



# Livox SLAMのインストールと 点群の処理方法

---

濱 侃（千葉大学大学院園芸学研究院・助教）

a.hama@chiba-u.jp

松本 祐太郎（千葉大学大学院園芸学研究科）

# 資料の取り扱いについて

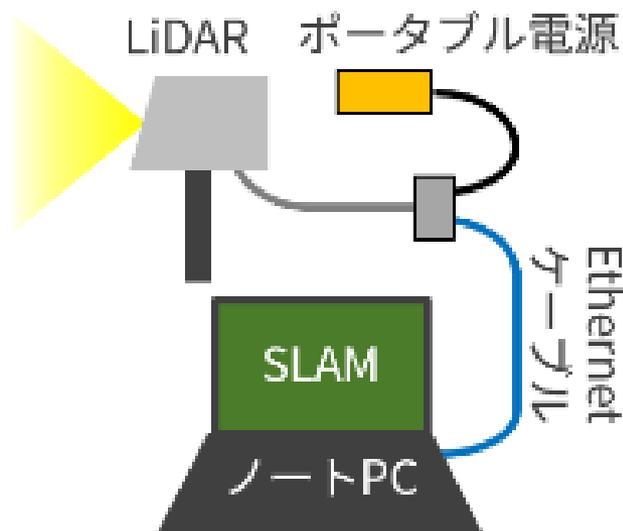
---

- この資料は、第9回低空空撮技術活用研究会の実習  
(2022年2月28日～3月2日) で使用したものです。
- この資料を使用した場合は、論文や報告書等の成果物の中で、その旨を明記してください（本文中で記載、または謝辞に記載）。
- この資料は、The University of Hong Kong Mars-Labがgithubで公開しているSDKを使用しています。

# 実習に必要な機材

---

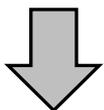
- Livox社のLiDAR (Mid-70, Avia, Horizonなど)
- OSにUbuntu 18.04を使用したPC  
Ubuntuのインストールについては“livox\_mapping\_INSTALL.docx”を参照
- Ethernetケーブル
- ポータブル電源 10 ~ 15 V DC (with Converter 2.0: 9~30V DC)



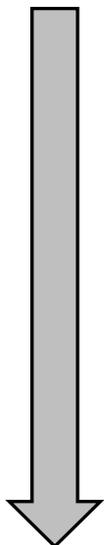
# 実習の流れ

---

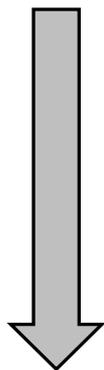
1  
日  
目



2  
日  
目  
A  
M



2  
日  
目  
P  
M



- 導入（イントロ）
- PC環境のセットアップ
- Livox-SDKのインストール
- Livox\_ros\_driverのインストール
- SLAMのインストール
  - ◆ Livox\_mappingのインストール（Midシリーズ）
  - ◆ FAST-LIOのインストール（Avia, Horizon用）
- データの取得テスト
  - ◆ リアルタイムマッピング
  - ◆ 後処理マッピング
- CloudCompareを用いた点群処理

# Livox社の台頭（DJIの子会社）

- 低価格帯（破格の安さ）のLiDARを発表  
➔ テスラの中国ライバル企業であるXpengでも採用



LiDARの価格  
9万円～



# 自動運転とLiDAR

---

- 自動運転用のセンシング技術として注目され、関連技術は急速に進歩している (Li et al., 2020)
- SLAM (Simultaneous Localization And Mapping) の発展
  - ◆ 周囲の3次元地図を自動的に生成するアルゴリズム



LiDARの位置・向きの情報を使用せずに  
3次元地図の作成が可能に

- 低価格化

- ◆ 自動運転用としての需要が高まり、低価格な製品が登場 (Hu et al., 2021)

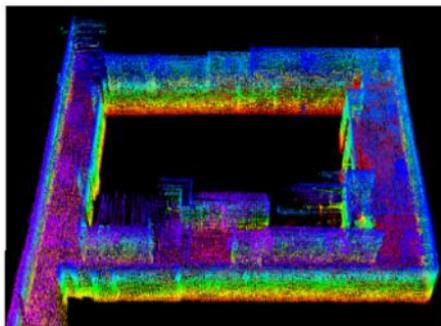


低コストかつ簡便にLiDARを使えるように

LiDARを使用した研究の障壁がなくなりつつある

# 今回使用するSLAMについて

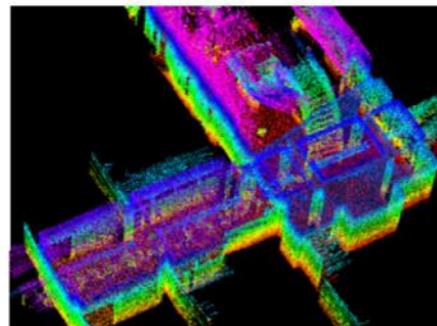
- 元になっている技術：LOAM (Laser Odometry and Mapping)
- ➔ 点群からエッジ (corner) 上の点と平面 (surface) 上の点の特徴点を抽出してレジストレーションに用いる。地図は、2Dや3Dの点群地図やグリッドマップ、ボクセルマップとして表現可能。



(a)



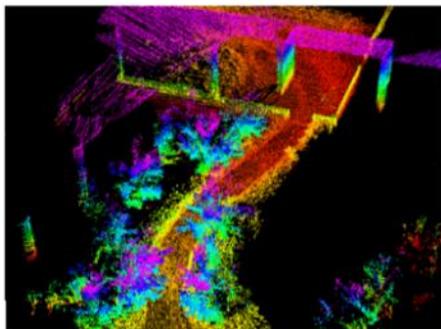
(b)



(c)



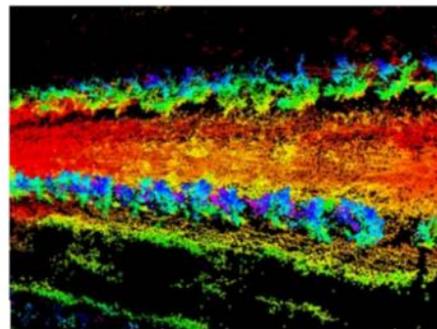
(d)



(e)



(f)



(g)



(h)

# LOAM (Laser Odometry and Mapping)

---

- 点群の密度はフォトグラメトリーなどに比べて粗く，点群同士のマッチングにおいて，特徴が十分でない場合も。。。
  - ◆ 障害物が少ないような場所では移動体の位置を見失ってしまう。
  - ◆ 点群のマッチングは処理負荷が高く，高速化の工夫が必要。
- LOAMでは，移動量の推定 (Odometry) と地図作成 (Mapping) の頻度を変えることでリアルタイム処理に対応。  
※実習でも試してみる
- 位置推定にはIMUなどの他の計測結果を組み合わせ可能。

## Livox-SDKとPCスペック (Livox\_mapping, FAST-LIO)

---

- 今回使用するSDKやCloudCompareは, **CPUが最も重要**。  
もちろんメモリ (RAM) も大切
- マッピングで生成される点群の量(密度)はPCの性能が高いほど多くなる。  
例) 同じBAGファイル (生データ) でマッピング
  1. ノートPC(Core i5 7300U)で実行 → PCDファイルサイズ150MB
  2. デスクトップPC(Core i7 4770)で実行 → PCDファイルサイズ290MB
  3. デスクトップPC(Core i9 9900k)で実行 → PCDファイルサイズ350MB

## Livox-SDKとPCスペック (Livox\_mapping, FAST-LIO)

---

- rosbagの再生速度を遅くするとマッピングされる点群の量は多くなる。
- 再生速度の変更は実行時末尾に `-r <実行速度>` を追加するだけ。

`rosbag play <bagファイルのパス・ファイル名>.bag -r 0.5`

この場合、再生速度は0.5倍になる。

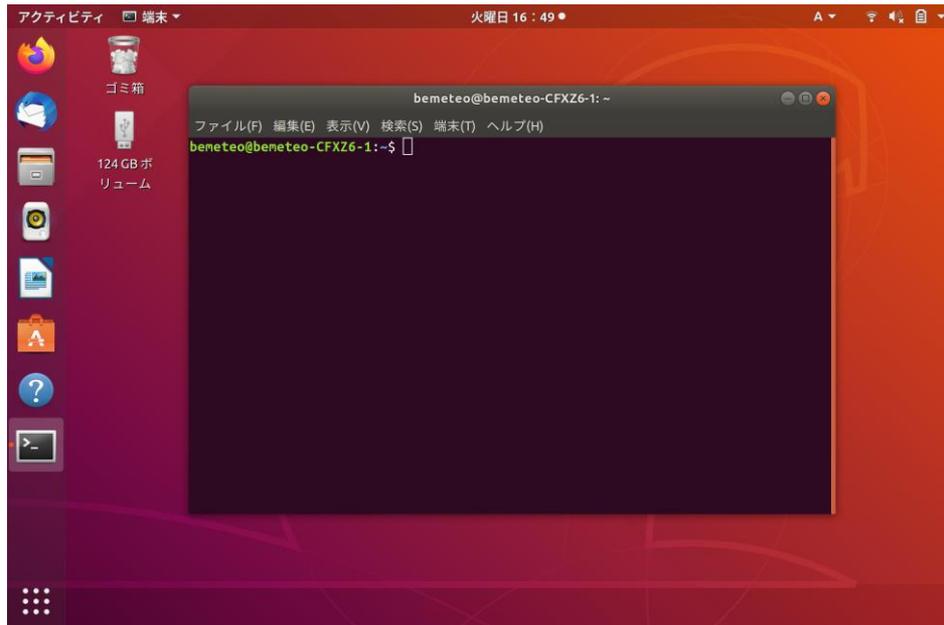
例) 2021/7/27撮影のBAGファイルでマッピング

1. 通常速度で実行 → PCDファイルサイズ463MB
2. 0.5倍速で実行 → PCDファイルサイズ758MB
3. 2.0倍速で実行 → PCDファイルサイズ341MB

# SLAMのインストール準備

PCのOSはUbuntu 18.04を使用

端末を開く (Ctrl+Alt+T)



- まずはLivox\_mapping, FAST-LIOの動作に必要なプラットフォームなどをインストールする（以降のコマンドを入力して実行）

# SLAMのインストール準備

---

## ROS Melodicのインストール

- `sudo apt update`
- `sudo apt upgrade`
- `sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'`
- `sudo apt install -y curl`
- `curl -sSL 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1CF6E31E6BADE8868B172B4F42ED6FBAB17C654' | sudo apt-key add -`
- `sudo apt install -y ros-melodic-desktop-full`
- `sudo apt install python-rosdep`
- `sudo rosdep init`
- `rosdep update`
- `cd`
- `echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc`
- `source ~/.bashrc`
- `sudo apt install -y python-rosinstall python-rosinstall-generator python-wstool build-essential python-catkin-tools`

# SLAMのインストール準備

---

## Point Cloud Libraryのインストール

- `sudo apt install libpcl-dev`
- `sudo apt install cmake`

## Eigenのインストール

- `cd`
- `git clone https://gitlab.com/libeigen/eigen.git`

## OpenCVのインストール

- `sudo apt-get install libopencv-dev python3-opencv`

# SLAMのインストール準備

---

## Livox SDKのインストール

- `cd`
- `sudo apt install cmake`
- `git clone https://github.com/Livox-SDK/Livox-SDK.git`
- `cd Livox-SDK`
- `cd build && cmake ..`
- `make`
- `sudo make install`

## Livox ros driverのインストール

- `cd`
- `git clone https://github.com/Livox-SDK/livox_ros_driver.git`  
`ws_livox/src`
- `cd ws_livox`
- `catkin_make`
- `source ./devel/setup.sh`

# Livox\_mappingのインストール（Midシリーズ）

---

準備が整ったので、最後に[Livox\\_mapping](#)をインストールする

- `cd`
- `mkdir -p catkin_ws/src`
- `cp -r ~/ws_livox/src/livox_ros_driver ~/catkin_ws/src/`
- `cd ~/catkin_ws/src`
- `git clone https://github.com/Livox-SDK/livox_mapping.git`
- `cd ..`
- `catkin_make` ※エラーでるかも要注意
- `source ~/catkin_ws/devel/setup.bash`



続いて保存場所などの詳細設定を行う

# FAST-LIOのインストール (Avia, Horizon)

---

準備が整ったので、最後にFAST-LIOをインストールする

- `cd` 15ページのLivox\_mappingのインストールを行った場合は  
赤枠内をとばす
- `mkdir -p catkin_ws/src`
- `cp -r ~/ws_livox/src/livox_ros_driver ~/catkin_ws/src/`
- `cd ~/catkin_ws/src`
- `git clone https://github.com/hku-mars/FAST_LIO.git`
- `cd FAST_LIO`
- `cd include`
- `git rm -r ikd-Tree`
- `git submodule add https://github.com/hku-mars/ikd-Tree`
- `cd ikd-Tree/ikd-Tree`
- `mv *.* ../`
- `cd ~/catkin_ws`
- `catkin_make`
- `source devel/setup.bash`



続いて保存場所などの詳細設定を行う

# SLAMの詳細設定

## 点群ファイル保存場所の設定

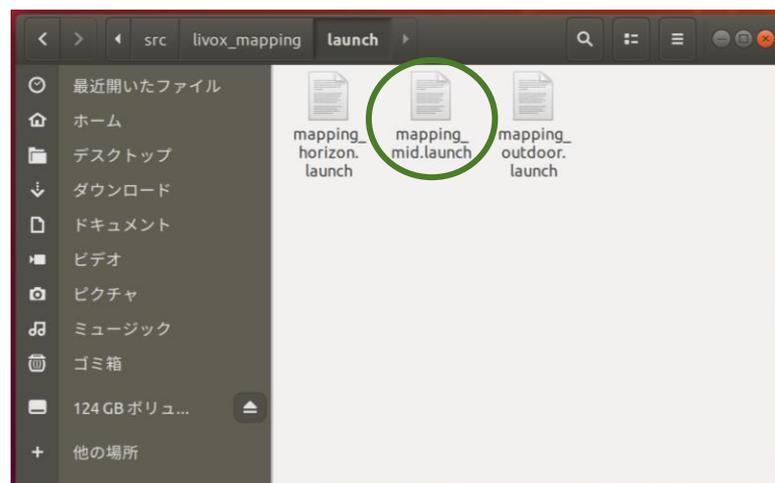
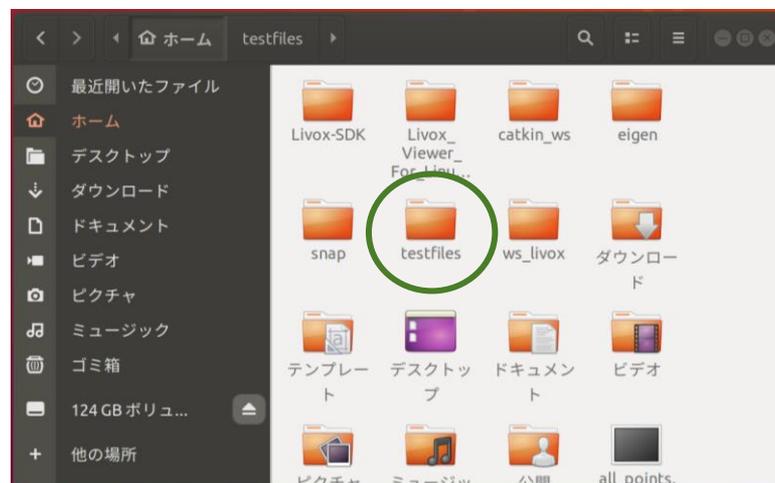
- 任意の場所に保存場所を作る  
(例: `/home/<ユーザー名>/testfiles`)

- `~/catkin_ws/src/livox_mapping/launch/mapping_mid.launch`を開く

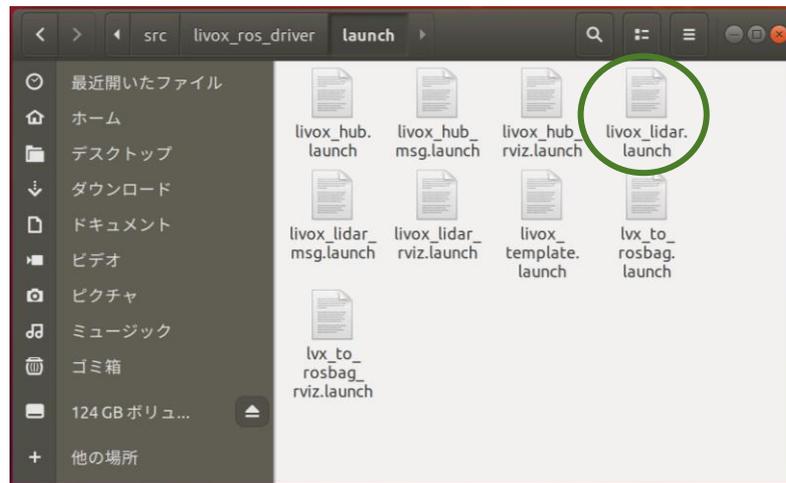
### ◆ 9行目を変更

```
<param name="map_file_path" type="string" value="/home/<ユーザー名>/testfiles" />
```

→保存



# SLAMの詳細設定 (livox\_mapping, FAST-LIO)



- ~/catkin\_ws/src/livox\_ros\_driver/launch/  
livox\_lidar.launchを開く (FAST-LIOの場合は, livox\_lidar\_msg.launch)

- ◆ 11行目を変更

```
<arg name="rosvbag_enable" default="true"/>
```

- ◆ 39行目を変更

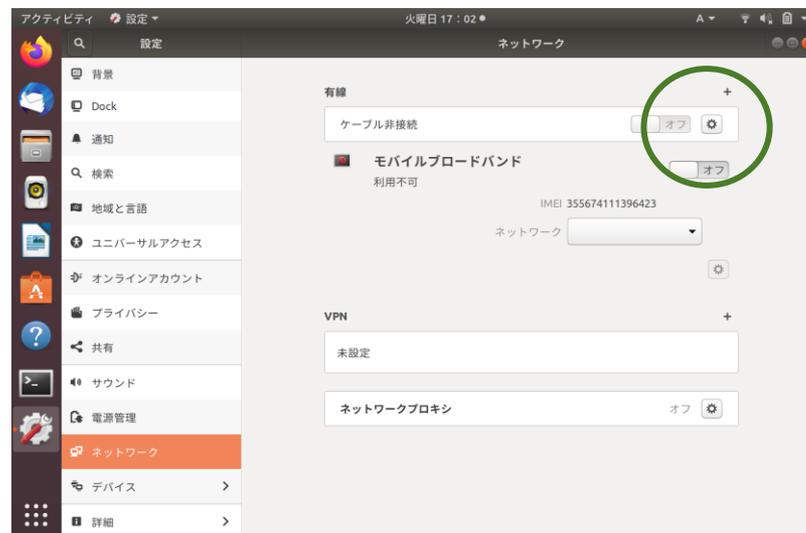
```
<node pkg="rosvbag" type="record" name="record" output="screen"  
args="-a -0 /home/<ユーザー名>/testfiles/test.bag"/>
```

→保存

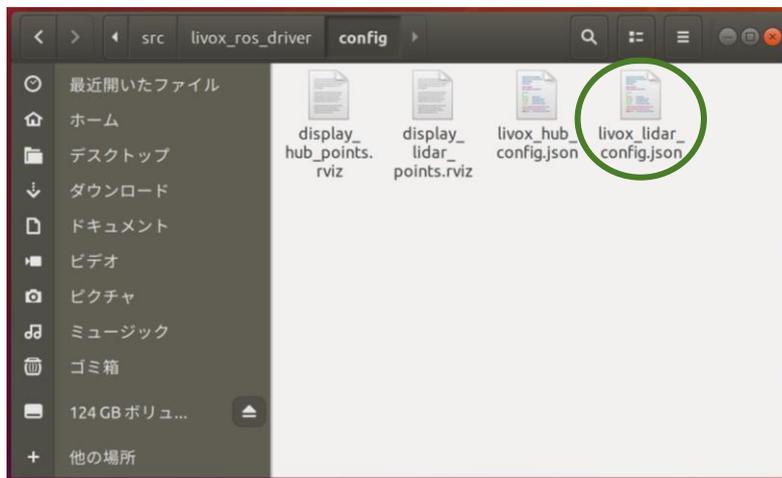
# SLAMの詳細設定

## 有線LANの設定

- 設定→ネットワークを開く
- IPv4の設定を以下のように変更
  - ◆ IPv4メソッド: 手動
  - ◆ アドレス: 192.168.1.50
  - ◆ ネットマスク: 255.255.255.0
  - ◆ ゲートウェイ: 192.168.1.1



# SLAMの詳細設定



## 撮影モードの設定

- catkin\_ws/src/livox\_ros\_driver/configにあるlivox\_lidar\_config.jsonを開く
- “return\_mode”の数字を変更する
  - ◆ 0: single first return (最初の反射を記録)
  - ◆ 1: strongest single return (最も強度の強い反射を記録)
  - ◆ 2: dual return (反射を2回まで記録)
  - ◆ 3: triple return (反射を3回まで記録)

# SLAMの実行 - 撮影しながら行う場合 (Livox\_mapping)

LiDAR, PC, 電源を接続してLiDARを起動し, 端末を開いて以下のコマンドを実行する

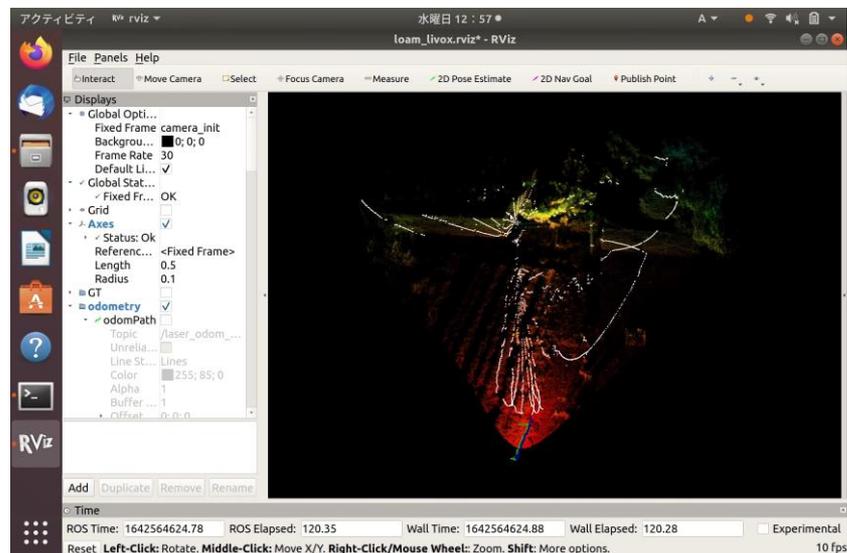
- `cd ~/catkin_ws/src`
- `source ~/catkin_ws/devel/setup.bash`
- `roslaunch livox_mapping mapping_mid.launch`

rvizが立ち上がるので, “Ctrl+Z”でプロセスを一時停止し, 以下のコマンドを実行する

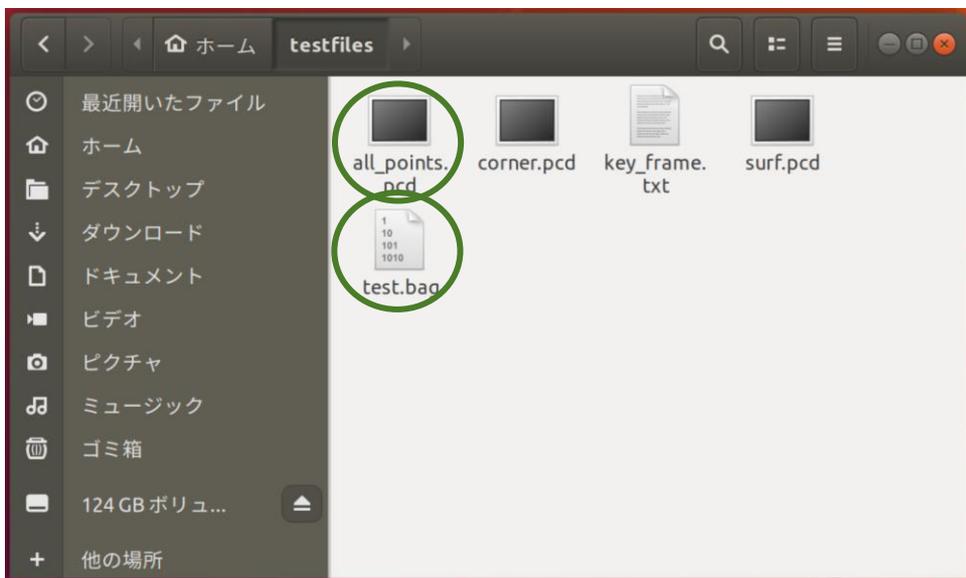
- `roslaunch livox_ros_driver livox_lidar.launch`



マッピングが開始される



# SLAMの実行 - 撮影しながら行う場合 (Livox\_mapping)



**all\_points.pcd**  
SLAMによって生成された  
点群データ

**test.bag**  
LiDAR観測データを記録

マッピングを終了する場合は端末を閉じる

- 先ほど作成した保存場所にファイルが保存される
- **ファイルは上書きされるため**, 続けて撮影する場合は保存されたファイルの名前を変更する

# SLAMの実行 - 撮影しながら行う場合 (FAST-LIO)

---

LiDAR, PC, 電源を接続してLiDARを起動し, 端末を開いて以下のコマンドを実行する

- `cd ~/catkin_ws/src`
- `source ~/catkin_ws/devel/setup.bash`
- `roslaunch fast_lio mapping_avia.launch` (Avia)
- `roslaunch fast_lio mapping_horizon.launch` (Horizon)

rvizが立ち上がるので, “Ctrl+Z”でプロセスを一時停止し, 以下のコマンドを実行する

- `roslaunch livox_ros_driver livox_lidar_msg.launch`



マッピングが開始される

# SLAMの実行 - bagファイルから行う場合

---

端末を起動し、以下のコマンドを実行する

- `cd ~/catkin_ws/src`
- `source ~/catkin_ws/devel/setup.bash`
- `roslaunch livox_mapping mapping_mid.launch` (Mid)
- `roslaunch fast_lio mapping_avia.launch` (Avia)
- `roslaunch fast_lio mapping_horizon.launch` (Horizon)

rvizが立ち上がるので、“Ctrl+Z”でプロセスを一時停止し、以下のコマンドを実行する

- `rosbag play <bagファイルのパス・ファイル名>.bag`

補足:

rosbagの再生速度を遅くすると生成される点の量（密度）が増加  
変更は末尾に `-r <実行速度>` を追加するだけ

- `rosbag play <bagファイルのパス・ファイル名>.bag -r 0.5`

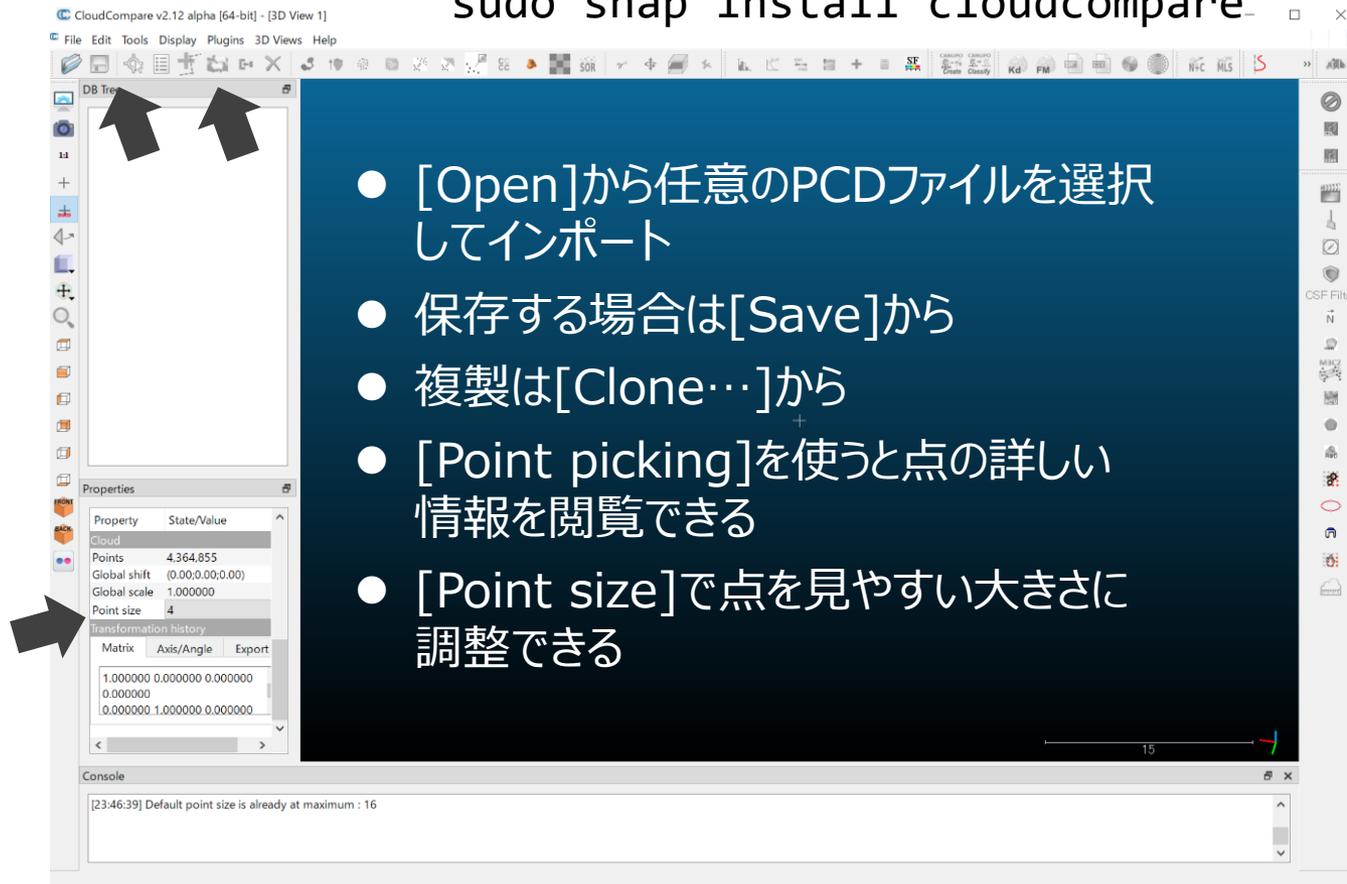
# 点群データの処理

## CloudCompareを用いた点群データの処理

### ● 基本操作

```
sudo apt install snapd
```

```
sudo snap install cloudcompare
```



The screenshot shows the CloudCompare interface with a blue overlay containing the following list of operations:

- [Open]から任意のPCDファイルを選択してインポート
- 保存する場合は[Save]から
- 複製は[Clone...]から
- [Point picking]を使うと点の詳細な情報を閲覧できる
- [Point size]で点を見やすい大きさに調整できる

# 点群データの処理

## ● 切り出し

① [Segment] をクリック

② 範囲を指定

③ 切り出しか切り抜きを選択して実行(✓)

対象の選択を  
忘れずに

CloudCompare v2.12 alpha [64-bit] - [3D View 1]

File Edit Tools Display Plugins 3D Views Help

DB Tree

- all\_points.pcd (E:/LIDAR/...
- all\_points

Properties

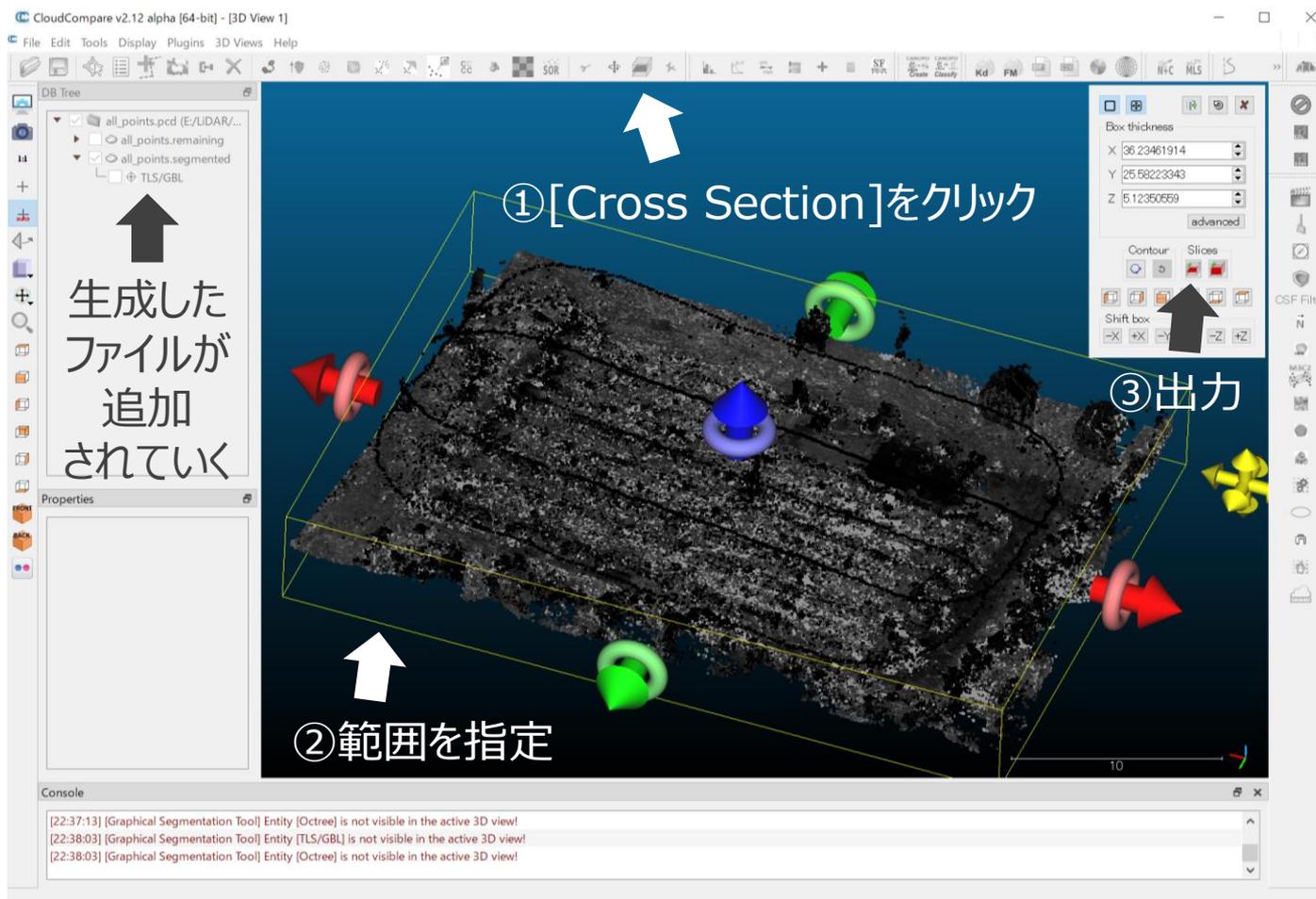
Property	State/Value
X:	476.457
Box dims... Y:	500.151
Z:	70.5183
Shifted box... X:	-117.042
Y:	-16.6695
Z:	-11.9566
Global box ... X:	-117.041763
Y:	-16.669533
Z:	-11.956624
Info	Object ID: 272 - Chil
Current Dis...	3D View 1
Cloud	

Console

```
[22:37:13] [Graphical Segmentation Tool] Entity [Octree] is not visible in the active 3D view!  
[22:38:03] [Graphical Segmentation Tool] Entity [TLS/GBL] is not visible in the active 3D view!  
[22:38:03] [Graphical Segmentation Tool] Entity [Octree] is not visible in the active 3D view!
```

# 点群データの処理

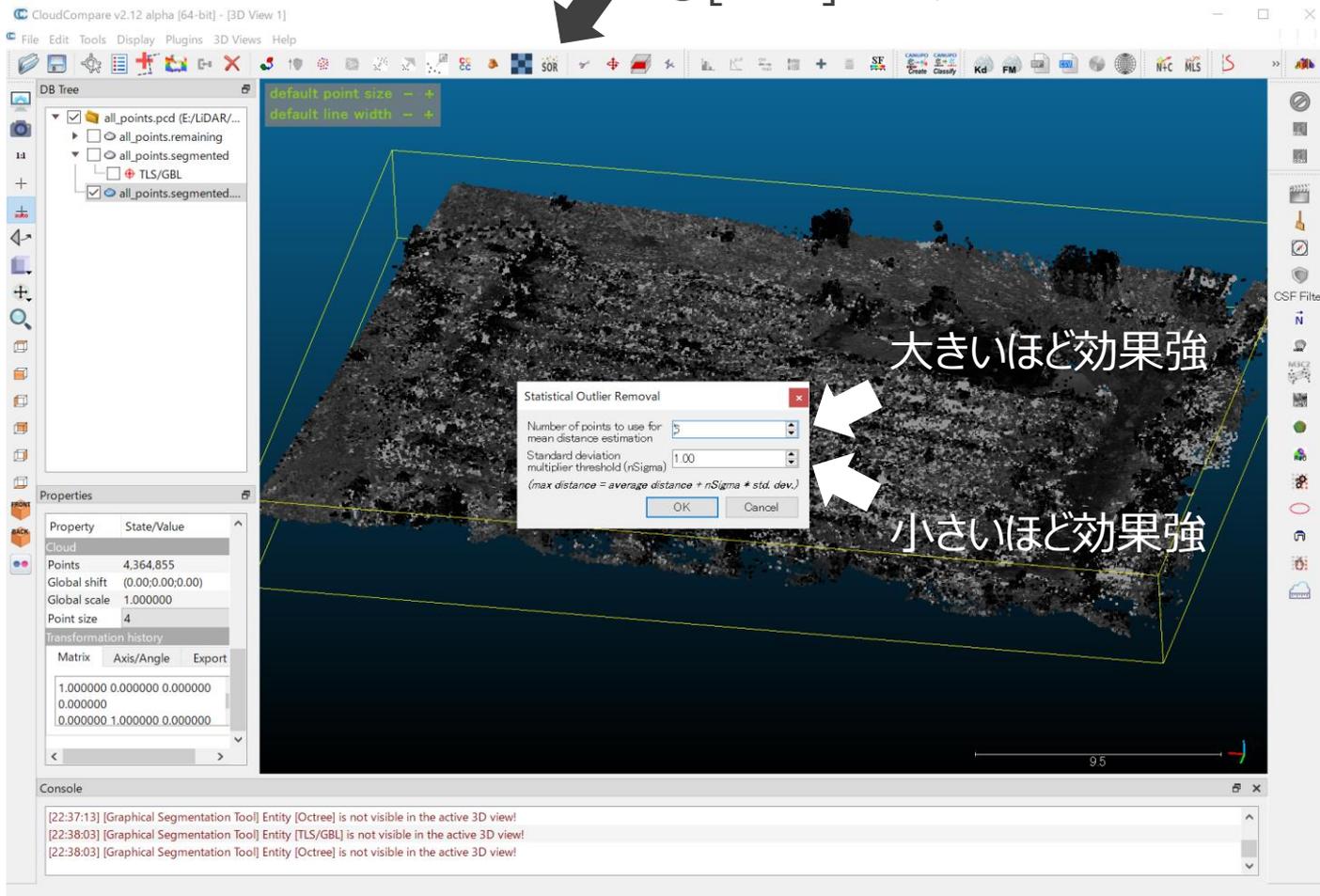
- Cross Sectionを使うと任意の立方体の範囲で切り出せる



# 点群データの処理

## ● ノイズ除去

①[SOR]をクリック

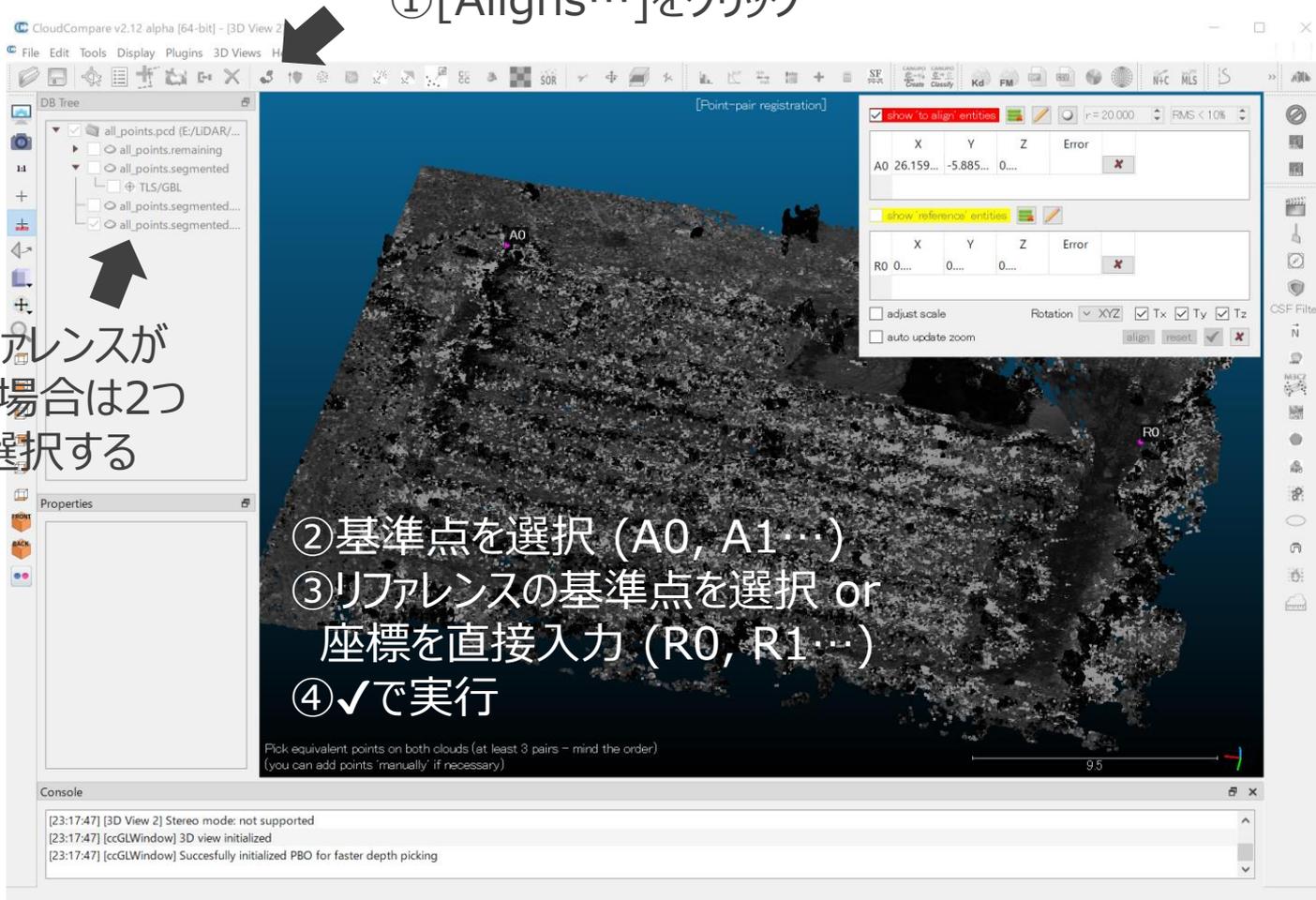


# 点群データの処理

## ● 位置合わせ

① [Aligns...]をクリック

リファレンスがある場合は2つ  
選択する

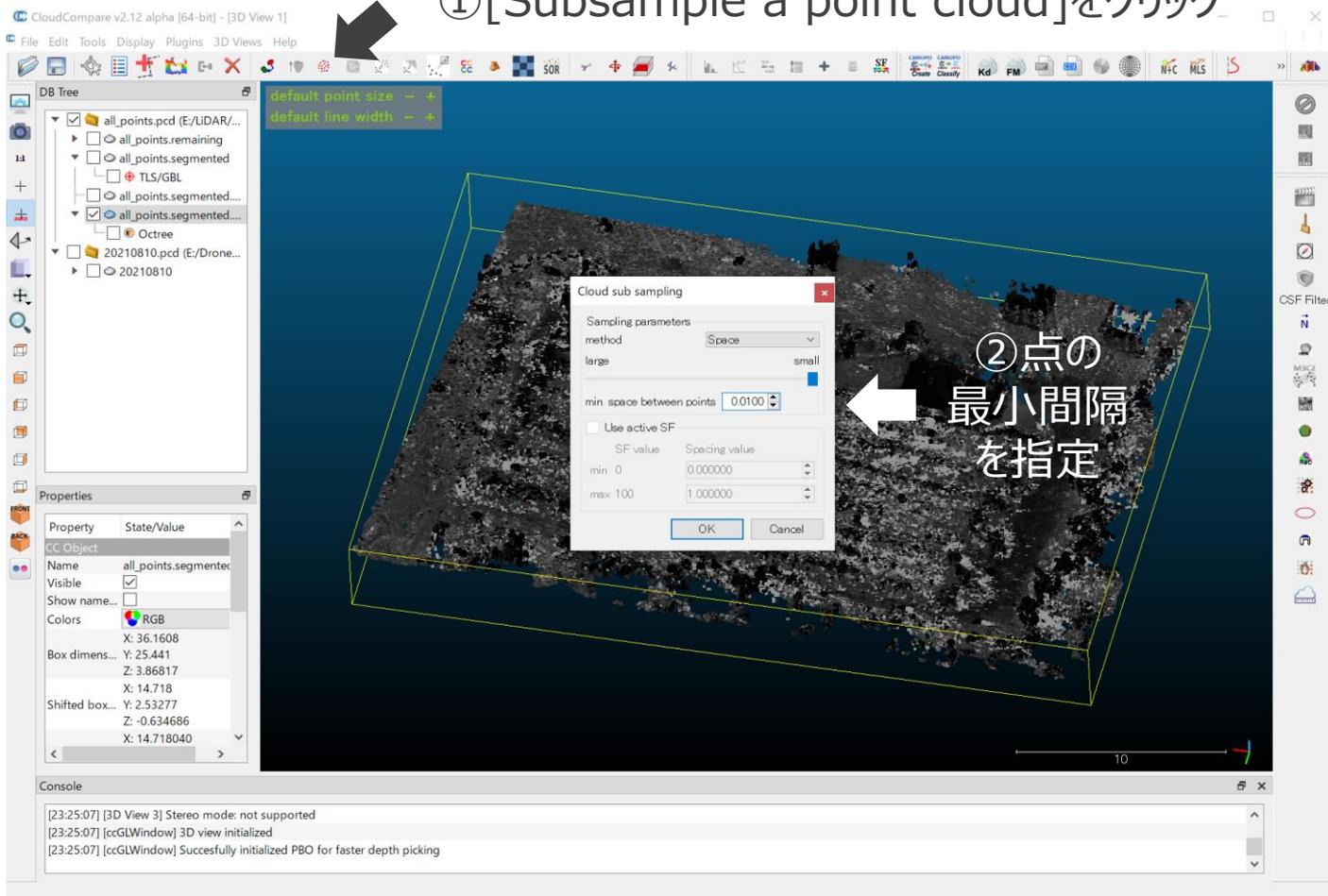


- ② 基準点を選択 (A0, A1...)
- ③ リファレンスの基準点を選択 or 座標を直接入力 (R0, R1...)
- ④ ✓で実行

# 点群データの処理

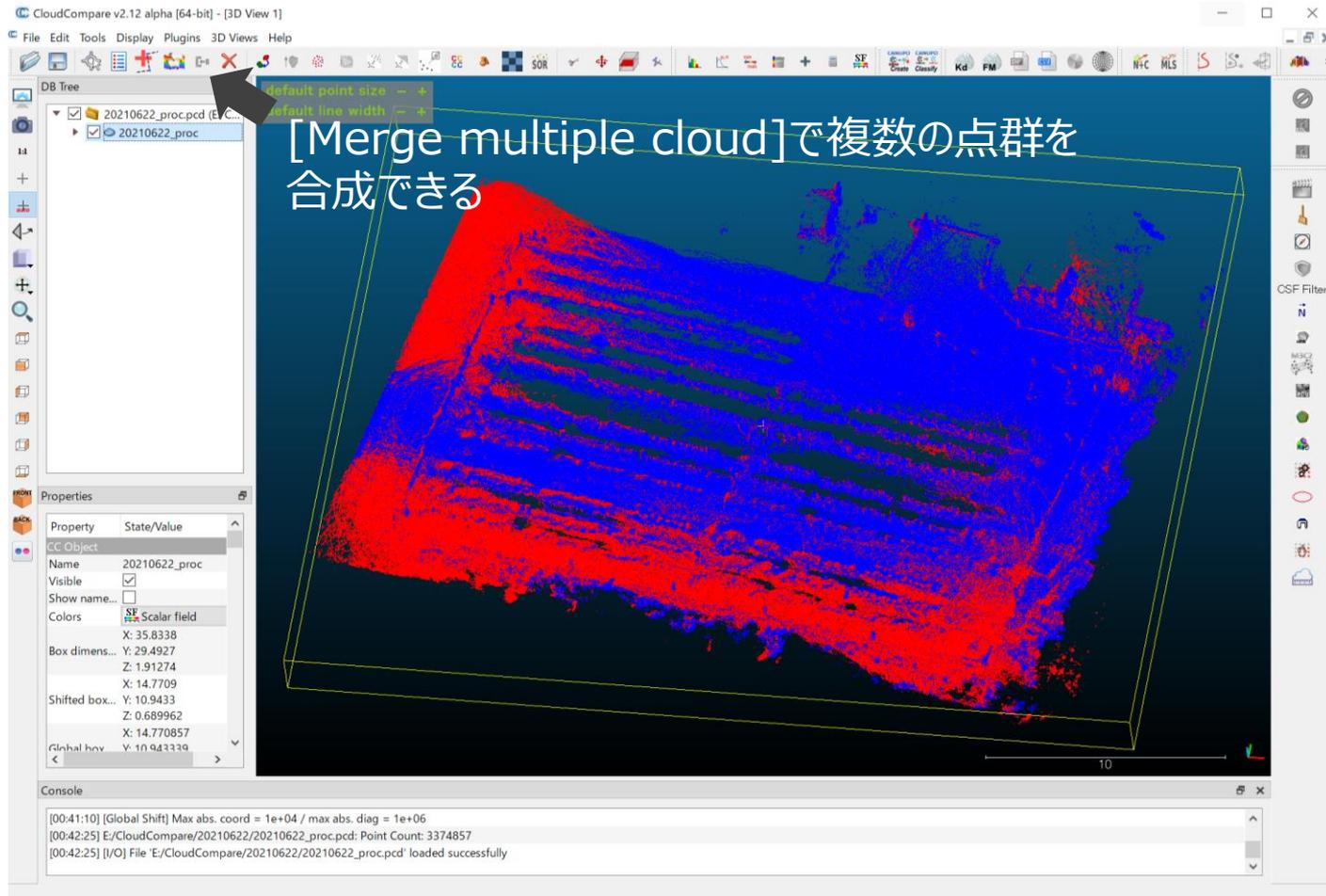
## ● リサンプリング

① [Subsample a point cloud] をクリック



# 点群データの処理

## ● 点群の合成



# 点群データの処理

- 点群を標高別に色分け

[Tools]→[Projection]→  
[Export coordinates to SFs]  
をクリック

[Color Scale]から  
色味を変更可能

Z軸の値で色分けした結果

```
[23:46:39] Default point size is already at maximum : 16
[00:07:08] E:/CloudCompare/20210810/20210810_proc.pcd: Point Count: 4457428
[00:07:09] [I/O] File 'E:/CloudCompare/20210810/20210810_proc.pcd' loaded successfully
```

# 点群データの処理

## ● ラスタ化

① [Convert...] をクリック

② ピクセルサイズ、集計方法、内挿の有無などを選択

③ [Update grid] をクリック

④ [Raster] をクリック

⑤ [OK] で保存先を選択

The screenshot shows the CloudCompare v2.12 alpha interface. The main window displays a 3D point cloud of a building. The 'Rasterize' dialog box is open, showing the following settings:

- Grid: step: 0.030000, size: 1260 x 1086, active layer: Height grid values, range: 2.0289 [-0.180521 ; 1.84838]
- Projection: direction: Z, cell height: average, interpolate SFs: average value, resample input cloud: unchecked
- Empty cells: Fill with: interpolate, value: 0.000000
- Update grid: button
- Export: Cloud, Mesh, Hillshade, Contour plot
- Export per-cell statistics as SFs: average height (checked), min height, max height, height std. dev, height range
- Buttons: Raster, Image, Matrix

The 'Raster export options' dialog box is also open, showing the following settings:

- Raster dimensions: 1260 x 1086
- Export RGB colors: unchecked
- Export heights: checked
- Export active layer: unchecked
- Export all scalar fields: unchecked
- Export density (population per cell): unchecked
- Buttons: OK, Cancel

The 'Properties' panel on the left shows the object name '20210810\_proc' and its bounding box dimensions.

# 点群データの処理

## ● メッシュ化



# 点群データの処理

## ● 植生の除去

The screenshot displays the CloudCompare v2.12 alpha interface. The 'Plugins' menu is open, showing a list of available plugins. The 'CSF Filter' plugin is highlighted, and a white arrow points to it with the text '[CSF Filter]をクリック' (Click [CSF Filter]). The main 3D view shows a point cloud of a plant structure, rendered in a color gradient from blue to green. The 'DB Tree' on the left shows the loaded data, including a folder '21101616' and a sub-object 'ground'. The 'Properties' panel at the bottom left shows the 'ground' object's properties, including its name, visibility, and a 'Scalar field' (SF) with a range from 522.854 to 25.3169. The 'Console' at the bottom shows a message: '[15:54:39] [I/O] File 'C:/Users/a4k4i/Desktop/2110161611/yoDoi\_dextral.bin' loaded successfully'.

CloudCompare v2.12 alpha [64-bit] - [3D View 1]

File Edit Tools Display Plugins 3D Views Help

DB Tree

- 21101616
  - ground

Properties

Property	State/Value
Name	ground
Visible	<input checked="" type="checkbox"/>
Normals	<input type="checkbox"/>
Show name...	<input type="checkbox"/>
Colors	SF, Scalar field
Box dims...	X: 522.854 Y: 275.755 Z: 71.6368
Shifted box...	X: -180.586 Y: -64.3656 Z: 25.3169

Console

```
[15:54:32] [BIN] Version 5.2 (coords: float / scalar: float)  
[15:54:39] [I/O] File 'C:/Users/a4k4i/Desktop/2110161611/yoDoi_dextral.bin' loaded successfully
```