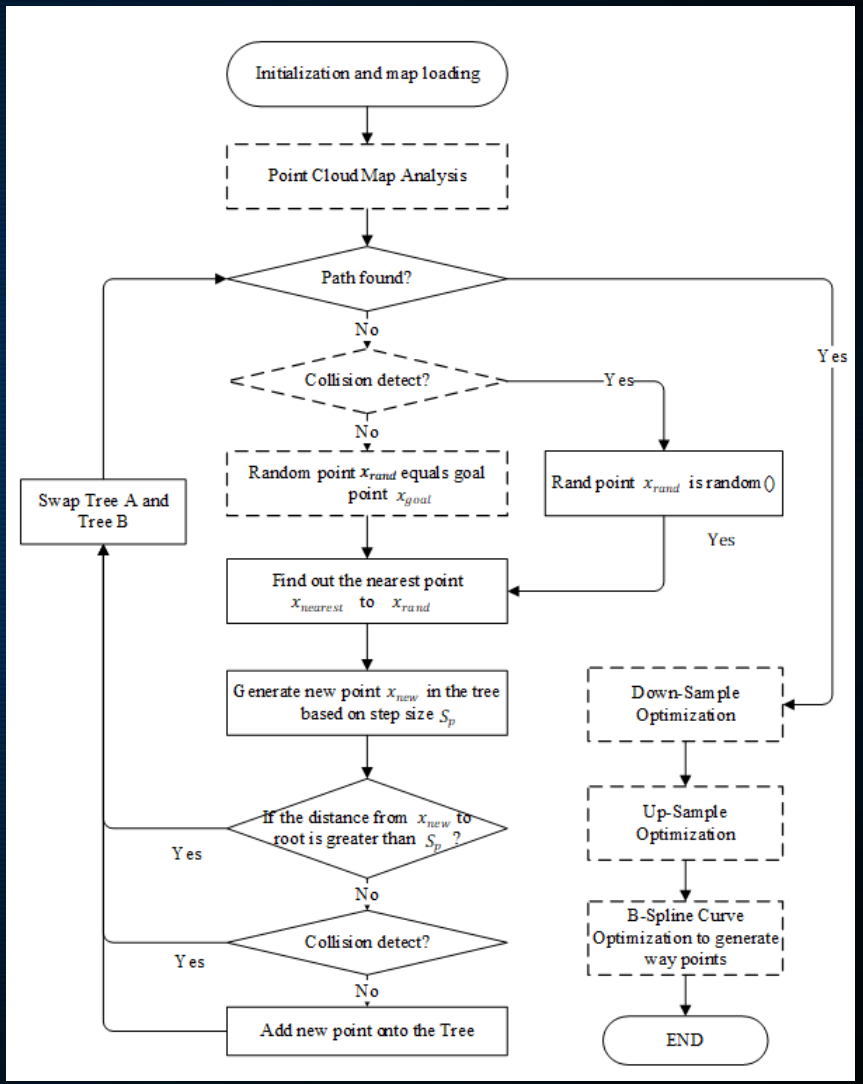
Algorithm 2 EXTEND(T, x)

$FLAG \leftarrow 0$; $i \leftarrow 0$;
if COLLISIONDETECT = NOCOLLISION **then**
  $x_{rand} \leftarrow x$
else
  $x_{rand} \leftarrow$ Sample(i)
end if
$x_{nearest} \leftarrow$ NEARESTNODE(T, x)
$x_{new} \leftarrow$ STEER $(x_{nearest}, x)$
if COLLISIONDETECT = NOCOLLISION **then**
  $V \leftarrow \{x_{new}\}$; $E \leftarrow \{x_{new}, x_{near}\}$
  T $\leftarrow$ {V, E}
  if $|x_{new} - x| < StepSize$ **then**
    return FLAG = 1
  end if
end if
return T, FLAG





(a) $p_g = 0$ (b) $p_g = 0.1$ (c) $p_g = 0.5$

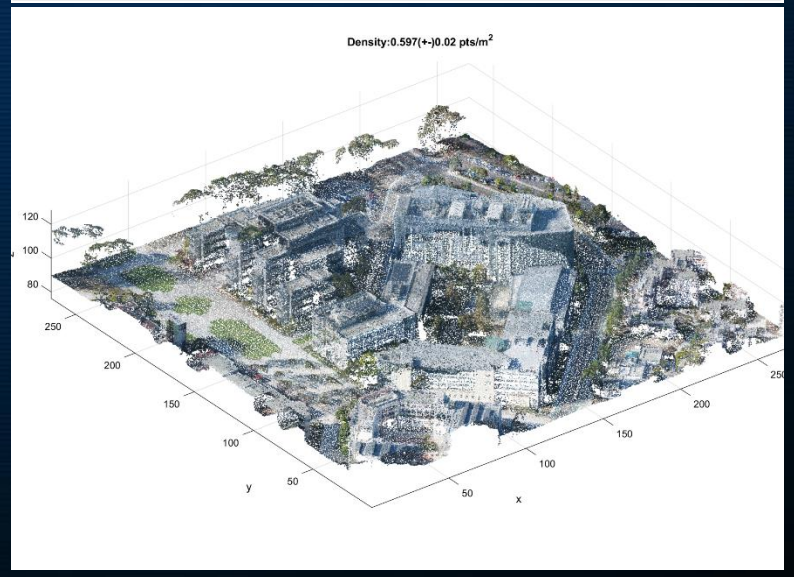


Point Cloud Map Analysis

Density level	δ (pts/m²)	Applications
Sparse point clouds	[0.5,1)	Normally collected for large scale digital height models.
Low density point clouds	[1,2)	For flood modelling applications.
Medium density point clouds	[2,5)	Suitable for satisfying most usages.
High density point clouds	[5,10)	For capturing the details of buildings.
Extremely dense point clouds	[10,∞)	For capturing more and all the details.

Algorithm 3 PointCloudAnalysis($ptClMap$)

```
subdata ← 1%RandomSample(ptClMap);
K ← {NeighborNumber}; r ← {radius};
for i =1...subdata.Size() do
    dists ← FindNN(subdata(i),ptClMap,K)
    dists_min(i) ← Min(dists);
    index ← FindNR(subdata(i),ptClMap,r)
    index_number(i) ← index.Size();
end for
ptClDensity ←SurfDensity(index_number, r)
if ptClDensity ≤ LowerBound then
    return Fail
else
    StepSize ← SA(dists_min)
    CSpace ← CSA(ptClMap)
    return StepSize,CSpace
end if
```

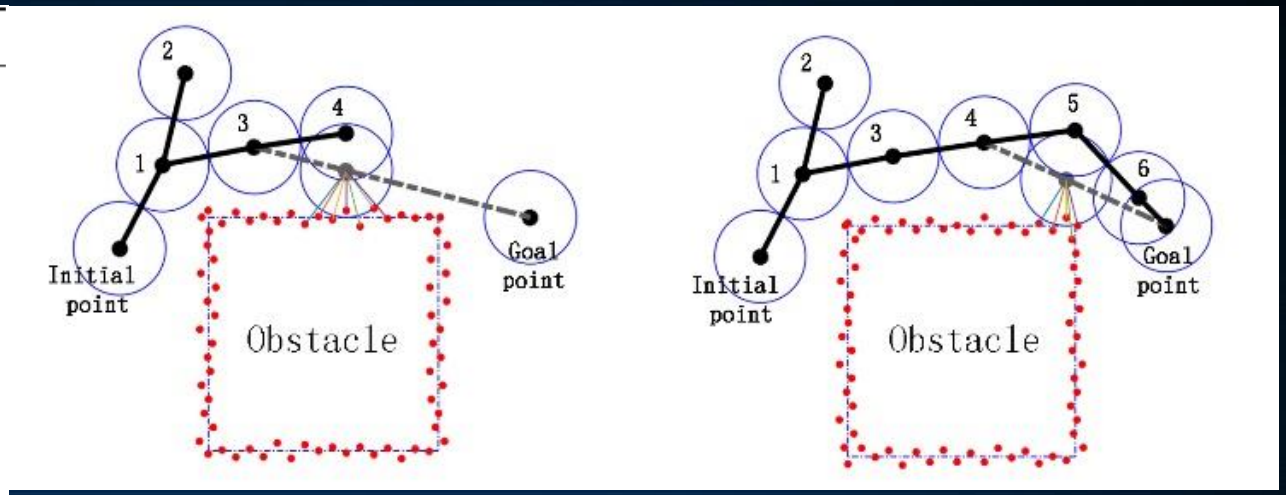


Obstacle Avoidance Strategy

Algorithm 4 CollisionDetect($x_{curr}, x_{next}, ptClMap$)

```
Status ← NoCollision;
if OutofRange(x_curr, ptClMap) = True then
    return Status ← Collision
end if
x_mid ← Interp (x_curr, x_next)
if CheckPoint(x_mid, ptClMap)=Hit then
    return Status ← Collision
end if
return Status
function CheckPoint(x_mid, ptClMap)
    SafeDist ← {S};
    for i = 1...K do
        if FindNN(x_mid, ptClMap, i) ≤ S then
            return Hit
        end if
    end for
end function
```

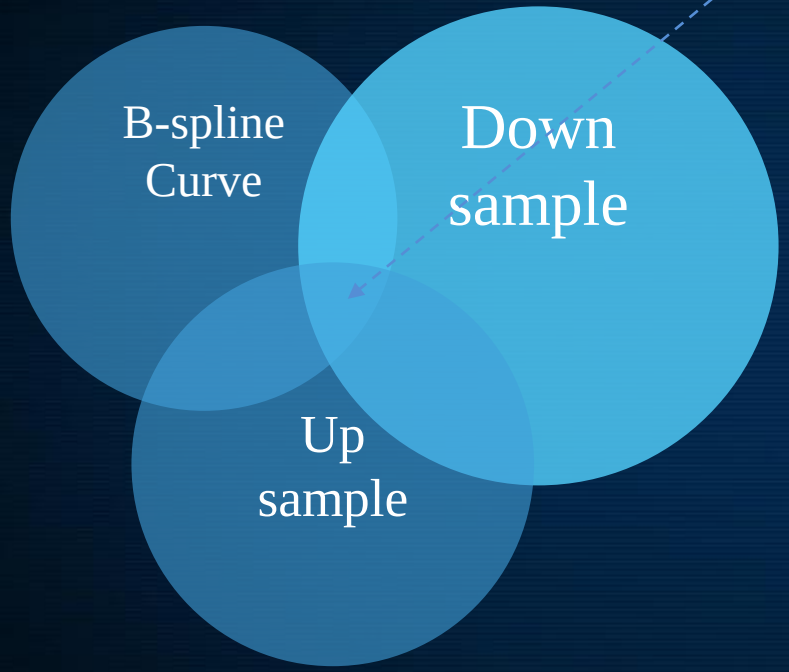
$$S = (0.5 \sim 0.8) \times StepSize \quad (1)$$



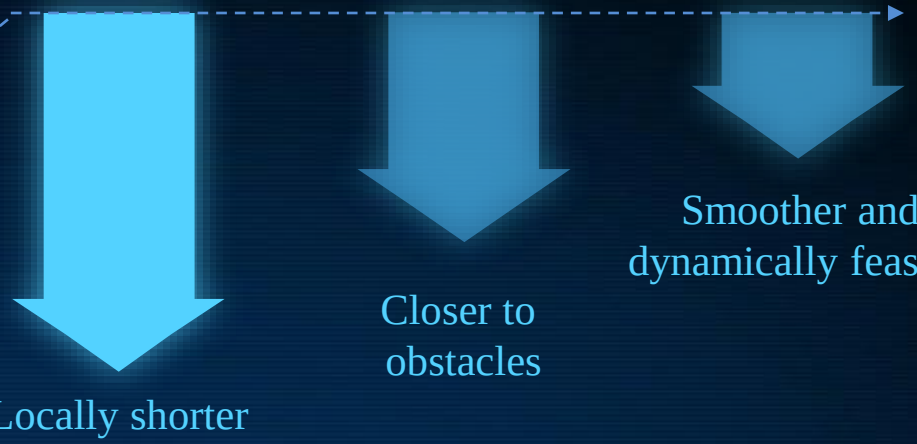
(a) (b)

Kd-Trees is a binary data structure inveted in 1970s by Jon Bentley [38]. Kd-Trees algorithm can separate into two parts[39]: first part is the algorithm of constructing the kd-tree; second part is algorithm of searching for the nearest neighbor. According to the complexity analysis in [39], the time complexity of kd-trees nearest neighbor algorithm is O(n^(1-1/d)+k), where d is the d-dimension and k is the k nearest neighbors need to retrieve.

Trajectory Optimization



Three-step Optimization



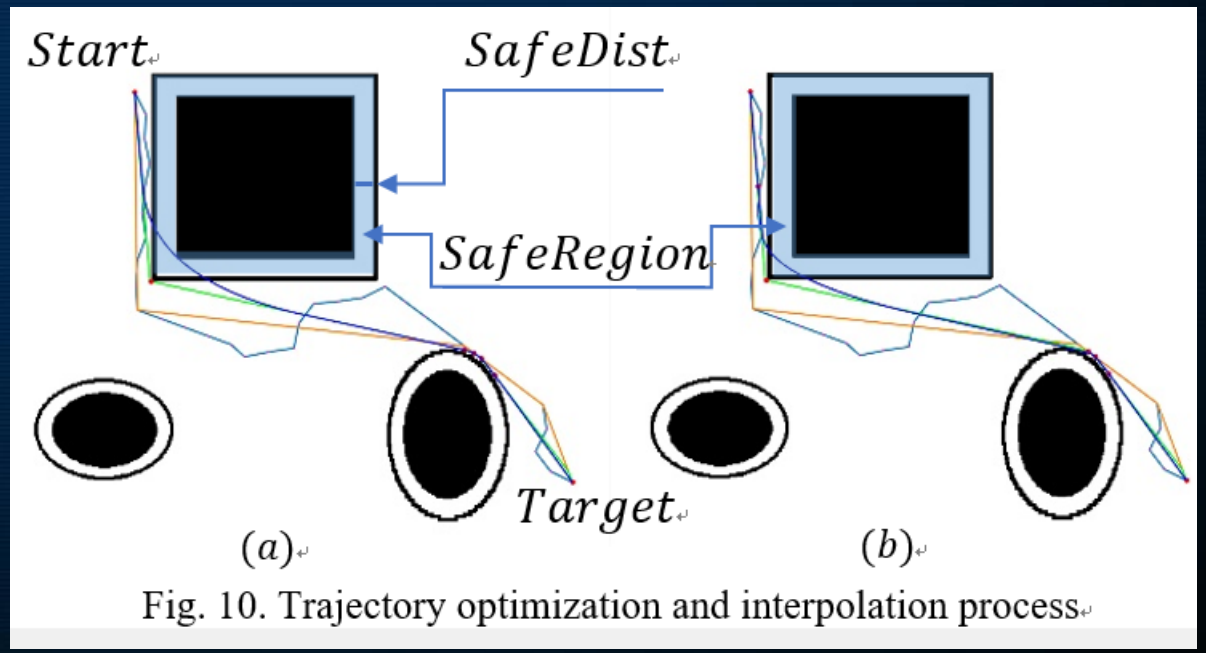
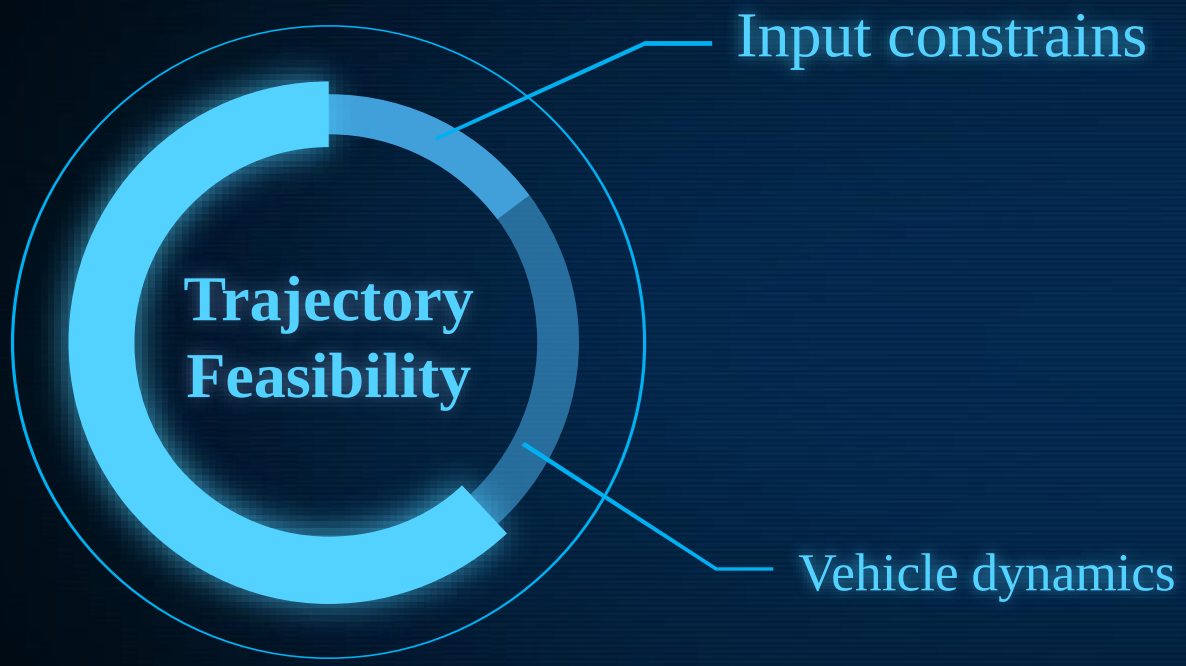


Fig. 10. Trajectory optimization and interpolation process

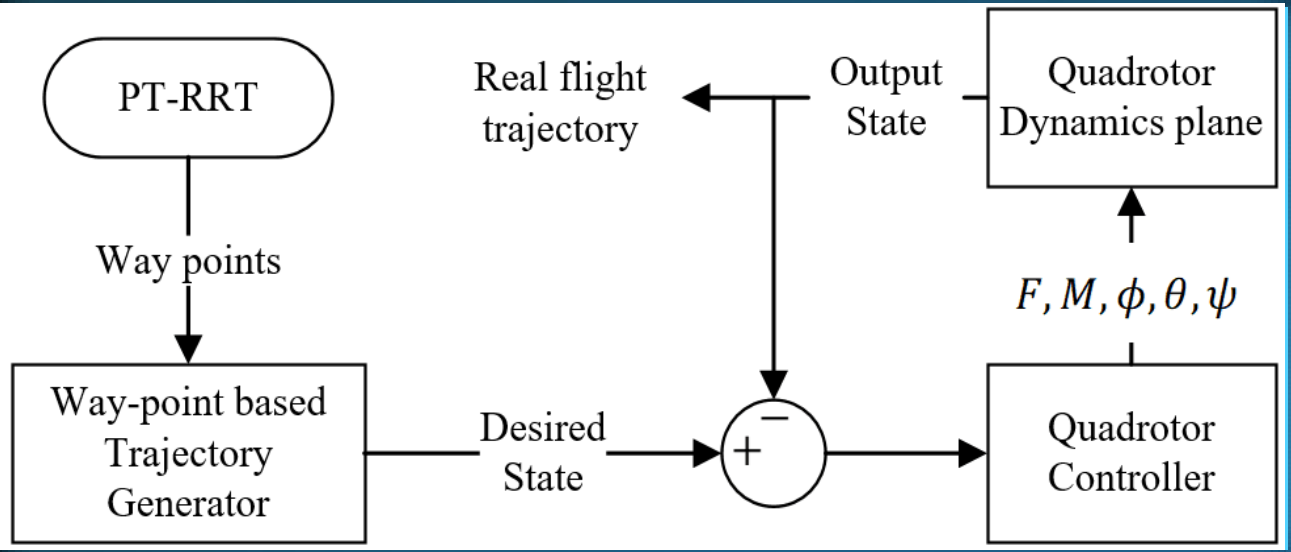
Way-point based Closed Loop Quadrotor Control



According to the method in [42], a quadrotor system has the differential flatness property, makes it possible to reduce optimal trajectories to a sequence of 3-D position and yaw angle and their derivatives. And also, as presented by Hehn et al. [43], trajectory feasibility constraints includes vehicle dynamics and input constraints, in which control inputs can be calculated from the generated trajectory. In this section, we adapt ideas from these two papers and design the way-point based closed-loop control.

[42] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” Proc. - IEEE Int. Conf. Robot. Autom., pp. 2520–2525, 2011.
[43] M. Hehn and R. D’Andrea, “Quadrocopter trajectory generation and control,” IFAC Proc. Vol., vol. 44, no. 1 PART 1, pp. 1485–1491, 2011.

Way-point based Closed Loop Control



The output state from quadrotor dynamics is the quadrotor state $x_{state} \in \mathbb{R}^{12}$, given in (6). In order to make the dimensions consistent, both Tra_{state} and x_{state} have to transform into a middle quadrotor state $Q_{state} \in \mathbb{R}^{13}$ before feeding in the quadrotor controller, given in (7).

$$Tra_{state} = [p_{xyz} \quad v_{xyz} \quad a_{xyz} \quad \psi \quad \dot{\psi}]^{T} \quad (5)$$

$$x_{state} = [p_{xyz} \quad v_{xyz} \quad q_{wxyz} \quad \omega_{pqr}]^{T} \quad (6)$$

$$Q_{state} = [p_{xyz} \quad v_{xyz} \quad el_{xyz} \quad \omega_{pqr}]^{T} \quad (7)$$

Where,

$p_{xyz} = [x \quad y \quad z]$, is the position in world coordinates;

$v_{xyz} = [\dot{x} \quad \dot{y} \quad \dot{z}]$, is the velocity in world coordinates;

$a_{xyz} = [\ddot{x} \quad \ddot{y} \quad \ddot{z}]$, is the acceleration in world coordinates;

$el_{xyz} = [\phi \quad \theta \quad \psi]$, is the Euler roll, pitch and yaw;

$\omega_{pqr} = [p \quad q \quad r]$, is the angular velocity around body xyz-axis;

$q_{wxyz} = [q_w \quad q_x \quad q_y \quad q_r]$, is the quaternion.

Quadrotor Controller

The inputs of controller are desired quadrotor middle state Q_{state}^{des} and current quadrotor middle state Q_{state}. The desired quadrotor middle state Q_{state}^{des} is calculated based on desired trajectory state Tra_{state}. The current quadrotor middle state Q_{state} is calculated from the quadrotor current state $\mathrm{x}_{\mathrm{state}}$. The controller is given in (8),(9),(10),(11).

The outputs of quadrotor controller are $\mathrm{F}, \mathrm{M}, \phi, \theta, \psi$ and $\mathrm{x}_{\mathrm{state}}$. These outputs will be feeding into the quadrotor dynamics plane to solve for the state space equations.

$$a_{xyz} = a_{xyz}^{des} + K_v\left(v_{xyz}^{des} - v_{xyz}\right) + K_p\left(p_{xyz}^{des} - p_{xyz}\right) \tag{8}$$

$$\begin{cases} \phi = \frac{1}{g}\left(a_x sin\,\psi^{des} - a_y \cos\psi^{des}\right) \\ \theta = \frac{1}{g}\left(a_x\, cos\,\psi^{des} + a_y\, sin\,\psi^{des}\right) \\ \psi = \psi^{\mathrm{des}} \end{cases} \tag{9}$$

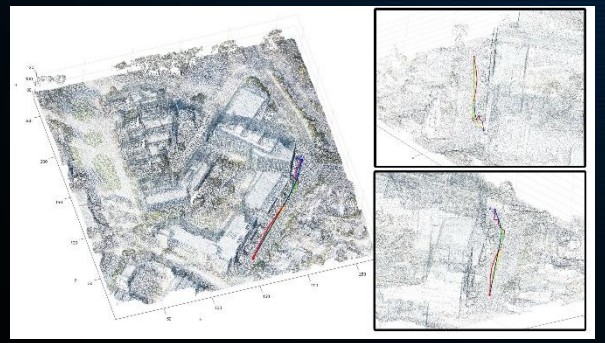
$$F = m(g + a_z) \tag{10}$$

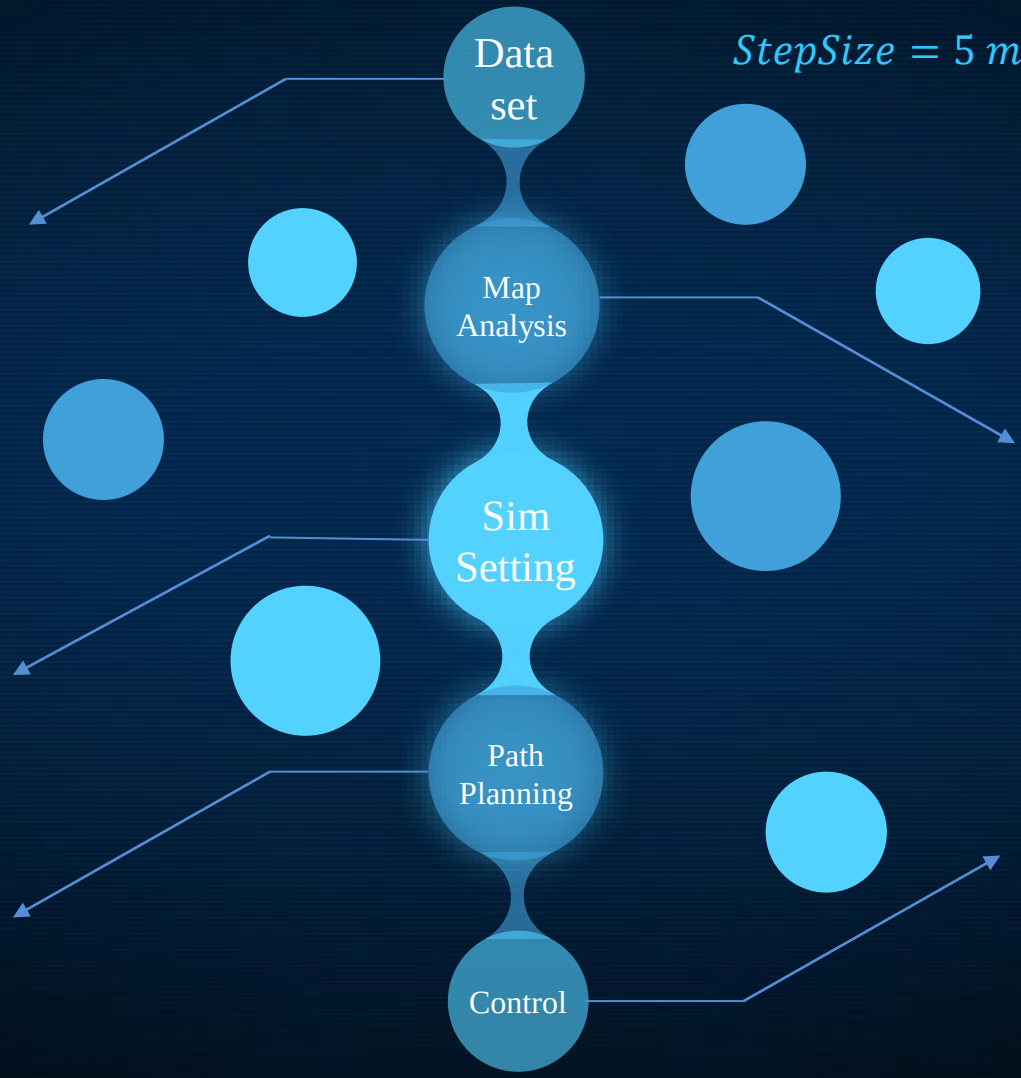
$$\mathrm{M} = \mathrm{I}_{\mathrm{xyz}}\left(K_{vm}\left(\omega_{pqr}^{des} - \omega_{pqr}\right) + K_{pm}\left(el^{des} - el\right)\right) \tag{11}$$

Simulations and Result Discussion

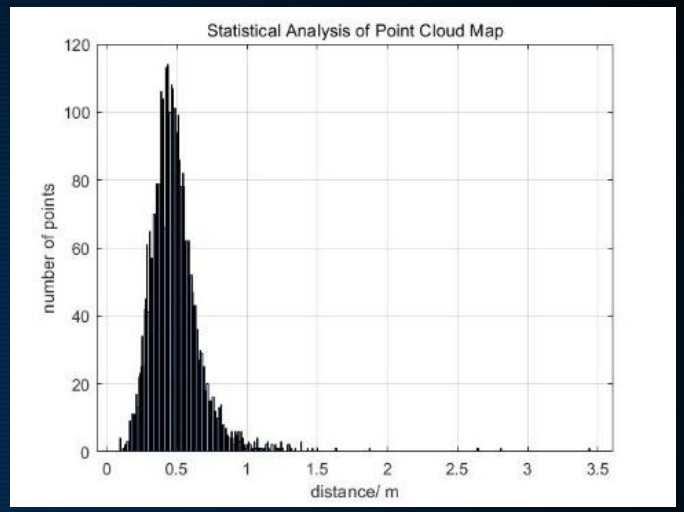
Datatset	Jacob School of Engineering@UCSD
3D model .ply format size	10.8 MB
Number of points	405934
Point cloud density	$0.597 \pm 0.01\ pts/m^2$
Actual size	$(270 \times 270 \times 52)$m

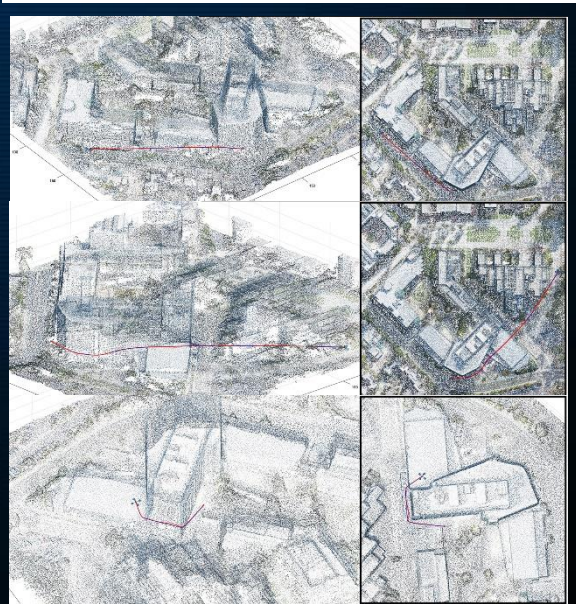
No.	Description		
	Initial point	Goal point	Verification
1	(230,120,120)	(150,30,90)	Target orientation
2	(230,120,120)	(90,225,95)	Up-Sample Optimization
3	(158,150,96)	(175,185,100)	B-Spline Optimization



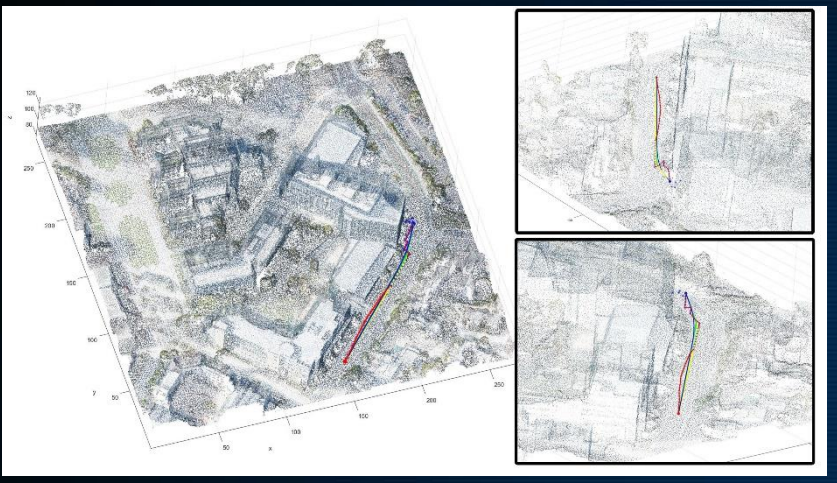


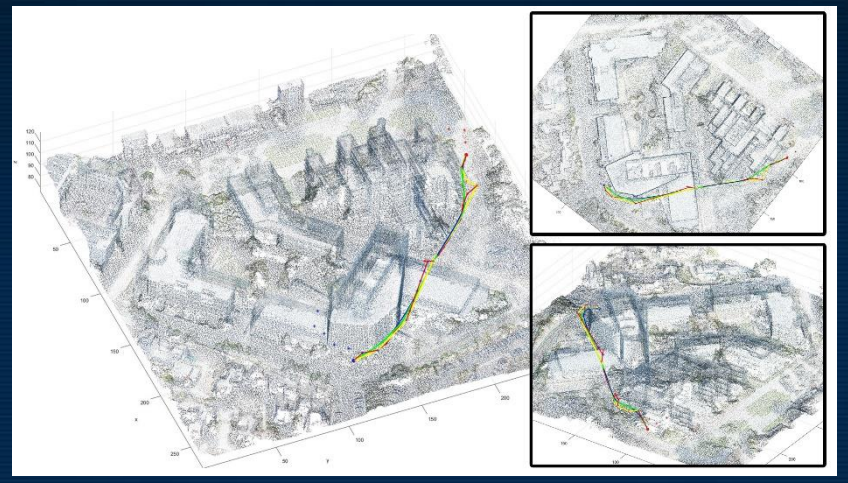
$StepSize = 5\ m$, and safety distance: $SafeDist = 3\ m$

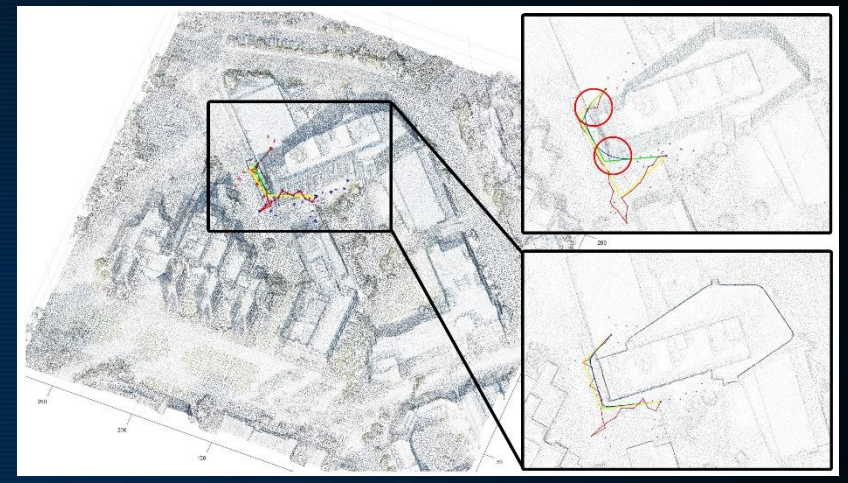


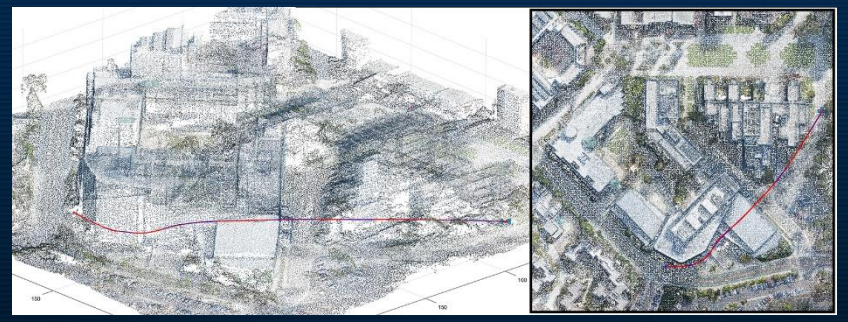


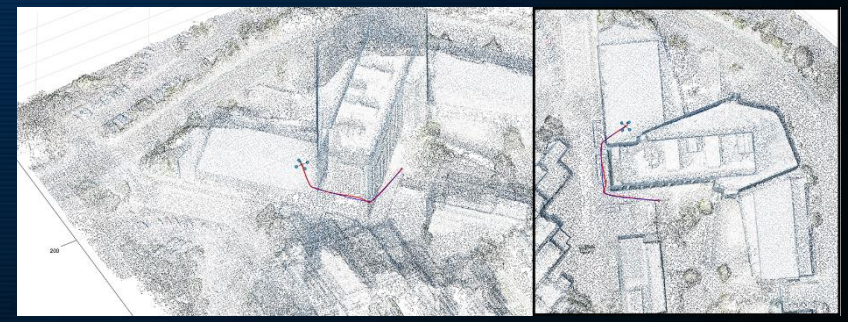
Simulations and Result Discussion

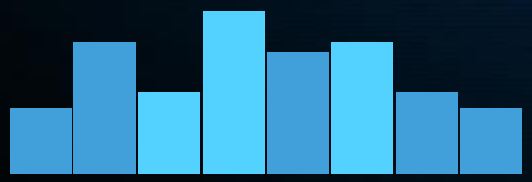












https://youtu.be/wdU3QXmmEFg

Plan for JINT Journal Paper

- Compress my ICUAS paper to 30%
- Enhance algorithm part: talk more about its theoretical basis, concepts and 2D performance. (20%)
- Enhance the quadrotor UAV control part: talk more about the way point based control and the inner loop control. (5%)
- Enhance the experiment part: more tests on different point cloud maps (20%)
- Enhance the discussion part and include the comparison: (25%)

Thanks for your listening

Q&A? Comments?