

## タイトル：

### 第 2 回 スマートフォンの LiDAR センサで 3 次元モデルづくり



3 次元モデルの例：scaniverse アプリで筆者が作成

第 1 回はフォトグラメトリを使った 3 次元モデル生成についてご紹介をしました。第 2 回は iLiDAR センサを使った 3 次元モデル生成と点群データの可視化についてご紹介をします。

## 点群データとは



点群データの例：CloudCompare で筆者が作成

点群（てんぐん）データは点（ポイント）の集まりのデータであり、直交座標値（ $X, Y, Z$ ）+色情報（ $R, G, B$ ）+  $\alpha$ で構成された座標と色情報のある3次元の空間情報データです。ポイントクラウド(point cloud)とも呼ばれています。

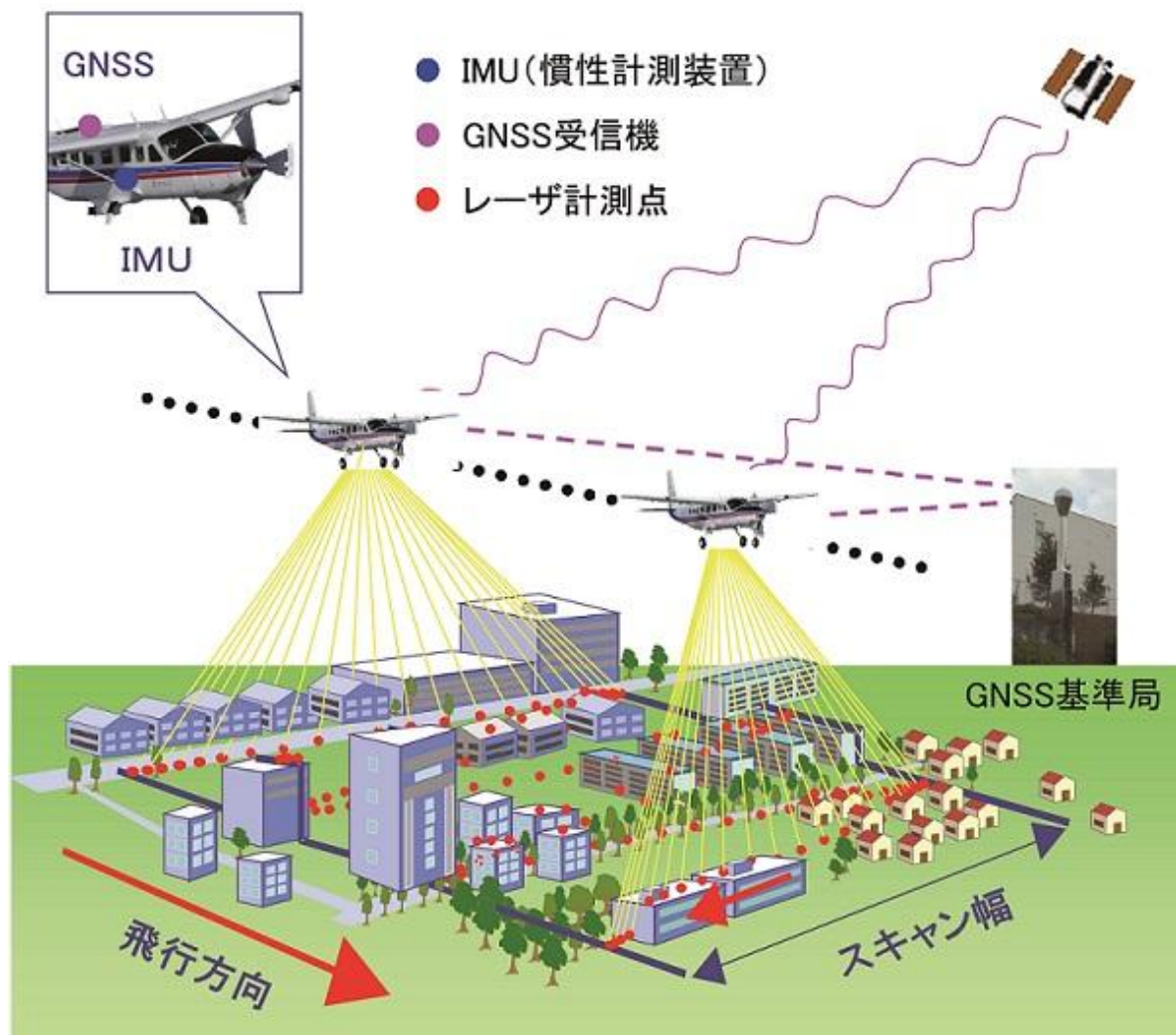
既存の画像データやデジタル写真も同じ原理であり、詳細なドットを高密度に集めることで滑らかな質感を再現しています。点の密度は高くなれば高精度のデータとなりますが、点の密度に比例してデータサイズが膨大となります。

## 点群データの作成

点群データは、3次元レーザスキャナ、UAV 搭載カメラ(Unmanned aerial vehicle)、地上設置型レーザスキャナ、スマートフォン搭載カメラ等で計測して作成されます。

## レーザ計測

点群データの作成は、物体表面をレーザの反射を用いて多数の点の3次元座標を計測し作成されます。ここで実際に行われているレーザ計測方式を簡単に紹介します。



航空レーザ測量の仕組み：Web サイトより

[https://www.gsi.go.jp/kankyochiri/Laser\\_senmon.html](https://www.gsi.go.jp/kankyochiri/Laser_senmon.html)

- 航空レーザ測深（ALB：Airborne Laser Bathymetry）
  - レーザスキャナを搭載した航空機やヘリコプターから計測
  - 複数のレーザで、陸上と水部（河川や海の水深）を計測する方式
- 航空レーザ測量（LP：Laser Profiler）
  - レーザスキャナを搭載した航空機やヘリコプターから計測
  - 陸上を計測する方式
  - 従来から幅広く実施されている方式
- 移動計測車両（MMS：Mobile Mapping System）
  - レーザスキャナを搭載した車両が道路を走りながら計測
  - 高精度な位置情報の収集が可能
- 地上型または固定型
  - 移動しないで計測する方式で、3次元レーザスキャナ等の計測機を使用して計測
- iPhone/iPad
  - iPhone12 以降の Pro/Pro Max、iPad Pro(2020)以降 に搭載された LiDAR センサで計測

- 計測範囲を最大 5m の距離に制限することでコストを抑えている
- 照射密度を低くしてコストを抑えている
- 小さいモノや複雑なモノを 3 次元モデル化するのは不得意
- 家の中の部屋くらいの広さまでの測定に向いている

## LiDAR センサとは

LiDAR センサ (Light Detection and Ranging(光による検知と測距)/Laser Imaging Detection and Ranging(レーザ画像検出と測距)) は、赤外線等を使った深度(depth)センサーで、レーザの光が反射してから戻ってくるまでの時間をもとにし、被写体となった空間の奥行きや物体の 3 次元的な形情報を取得します。

## 3 次元モデルスキャン

レーザ光を使用する LiDAR センサで 3 次モデルを作成する場合、2 つのスキャン方法があります。

### ● 点群スキャン

点でスキャンする方法

- ・ メリット
  - ✧ 複雑なモデルの形状を正確に表す
  - ✧ スキャンスピードが速い
- ・ デメリット
  - ✧ 点群自体の編集が少し難しい



- メッシュスキャン

三角形のポリゴンメッシュ（面形状）でする方法

- ・ メリット

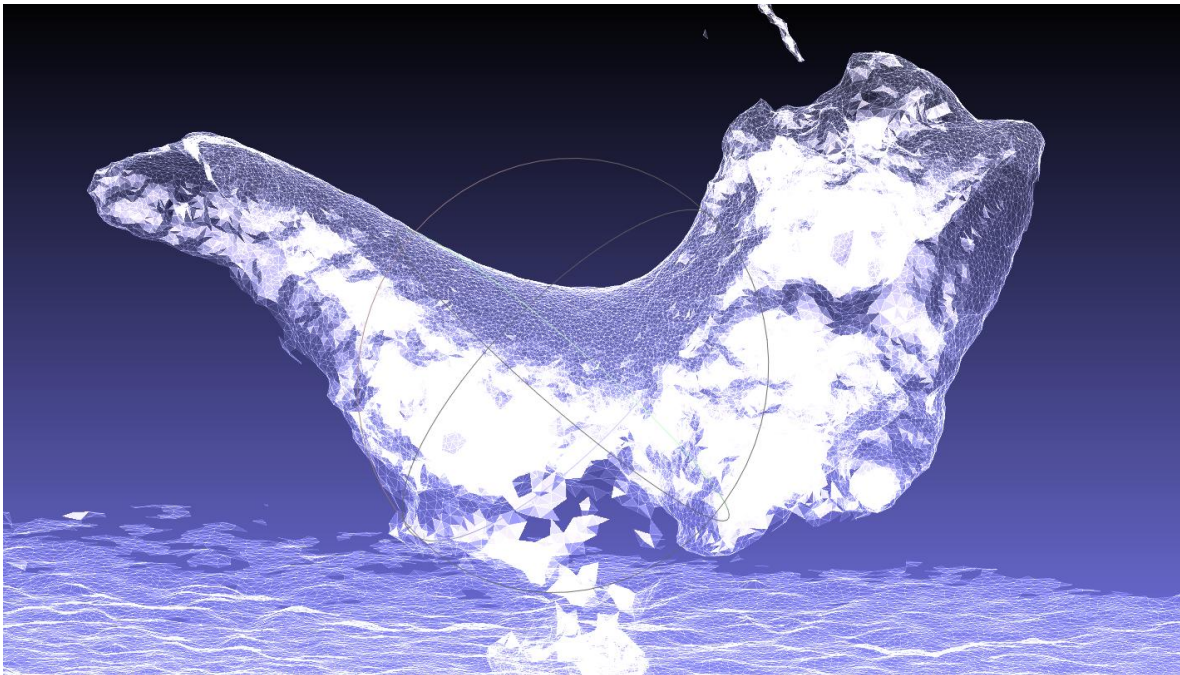
- ✧ 文字やテクスチャなど面の細かい情報もスキャン可能

- ✧ CG ソフトにエクスポートして編集可能

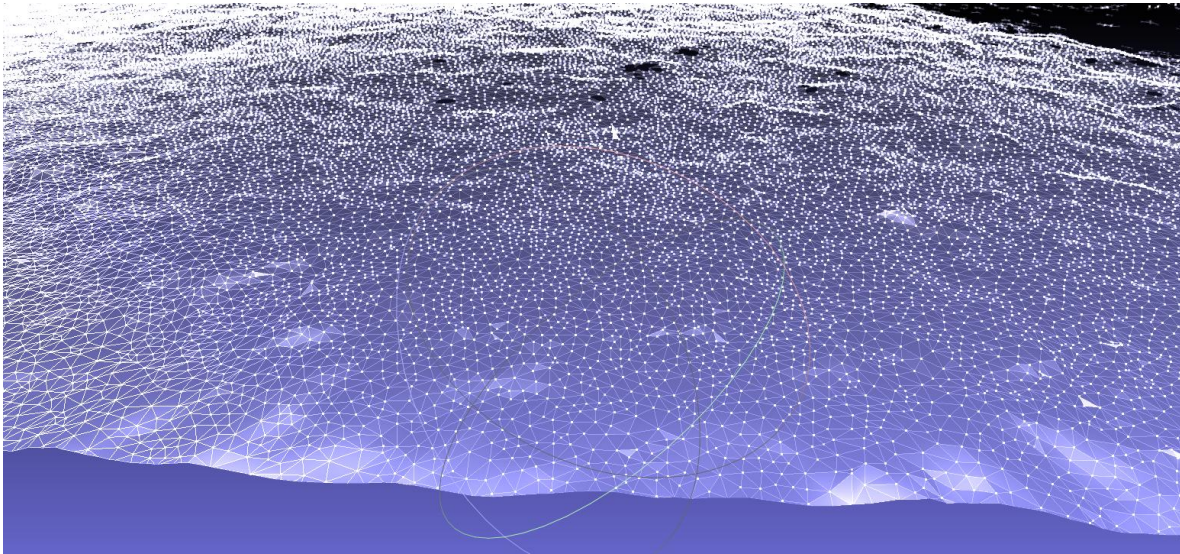
- ・ デメリット

- ✧ データサイズが大きい

下記の例は、三角形のポリゴンメッシュ（面形状）の3次元モデルとなります。2枚目の拡大図をみると、三角形の集合体で構成されていることがわかります



メッシュデータ例：筆者が *MeshLab* で作成



メッシュデータを拡大表示：筆者が *MeshLab* で作成

スマートフォンの LiDAR センサを使用して 3 次元モデルを作成する場合は、モバイルスキャン協会 (<https://mobilescan.jp/>) のページに、スマートフォンアプリの特徴とスマートフォンでスキャンする方法の説明があるので参考にするとよいと思います。

### 3 次元モデルのファイル形式

3 次元モデルのファイルにはさまざまなファイル形式があります。「点群データ系」と「メッシュデータ系」の代表的なファイルフォーマットを紹介します

## 点群データ系

### LAS 形式

- オープンデータで公開されている点群データはこの形式で公開されていることが多いです。航空機レーザ測量の標準フォーマットでもあります
- ASPRS(米国写真測量学会 : American Society for Photogrammetry and Remote Sensing) によって作成された形式
  - ・ LAS 1.4 Specification Approved by ASPRS Board  
(<https://www.asprs.org/news/press-releases/press-release-archives/las-1-4-specification-approved-by-asprs-board.html>)

### LAZ 形式

- LAS 形式を圧縮したデータ形式
- LASzip - free and lossless LiDAR compression (<https://laszip.org>)

### PLY 形式 (Polygon File Format)

- 3次元スキャナから3次元データを格納するために設計されたデータ形式
- テキストデータまたはバイナリデータの形式がある
- 点群またはメッシュのデータ形式がある
- PLY - Polygon File Format (<http://paulbourke.net/dataformats/ply/>)

### PCD 形式 (Point Cloud Data)

- PCL (Point Cloud Library) <http://pointclouds.org/> で使用しているデータ形式



- PCL は 3 次元の点群データを扱う、C++言語によるオープンソースのソフトウェアライブラリ
- The PCD (Point Cloud Data) file format  
[https://pointclouds.org/documentation/tutorials/pcd\\_file\\_format.html](https://pointclouds.org/documentation/tutorials/pcd_file_format.html)

## メッシュデータ系

### *STL 形式 (Standard Triangulated Language)*

- 3 次元 CAD ソフト用のファイルフォーマットの 1 つ
- 対応している 3 次元プリンターが多い
- 3 角形の集合として立体を表現（サーフェスモデル）
- テキストデータまたはバイナリデータの形式がある
- The STL Format - Standard Data Format for Fabbers  
([http://www.fabbers.com/tech/STL\\_Format](http://www.fabbers.com/tech/STL_Format))

### *OBJ 形式*

- Wavefront Technologies 社によって開発されたファイル形式
- 多くの 3 次元モデリング、レンダリングソフトが対応
- 動かない 3 次元オブジェクトに適している
- Obj Specification as used by Wavefront  
(<http://www.martinreddy.net/gfx/3d/OBJ.spec>)

## FBX 形式

- AutoDesk 社が公開したファイル形式
- 3DCG(3 Dimensional Computer Graphics)のデファクトフォーマット
- 動く 3 次元オブジェクトに適している
- Adaptable file format for 3 次元 animation software  
(<https://www.autodesk.com/products/fbx/overview>)

## 点群データの編集ツール

点群データの表示、変換、サイズ変更をする場合は、第 1 回で紹介をした

MeshLab(<https://www.meshlab.net/>)の他に CloudCompare

(<http://www.danielgm.net/cc/>)も使用してみてください。

CloudCompare は点群データ処理のオープンソースのソフトウェアで、Windows、Mac、Linux 版があり、操作性（インターフェイス）はわかりやすく扱いやすく、点群データを表示確認するのに便利なツールです。




## CloudCompare

3D point cloud and mesh processing software  
Open Source Project

Want to support/help us?



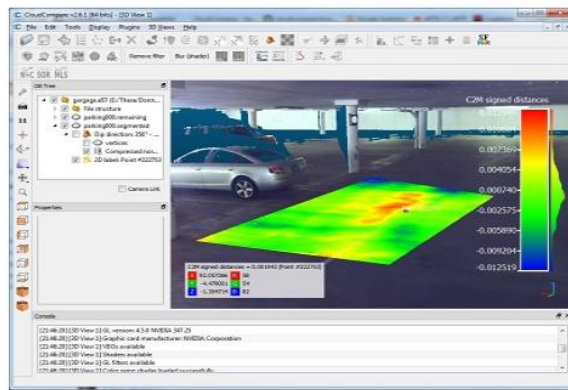
Follow the project on 

[Home](#) - [Presentation](#) - [Download](#) - [Github](#) - [Tutorials](#) - [Documentation](#) - [Forum](#) - [Declare a bug](#)

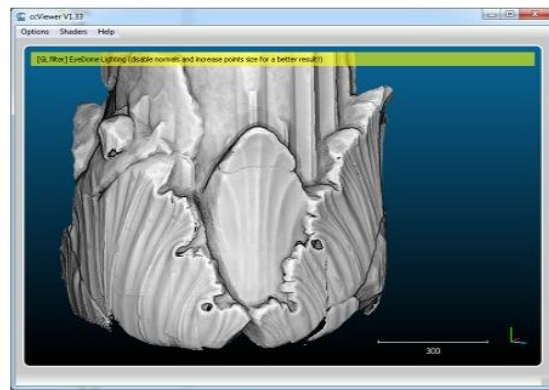
Welcome to the official website of the **CloudCompare** project.

This is a Free project. **Just as people and countries should be and remain.**

Want to know when a new release comes out? [Subscribe to the newsletter](#)



**CloudCompare** (view, edit and process)



**ccViewer** (light viewer only)

CloudCompare : <https://www.danielgm.net/cc/>

## 事例：レーザスキャナを搭載した MMS 車で道路維持管理



G 空間情報センター：移動計測車両（MMS）で取得した 3 次元点群データの表示事例

<https://www.geospatial.jp/ckan/dataset/shizuoka-2019-pointcloud/resource/c784a23f-9314-43a6-903e-e656578c23da>

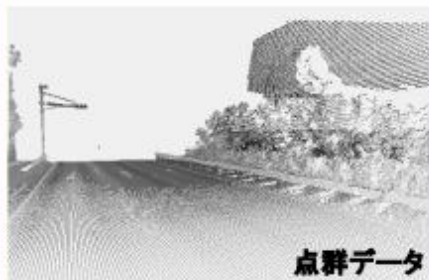


## MMS(モービルマッピングシステム)

GNSS、レーザースキャナ、カメラなどの機器を搭載し、走行しながら3次元の道路の形状・データを高精度で効率的に取得

### レーザー

物体に照射したレーザ光の反射波により点群データの取得が可能



点群データ

### カメラ

取得した画像により地物等を判別し、点群データに地物情報の付加が可能

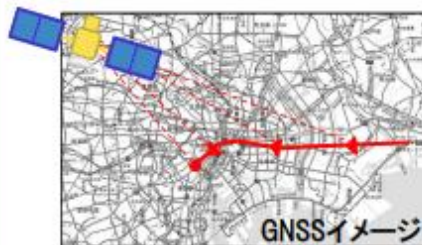


カメラ画像



### GNSS(Global Navigation Satellite System(s)) (汎地球測位航法衛星システム)

衛星を用いた測位システムの総称で、継続的な位置取得により経路把握が可能



GNSSイメージ



センシング装置を搭載した車両

移動計測車両（MMS : Mobile Mapping System）の例

<https://www.mlit.go.jp/report/press/content/001498067.pdf>

国土交通省：MMS による三次元点群データ等の提供事業を開始

[https://www.mlit.go.jp/report/press/road01\\_hh\\_001577.html](https://www.mlit.go.jp/report/press/road01_hh_001577.html)

移動計測車両（MMS : Mobile Mapping System）は、高精度計測車両等の測量用の車両で走行しながら、GPS、動画、レーザスキャナ、センサを組み合わせで一度に道路の表面と道路周辺の情報を収集します。

GPS や各種センサは、角速度・加速度を高精度に計測する慣性計測装置（IMU : Inertial Measurement Unit）や車輪の回転等と連動して制御されているので、高精度のデータを収集することが可能です。

収集した道路情報、位置情報は道路周辺の基盤情報となり、道路管理業務や防災業務等に活用されています。例えば、道路表面のひび割れやわだち掘れや道路の経年変化の記録、道路周辺の道路施設や法面※1、道路標識・表示確認などの管理業務に活用されています。

※1 法面（のりめん）：切土や盛土により作られる人工的な斜面のこと

## スマートフォンで作成した点群データを可視化

スマートフォンで作成した点群データを Google Colaboratory で可視化します。

### 3次元モデルのデータ作成

3次元モデルのデータは Scaniverse (<https://scaniverse.com>)を使用します。

Scaniverse は、Ingress やポ Pokémon GO 等の AR ゲームを開発している Niantic, Inc.( <https://nianticlabs.com/>)の iOS 向けアプリです

#### 特徴

- 無料で利用可能
- LiDAR センサを搭載した iPhone/iPad のみ対応
  - ・ iPhone12Pro/13Pro/14pro シリーズ、iPad Pro(2020 以上)
- スキャンモード: LiDAR スキャン/フォトグラメトリ
- 3次元モデルエクスポート: FBX,OBJ,GLB,USDA, STL, PLY, LAS

LiDAR センサを非搭載の iPhone/iPad でも、iOS 14.0 以降、および A12 Bionic チップ以降を搭載したデバイスをサポートする、Scaniverse 2.0 が 2022/9/15 に発表されました。Scaniverse 2.0 は独自開発の ManyDepth という技術を使って 3次元モデルを生成することが可能だそうです。

- Scaniverse 2.0 adds support for non-LiDAR devices(<https://blog.scaniverse.com/scaniverse-2-0-adds-support-for-non-lidar-devices-e5b22b6b3300>)

Android ユーザの方は、WIDAR(<https://widar.io/>)を使ってフォトグラメトリで 3 次元モデルを作成してください。こちらも無料で利用可能です

Scaniverse の使用方法の詳細はこちらに参考にしてください。

※ Scaniverse を使ってスマホで 3 次元スキャンをしよう！

(<https://note.com/iwamah1/n/nc8a5427157ef>)

## 点群データ (PLY ファイル) を Open3D と Matplotlib で可視化

Google Colaboratory で点群データ(PLY ファイル)を可視化してみましょう。

3 次元モデルデータは、スマートフォンアプリから、PLY(点群データ)と OBJ(メッシュデータ)ファイルをエクスポートしてください。

### Open3D をインストール

Open3D(<http://www.open3d.org/>)をインストールします。Open3D はオープンソースの 3 次元データ Python ライブラリです

Open3 次元インストール後にセッションを再起動してください

```
!pip install open3d
```

### Google ドライブをマウントします

Google ドライブをマウントして PLY ファイルをアップロードします

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

### Open3D と Numpy をインポートします

Open3D と Numpy を利用できるようにインポートします



```
import open3d as o3d
```

```
import numpy as np
```

点群ファイル（PLY）を読み込みます

```
cloud =
```

```
o3d.io.read_point_cloud("/content/drive/MyDrive/dataset/image_in/iruka.ply")
```

```
if cloud.is_empty(): exit()
```

Matplotlib をインポート

```
import matplotlib.pyplot as plt
```

```
from mpl_toolkits import mplot3d
```

データを np.array 形式に変換します

```
# Numpy の np.array 形式に変換します
```

```
points = np.asarray(cloud.points)
```

```
# 色データを取得します
```

```
colors = np.asarray(cloud.colors)
```

Matplotlib の表示サイズを指定します

```
import matplotlib.pyplot as plt
```

```
from matplotlib.pyplot import figure
```

```
plt.rcParams["figure.figsize"] = [10,10] # 図の幅と高さをインチで指定する
```

matplotlib を使って点群を表示します

```
ax = plt.axes(projection='3d') # axes3 次元を作成
```

```
ax.view_init(0, 0) # 軸の仰角と方位を（ラジアンではなく）度単位で設定
```

```
ax.axis("off") # 軸表示を OFF
```

```
ax.scatter(points[:,0], points[:,1], points[:,2], s=1, c=colors) # 散布図にデ
```

ータ設定

```
plt.show() # データ表示
```



### 軸の仰角と方位を変更します

- 方位角(ほういかく: azimuth): 北を 0 度として時計回りの角度
- 仰角(ぎょうかく: elevation angle): 水平からその角度分だけ上に向いた角度

```
ax = plt.axes(projection='3d')
```

```
ax.view_init(elev=90, azim=-45) # 軸の仰角と方位を（ラジアンではなく）度単位
```

で設定

```
ax.axis("off")
```

```
ax.scatter(points[:,0], points[:,1], points[:,2], s=1, c=colors)
```

```
plt.show()
```



## メッシュデータ (OBJ ファイル) を Open3d と Plotly で可視化

メッシュデータを Plotly(<https://plotly.com/python/>)を使って可視化します。

Plotly はオープンソース の Python グラフ描画ライブラリでインタラクティブに動かすことが可能になります。

### メッシュタイプ (OBJ ファイル) のデータを読み込み

```
mesh =
```

```
o3d.io.read_triangle_mesh("/content/drive/MyDrive/dataset/image_in/iruka.obj")
```

```
if mesh.is_empty(): exit()
```

## 法線を計算

```
mesh.compute_vertex_normals() # 頂点の法線を計算
```

```
mesh.compute_triangle_normals() # 三角形の法線を計算
```

## 頂点と三角形のデータを *np.array* 形式に変換

```
triangles = np.asarray(mesh.triangles)
```

```
vertices = np.asarray(mesh.vertices)
```

## 色を設定

三角形の法線データによって色を変化させる

```
colors = (0.5, 0.5, 0.5) + np.asarray(mesh.triangle_normals) * 0.5
```

```
colors = tuple(map(tuple, colors))
```

## *Plotly* で表示

*Plotly* をインストール

```
!pip install plotly
```

*Plotly* をインポート

```
import plotly.graph_objects as graph_objects
```

*Plotly* で表示。出力結果はマウスでインタラクティブに動かすことが可能です

```
fig = graph_objects.Figure(
```

```
    data=[
```

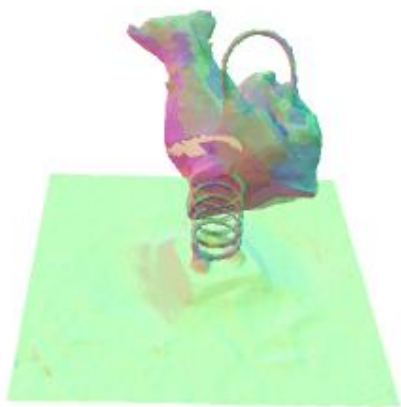
```
        graph_objects.Mesh3d(
```



```

        x=vertices[:,0], # 頂点の X 座標
        y=vertices[:,1], # 頂点の Y 座標
        z=vertices[:,2], # 頂点の Z 座標
        i=triangles[:,0], # 三角形の 1 番目の座標
        j=triangles[:,1], # 三角形の 2 番目の座標
        k=triangles[:,2], # 三角形の 3 番目の座標
        facecolor=colors, # 面の色を設定
        opacity=0.50)    # 表面の不透明度を設定
    ],
    layout=dict(
        scene=dict(
            xaxis=dict(visible=False), # X 軸非表示
            yaxis=dict(visible=False), # Y 軸非表示
            zaxis=dict(visible=False) # Z 軸非表示
        )
    )
)
fig.show() # 表示

```



## MMS の点群データ（Las ファイル）を Leafmap で可視化

MMS の点群データを可視化します。

可視化する MMS の点群データは、G 空間情報センター (<https://www.geospatial.jp/>) にある「VIRTUAL SHIZUOKA 静岡県 富士山南東部・伊豆東部 点群データ」をダウンロードして使用します。

このデータは、静岡県が CC BY 4.0/ODbL のデュアルライセンスのオープンデータとして公開されているデータで、ライセンスに準拠することで自由に使用することができます。

## MMS データのダウンロード

新規ユーザー登録 ログイン

G空間情報センター

データセット / 組織 / カテゴリ / アプリ

/ 組織 / 静岡県 / VIRTUAL SHIZUOKA 静岡県 ... / MMSデータ ダウンロードページ

### MMSデータ ダウンロードページ

URL: <https://gic-shizuoka.s3-ap-northeast-1.amazonaws.com/2020/Vectortile/MMS00/{z}/{x}/{y}.pbf>

移動計測車両（MMS）で取得した3次元点群データのダウンロードページです。

ダウンロード方法は次の通りです。

1. 地図を拡大、縮小し、ダウンロードしたいメッシュをクリックします。
2. ダウンロードをクリックすると指定したメッシュのデータのダウンロードが開始されます。

**注意**

- ファイルのサイズが数100MBから数GBあります。
- 通信状況によってはダウンロードに時間がかかる場合があります。

プレビュー

MMS データはダウンロードページ

(<https://www.geospatial.jp/ckan/dataset/shizuoka-2019-pointcloud/resource/7b986b95-c5a0-4a14-823c-129de28ee2a8>)

にある、静岡県の河津七滝ループ橋の点群データをダウンロードします。

河津七滝ループ橋のデータは、MESH\_NO「080F4060」ですので、このメッシュ番号のデータをダウンロードしてください。

点群データは大きなサイズですのでダウンロードする場合は通信環境の良い場所でダウンロードしてください。

ダウンロードするファイルは、圧縮したファイルで 08OF4060.7z (1.80 GB)となります。  
ファイルは解凍して 08OF4060.las (6.23 GB)となります。

### Leafmap, Open3d をインストール

Leafmap(<https://leafmap.org/>)は、Jupyter 環境や Google Colaboratory でインタラクティブなマッピングと地理空間分析を行うための Python パッケージです。

インストール後にセッションを再起動してください。

```
!pip install leafmap[lidar] open3d
```

### Google ドライブをマウントします

```
from google.colab import drive  
drive.mount('/content/drive')
```

### Open3d と Numpy をインポートします

```
import os  
import leafmap  
import open3d as o3d  
import numpy as np
```

### Leafmap で点群データ (las ファイル) を読み込みます。

Las ファイルは大きなファイルなので、読み込み時間は約 1 分 15 秒くらいかかります

```
%%time # 時間計測  
las =
```



```
leafmap.read_lidar("/content/drive/MyDrive/dataset/image_in/08OF4060.  
las")
```

点群のポイント数を確認

```
las.header.point_count
```

ポイント数は約 1.9 億(196,844,561)ポイントあります

Matplotlib をインポート

```
import matplotlib.pyplot as plt  
from mpl_toolkits import mplot3d
```

データを 3 次元配列に変換

Leafmap のデータから 3 次元配列を作成します

```
points = np.vstack((las.X, las.Y, las.Z)) # 3 次元座標  
colors = np.vstack((las.red, las.green, las.blue)) # RGB 情報
```

軸の順番を入れ替え

```
points = points.transpose()  
colors = colors.transpose()
```

## データを間引

データを間引きしてデータを小さくします

```
# データを 1/100 に間引く
```

```
step = 100
```

```
decimated_points = points[::step]
```

```
decimated_colors = colors[::step] / 65535.0 #データを 0.0-1.0 で正規化します
```

## データのサイズを確認

```
len(decimated_points)
```

ポイント数は約 190 万(1,968,446)ポイントに小さくなりました

## 点データ確認

実際の点データ確認します

```
decimated_points
```

## 色情報確認

RGB の色情報を確認します

```
decimated_colors
```

## Matplotlib 設定

Matplotlib の表示設定をします

```
import matplotlib.pyplot as plt
```

```
from matplotlib.pyplot import figure
```

```
plt.rcParams["figure.figsize"] = [15,15] # 図の幅と高さをインチで指定する
```

*Matplotlib* で点群表示

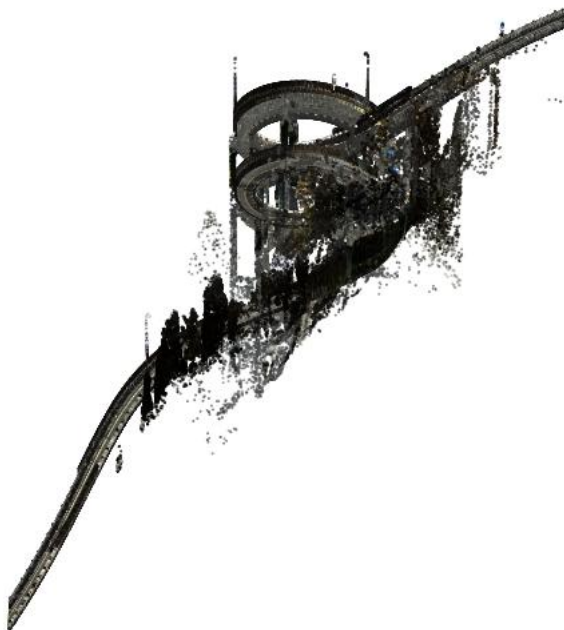
```
ax = plt.axes(projection='3d ')
```

```
ax.view_init(elev=45, azim=0)
```

```
ax.axis("off")
```

```
ax.scatter(decimated_points[:,0], decimated_points[:,1],  
decimated_points[:,2], s=1, c=decimated_colors)
```

```
plt.show()
```



## PyPotree で可視化

PyPotree(<https://github.com/centreborelli/pypotree>)で可視化をします。

PyPotree は点群に対応した WebGL ベースのオープンソースの点群レンダラーである Potree を Jupyter 環境や Google Colaboratory で動作することが出来るようにしたものです。

PyPotree をインストール

```
!pip install pypotree
```

PyPotree をインポート

```
import pypotree
```

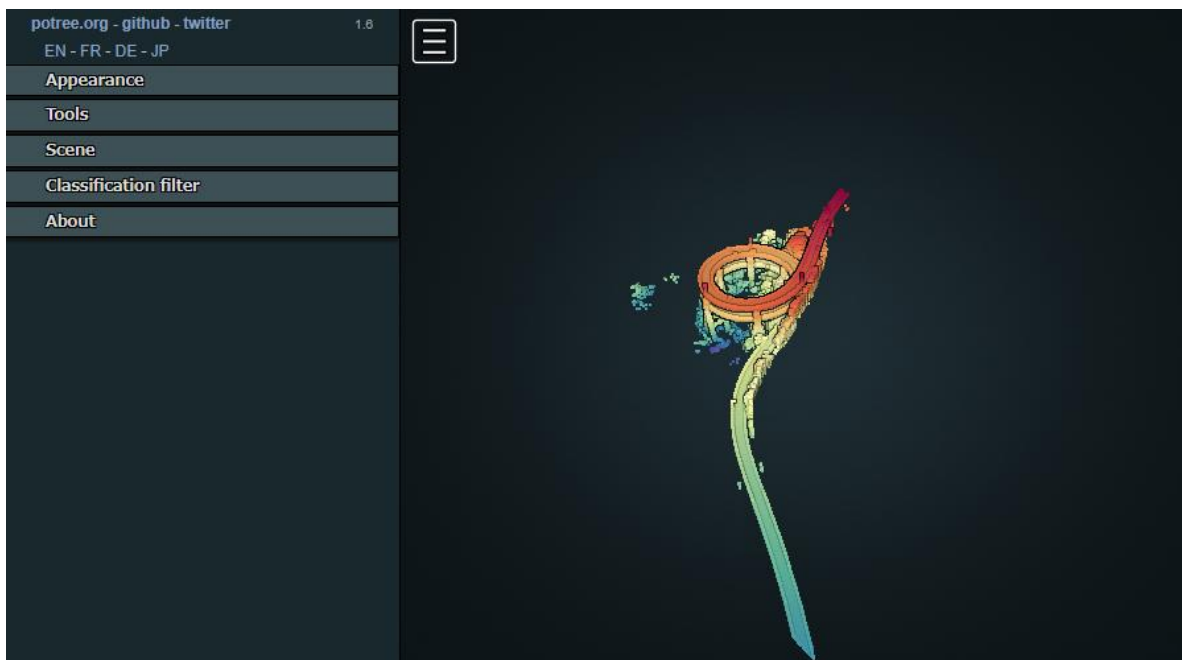
PyPotree で表示

```
# PotreeViewer を用いて 3 次元点群を表示するための設定
```

```
cloudpath = pypotree.generate_cloud_for_display(decimated_points)
```

```
# generate_cloud_for_display で生成したパスに点群を表示
```

```
pypotree.display_cloud_colab(cloudpath)
```



## 参考コード

- 技術的特異点 - 「open3 次元」一覧
  - <http://tecsingularity.com/category/open3次元/>
- Open3d Visualize in Google Colab
  - [https://colab.research.google.com/drive/1CR\\_HDvJ2AnjJV3Bf5vwP70K0hx3RcdMb?usp=sharing](https://colab.research.google.com/drive/1CR_HDvJ2AnjJV3Bf5vwP70K0hx3RcdMb?usp=sharing)