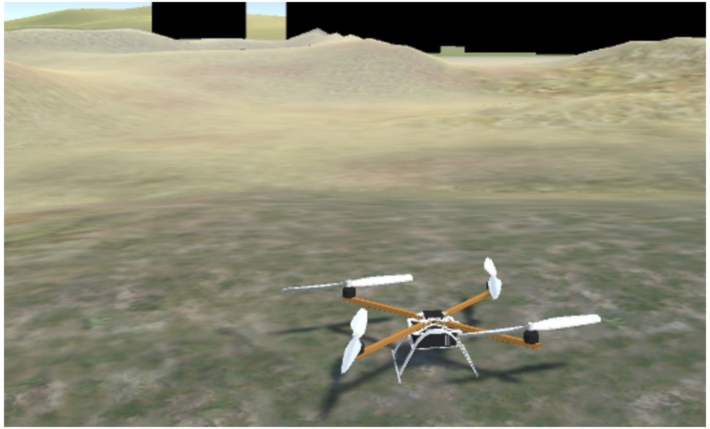
Challenge 3 - Area Mesh Generation

Group AMG: Student Presentation

Tutor: Prof.Dr.Markus Ryll

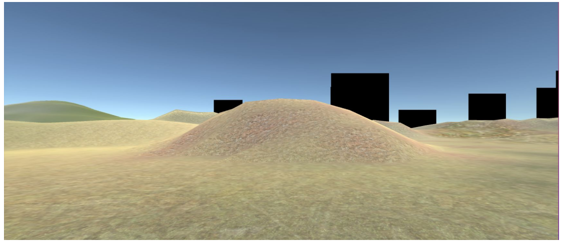


Presenter: Hang Li, Linya Ruan, Ziting Huang, Tiantian Wei

Date: 22.03.2023

Contents

Project Introduction



- **Unity environment:** slightly hilly terrain containing a polygon of n corners

- **Hardware:** drone equipped with a set of down-pointing stereo RGB Cameras

- **Goal:** Mesh Generation from the matching images captured by cameras

Task Overview

- trajectory generation
- perform feature-matching
- point-cloud generation use stereo_image_proc package
- point-cloud from features generation
- point cloud merging
- point cloud preprocessing
- mesh Generation from point cloud
- add color information to mesh

Task Overview

- **trajectory generation**
- perform feature-matching
- point-cloud generation use stereo_image_proc package
- point-cloud from features generation
- point cloud merging
- point cloud preprocessing
- mesh Generation from point cloud
- add color information to mesh

trajectory Generation

- ## requirements :

➢ Cover the terrain areas with distinct features.

➢ Minimize overlap of the RGB images captured.

➢ Prevent drone shake at each target point from affecting image quality by pausing for a few seconds at each point to capture the images.

trajectory Generation

- input parameters

Set maximum velocity and acceleration

```
BasicPlanner::BasicPlanner(){
    max_v_ = 5;
    max_a_ = 5;
} //
```

Set fixed height and goal positions

```
posx.push_back(-17);
posy.push_back(15);
posz.push_back(12);

posx.push_back(-32);
posy.push_back(15);
posz.push_back(12);
```

trajectory Generation

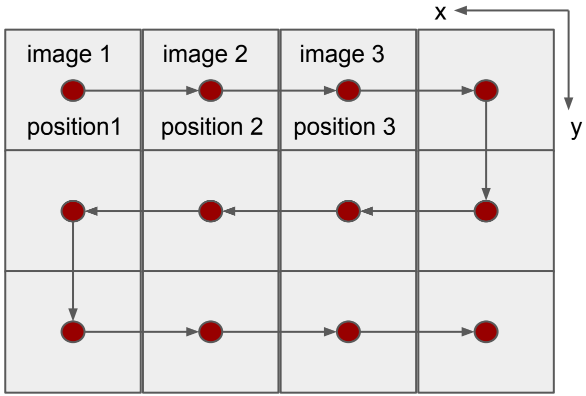
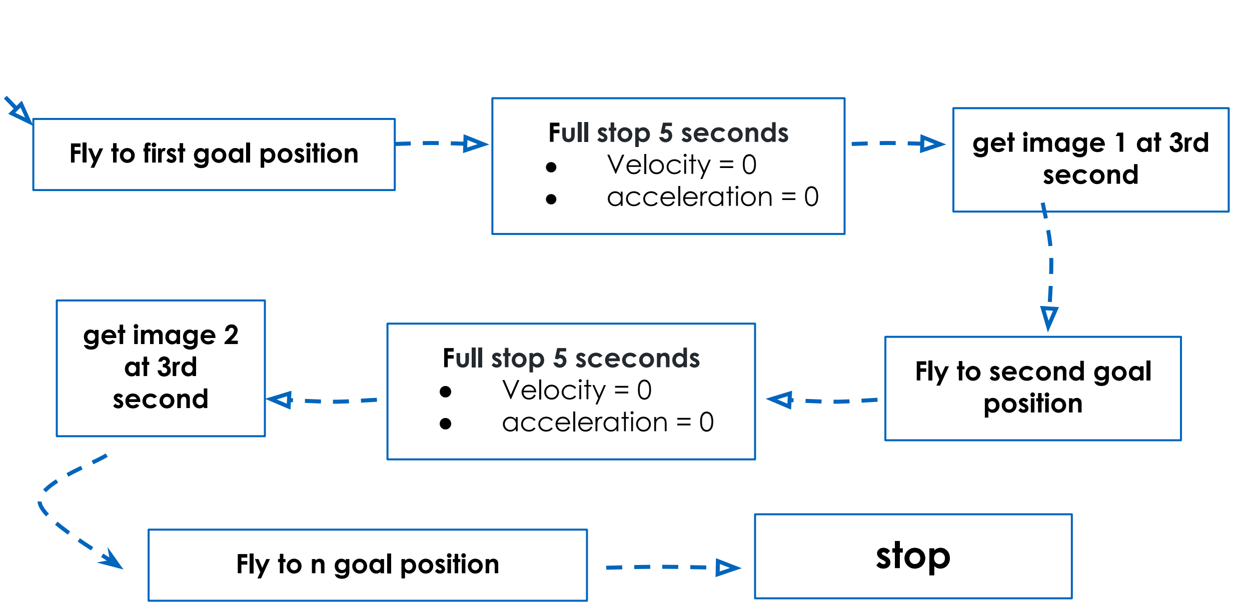


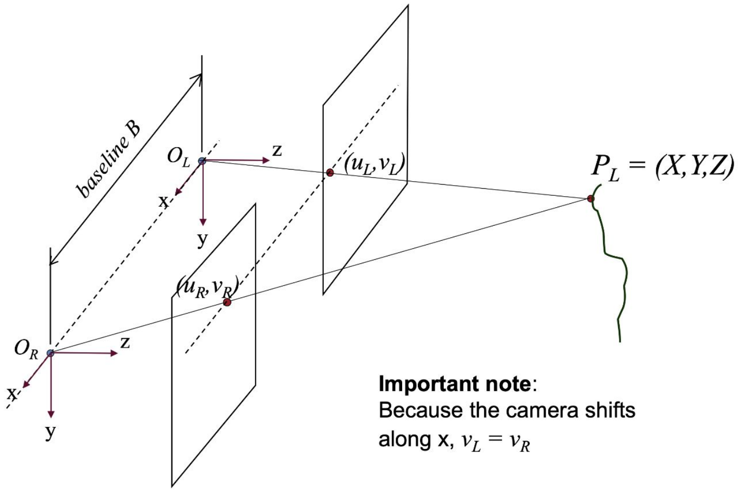
Fig.1. Path planning



Task Overview

- trajectory generation
- **perform feature-matching**
- point-cloud generation use stereo_image_proc package
- point-cloud from features generation
- point cloud merging
- point cloud preprocessing
- mesh Generation from point cloud
- add color information to mesh

perform feature-matching



Important note:
Because the camera shifts along x, $v_L = v_R$

Fig.2. Principle of Binocular Stereo Imaging

$$u_l = f \cdot \frac{X}{Z}$$

$$u_r = f \cdot \frac{(X - B)}{Z}$$

$$u_l - u_r = f \cdot \frac{b}{Z}$$

$$Z = \frac{B \cdot f}{(u_l - u_r)}$$

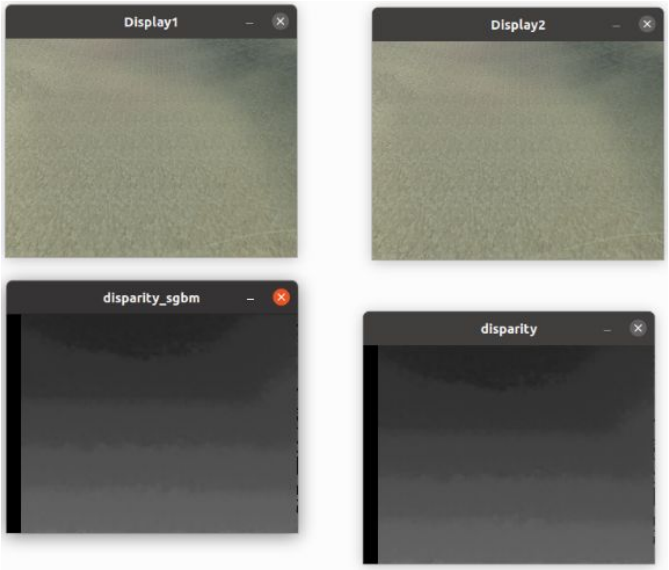


Fig.3. binocular disparity map

perform feature-matching

```
header:
  seq: 214116
  stamp:
    secs: 8388
    nsecs: 720000000
  frame_id: "Quadrotor/Sensors/DepthCamera"
height: 240
width: 320
distortion_model: "plumb_bob"
D: [0.0, 0.0, 0.0, 0.0, 0.0]
K: [120.00000000000001, 0.0, 160.0, 0.0, 120.00000000000001, 120.0, 0.0, 0.0, 1.0]
R: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
P: [120.00000000000001, 0.0, 160.0, 0.0, 0.0, 120.00000000000001, 120.0, 0.0, 0.0, 0.0, 1.0, 0.0]
binning_x: 0
binning_y: 0
roi:
  x_offset: 0
  y_offset: 0
  height: 0
  width: 0
  do_rectify: False
---
```

RGB Camera (Script)

Script	RGBCamera
Output Image	None (Render Texture)
Camera	None (Camera)
Width	320
Height	240
Field Of View	90
Far Clip Plane	65.535
Target Frame Rate	30
Monochrome	

$$width = 320$$

$$height = 240$$

$$FOV = 60$$

$$cx = width/2 = 160$$

$$cy = height/2 = 120$$

$$fx = 0.5 * width/\tan(FOV/2)$$

$$fx = fy = 160$$

$$b = 0.2$$

perform feature-matching

- **Problem:**

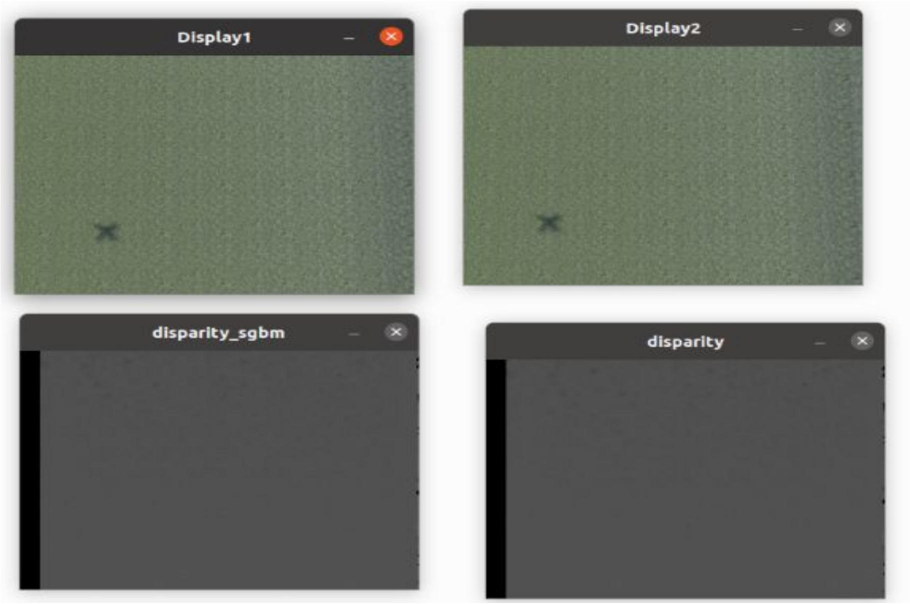


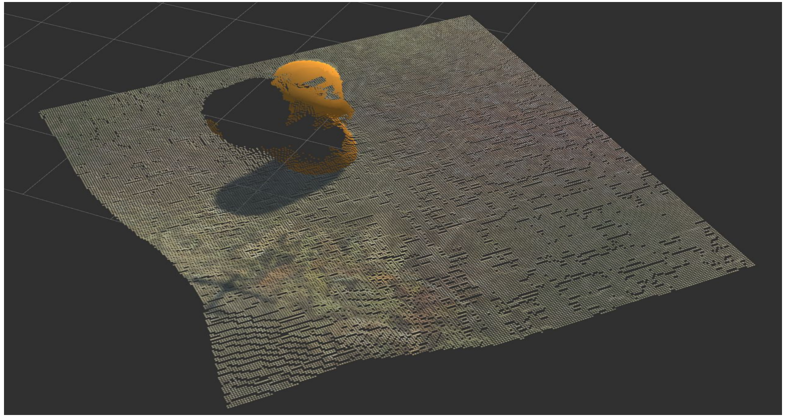
Fig.1. problem of binocular disparity map in a featureless environment

Task Overview

- trajectory generation
- perform feature-matching
- **point-cloud generation use stereo_image_proc package**
- point-cloud from features generation
- point cloud merging
- point cloud preprocessing
- mesh Generation from point cloud
- add color information to mesh

point-cloud generation use stereo_image_proc package

Input left and right images → Preprocess the images → Perform stereo matching → Calculate depth information → Generate point cloud or depth map



Task Overview

- trajectory generation
- perform feature-matching
- point-cloud generation use stereo_image_proc package
- point-cloud from features generation
- **point cloud merging**
- point cloud preprocessing
- mesh Generation from point cloud
- add color information to mesh

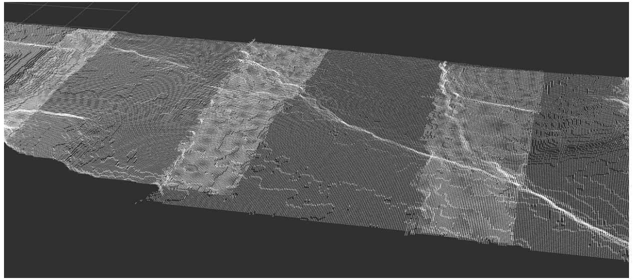
point cloud merging

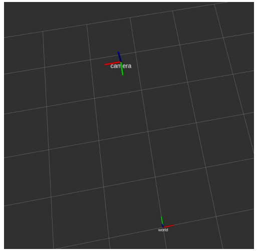
1. Transform point cloud to world frame

➢ `listener.lookupTransform` obtain transformation matrix.

➢ `pcl_ros::transformPointCloud` conduct transformation.

2. Sum the point cloud in world frame





Task Overview

- trajectory generation
- perform feature-matching
- point-cloud generation use stereo_image_proc package
- point-cloud from features generation
- point cloud merging
- **point cloud preprocessing**
- mesh Generation from point cloud
- add color information to mesh

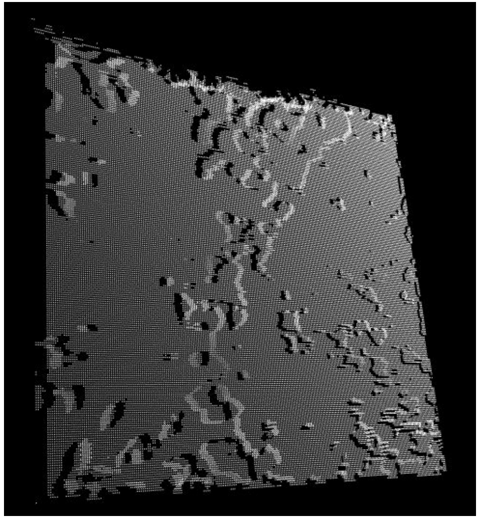
point cloud preprocessing

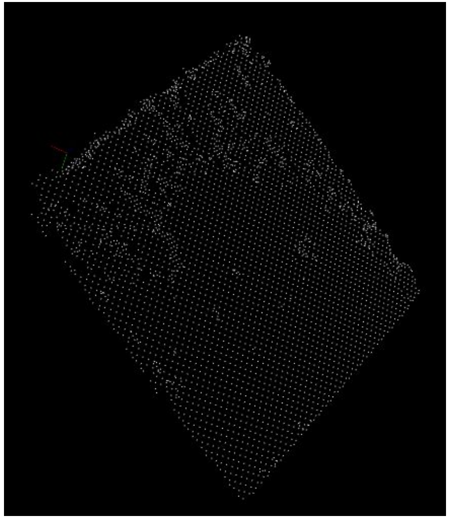
1. Downsampling:

➢ reduce the number of points
➢ while keeping the essential features

2. Filtering:

➢ outliers filter





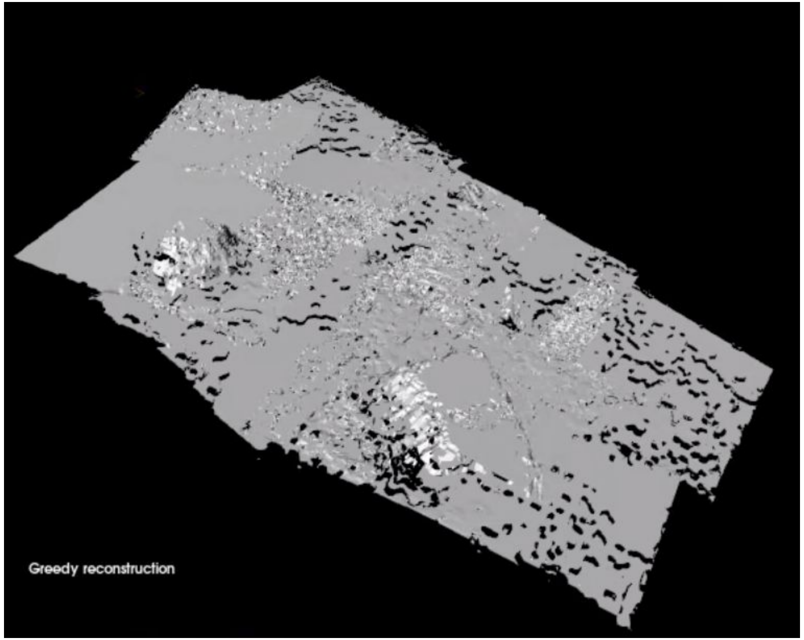
Task Overview

- trajectory generation
- perform feature-matching
- point-cloud generation use stereo_image_proc package
- point-cloud from features generation
- point cloud merging
- point cloud preprocessing
- **mesh Generation from point cloud**
- add color information to mesh

mesh generation from point cloud

Greedy triangle reconstruction

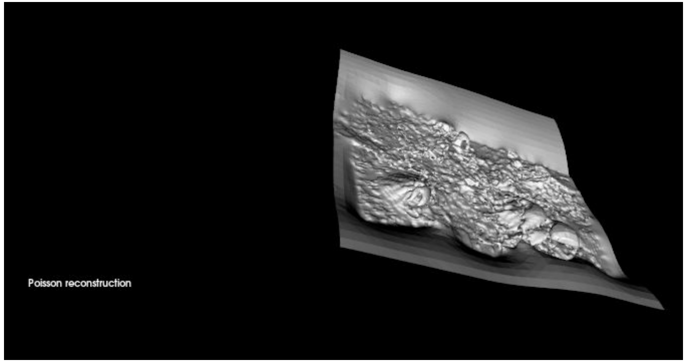
- Construct triangles iteratively between nearest neighboring point in the cloud
- Advantages:
  - Simplicity
  - fast
- Disadvantages:
  - effect by noise



mesh generation from point cloud

Poisson surface reconstruction

- construct a scalar function over the input points to generate a continuous surface representation through a process of implicit surface reconstruction
- Advantages:
  - Handling noisy data, smooth surface
  - Filling in missing data
  - mesh **without any holes**
- Disadvantages:
  - computational expensive



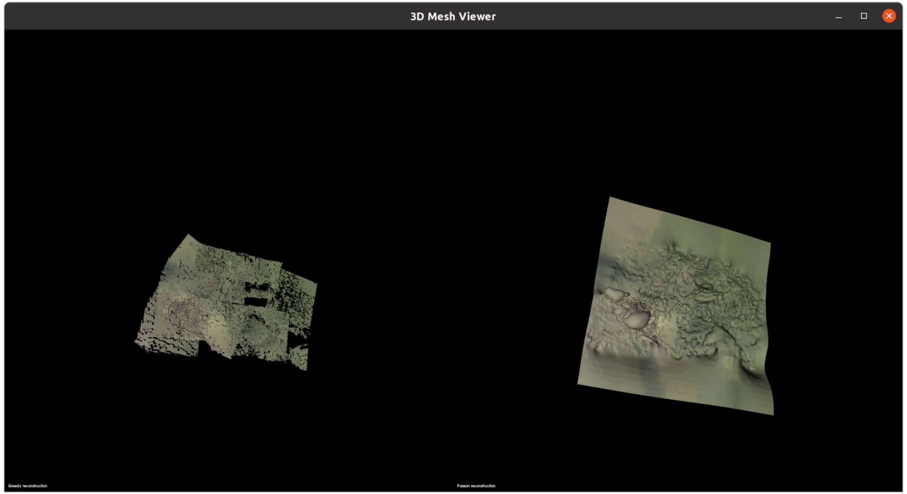
Task Overview

- trajectory generation
- perform feature-matching
- point-cloud generation use stereo_image_proc package
- point-cloud from features generation
- point cloud merging
- point cloud preprocessing
- mesh Generation from point cloud
- **add color information to mesh**

add color information to mesh

Color from RGB Point Cloud

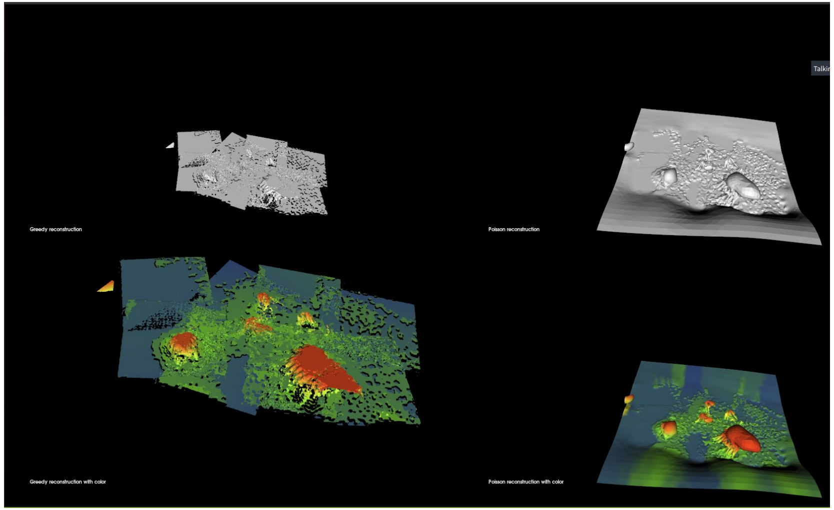
➢ Assigning each vertex of the mesh the RGB value of the closest point in the **original point cloud**



add color information to mesh

Height Visulization

- ➢ Assigning distinct RGB values based on height.
- ➢ blue: low -> red: high



Demo

Conclusion and Futurework

Biggest Problem: Fusion of Point Cloud

- ➢ Reasons might be : Imprecise of TF transform, noise, unstable

Can not do Point Cloud Register because no strong features

- ➢ Test on other unity environment

Reference

[1] https://johnwlambert.github.io/stereo/

[2] http://docs.ros.org/en/melodic/api/sensor_msgs/html/msg/CameraInfo.html

[3] http://wiki.ros.org/stereo_image_proc

[4] https://learnopencv.com/depth-perception-using-stereo-camera-python-c/

[5] http://wiki.ros.org/mesh_tools

[6]http://wiki.ros.org/stereo_image_proc#stereo_image_proc.2Fdiamondback.stereo_image_proc.2Fpoint_cloud2

Thank you!