# Tìm hiểu khái niệm LiDAR

## Lidar là gì ?

**Lidar** (viết tắt của *Light Detection and Ranging*): Là phương pháp cảm biến từ xa sử dụng laser để đo độ cao, khoảng cách của các đối tượng như mặt đất, rừng và tòa nhà,….. từ đó tạo ra bản đồ không gian 2D hoặc mô hình 3D của môi trường xung quanh.

Lidar hoạt động bằng cách sử dụng nguồn sáng cực tím khả kiến hoặc gần hồng ngoại. Lidar gửi ra một xung ánh sáng (Pulse), chờ đợi sự trở về (Return) và Thời gian mà xung laser mất để quay trở lại được sử dụng để tính toán khoảng cách tới đối tượng theo công thức: d = t \* c /2

Với:

c = tốc độ ánh sáng (299,792,458 m/s)

t = thời gian di chuyển (travel time)

d = khoảng cách (distance)

**Một số đặc điểm nổi bật:**

* Cho phép xây dựng mô hình đám mây điểm (Point cloud) với độ chính xác cao.
* Hoạt động độc lập với điều kiện ánh sáng tự nhiên (có thể hoạt động tốt cả ban ngày lẫn ban đêm).
* Được ứng dụng rộng rãi trong nhiều lĩnh vực như hệ thống thông tin địa lý (GIS), ô tô tự lái, robotics, khảo sát và viễn thám.

## 1.2. Các kiểu LiDAR

### 1.2.1. Phân loại theo ứng dụng và môi trường thu thập:

* LiDAR Địa hình (Topographic LiDAR): Sử dụng tia laser gần hồng ngoại để khảo sát mặt đất, đánh giá độ cao và xây dựng mô hình số độ cao (DEM) hay mô hình số bề mặt (DSM)
* LiDAR Metric: Sử dụng ánh sáng xanh để đo đạc độ sâu trong các môi trường dưới nước (đo độ sâu lòng sông hoặc biển).
* LiDAR Trên Cạn (Terrestrial LiDAR): Được lắp đặt cố định trên mặt đất, dùng để đo đạc các tòa nhà, đặc điểm tự nhiên và thiết lập mô hình 3D của các di tích lịch sử.
* LiDAR trên Không (Airborne LiDAR): Gắn trên máy bay, UAV hay drone; tích hợp các thành phần như cảm biến LiDAR, thiết bị định vị GPS và đơn vị đo vận tốc quán tính (IMU) để xác định chính xác vị trí, hướng và độ cao trong quá trình khảo sát quy mô rộng.

### 1.2.2. Phân loại theo công nghệ cảm biến và phương pháp quét

* LiDAR Quay Cơ Học:
  + Sử dụng dãy laser được sắp xếp theo chiều dọc, quay 360° để quét môi trường xung quanh.
  + Điểm hạn: mật độ đám mây điểm giảm theo khoảng cách.
* LiDAR Hybrid – Solid-State
  + Kết hợp gương quay đa giác và gương dao động để quét theo cả chiều ngang và dọc
  + Ưu điểm: có thể quét khung hình bằng một chùm laser duy nhất nhưng đòi hỏi tốc độ quét rất cao.
* Công Nghệ MEMS (Micro-Electro-Mechanical Systems)
  + Sử dụng gương quét cấp micrometer (siêu nhỏ) điều khiển bằng thanh cân treo.
  + Nhược điểm: giới hạn về góc quay và độ bền, dẫn đến trường nhìn hẹp và biến dạng ở biên.
* LiDAR ASIC: Tích hợp hàng trăm kênh laser trên một chip, giúp giảm kích thước, chi phí vận hành và tăng độ phân giải cũng như độ chính xác của dữ liệu thu thập (đám mây điểm). => Cân bằng giữa hiệu suất và độ tin cậy

## 1.3. Cấu tạo LiDAR

Một hệ thống LiDAR hiện đại thường bao gồm các thành phần sau:

### 1.3.1. Thành phần chính của hệ thống LiDAR

Bộ Phát Laser (Laser Scanner):

* Chức năng: Phát ra các xung laser
* Các thông số quan trọng: độ chính xác, mật độ điểm (resolution), phạm vi quét và tốc độ quét.

Bộ Thu (Receiver):

* Chức năng: Ghi nhận ánh sáng phản xạ từ các đối tượng xung quanh và truyền tín hiệu về cho hệ thống xử lý.

Bộ Xử Lý (Processor):

* Chức năng: Tính toán thời gian bay (TOF / Time of Flight) của các xung laser và chuyển đổi kết quả thành dữ liệu dạng đám mây điểm (Point Cloud) để dựng mô hình 3D.

### 1.3.2. Hệ thống định vị và định hướng

GNSS (Global Navigation Satellite System):

* Cung cấp thông tin vị trí, vĩ độ, kinh độ và độ cao chính xác của hệ thống.

IMU (Inertial Measurement Unit):

* Xác định hướng, tốc độ và gia tốc của nền tảng di động (máy bay, drone, ô tô).
* Giúp hiệu chỉnh các sai lệch do rung lắc và góc quay không vuông (off nadir).

### 1.3.3. Các thành phần bổ trợ khác

Máy tính và hệ thống lưu trữ dữ liệu: Xử lý và lưu trữ dữ liệu thu thập được, thường sử dụng định dạng dữ liệu LAS.

Hệ thống điều khiển: Quản lý quá trình quét và đồng bộ giữa các thành phần của hệ thống

## 1.4. Nguyên tắc hoạt động của LiDAR

Bước 1: Phát xạ xung laser

Hệ thống LiDAR phát ra một chùm tia laser ngắt quãng (pulse) theo hướng xác định.

Tia laser này có thể nằm trong các dải khả kiến, cực tím hoặc gần hồng ngoại.

Bước 2: Phản xạ (Return)

Khi tia laser va chạm với đối tượng (đất, cây cối, tòa nhà, con người, …), nó sẽ phản xạ trở lại hệ thống.

Các đối tượng có đặc tính phản xạ khác nhau (albedo) cho tín hiệu trả về có cường độ khác nhau.

Bước 3: Ghi nhận và Tính toán khoảng cách

Thời gian bay (Time-of-Flight – TOF):

Hệ thống đo thời gian từ lúc phát ra đến khi nhận lại tia laser.

Công thức tính khoảng cách:

**d = (t × c) / 2**  
giải thích: tia laser phải đi đến đối tượng và quay về nên thời gian tính được được chia đôi.

Quá trình này được lặp lại hàng triệu lần mỗi giây, thu thập được dữ liệu dạng “đám mây điểm” (Point Cloud) mô tả chi tiết không gian 3D.

Hệ thống cũng cần hiệu chỉnh các yếu tố như rung lắc, góc phát xung không vuông (off nadir) để đảm bảo độ chính xác cao.

Bước 4: Xử lý và Hiển thị dữ liệu

Dữ liệu thu được được xử lý bởi bộ xử lý (processor) và có thể kết hợp với dữ liệu hình ảnh (panchromatic, multispectral) để tạo ra các mô hình số bề mặt (DSM), địa hình (DEM) với chi tiết cao.

Hệ thống phân loại và tô màu dữ liệu LiDAR dựa trên đặc tính nhận dạng (theo tiêu chuẩn của American Society for Photogrammetry and Remote Sensing) để mô phỏng cảnh quan thực tế (màu xanh cho cây xanh, màu nâu cho đất, màu xám cho công trình).

# 2.Tìm hiểu dữ liệu Point Cloud

## 2.1. Khái niệm Point Cloud

**Point Cloud** là tập hợp các điểm dữ liệu trong không gian 3 chiều, mỗi điểm được xác định bởi tọa độ (x, y, z) và có thể kèm theo thông tin phụ như cường độ ánh sáng hoặc màu sắc.

2.2. Cách biểu diễn Voxel

Voxel là viết tắt của “volume pixel” – tức là pixel trong không gian 3 chiều.

2.3 Thư viện Open3DCác thư viện để trực quan hóa dữ liệu điểm như Point Cloud Library, Cloud Compare, MeshLab, Matlab, Autodesk Recap và đặc biệt là Open3D

Cấu trúc dữ liệu được **Open3D** sử dụng:

* Voxel Grid : được tạo hoặc bắt nguồn từ các Point Cloud
  + Voxel: là các điểm ảnh 3 chiều của hình ảnh 2D, viết tắt của điểm ảnh thể tích
* Octree: cấu trúc phân chia không gian 3D thành 8 phần (octants) để quản lý dữ liệu hiệu quả
* Meshes**:** lưới tam giác hoặc đa giác dùng trong đồ họa 3D.

Các định dạng dữ liệu phổ biến trong xử lý point cloud :

* .PLY**:** định dạng lưu trữ dữ liệu 3D dưới dạng tập hợp đa giác. Có thể chứa cả dữ liệu Point Cloud và Mesh
* .PCD**:** định dạng chuyên dụng cho dữ liệu point cloud
* .XYZ: biểu diễn các điểm trong không gian với tọa độ xyz ở định dạng văn bản

## 2.4. Cài đặt thư viện Open3D

Cài đặt thư viện hỗ trợ hiển thị ảo pyvirtualdisplay (thường dùng trên môi trường không có GUI)

!pip install pyvirtualdisplay  # Install the necessary library

!pip install open3d

import numpy as np

import matplotlib.pyplot as plt

import open3d as o3d

## 2.5. Đọc - Ghi dữ liệu Point Cloud

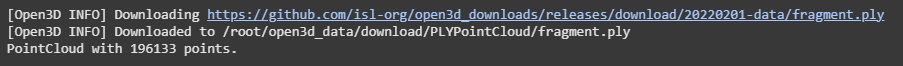
**Polygon Format(PLY)**

ply\_point\_cloud = o3d.data.PLYPointCloud() # Lấy dữ liệu mẫu PLY từ Open3D

ply = o3d.io.read\_point\_cloud(ply\_point\_cloud.path)  # Đọc file PLY

print(ply)

print(np.asarray(ply.points))  # Chuyển đổi danh sách điểm thành mảng NumPy để kiểm tra tọa độ



**[X, Y, Z]**

print(np.asarray(ply.points))

A screenshot of a computer screen

AI-generated content may be incorrect.

**PCD**

dataset =o3d.data.PCDPointCloud()

path='/content/drive/MyDrive/Khóa Luận/Data\_For\_Testing/ism\_test\_cat.pcd'

pcd\_01=o3d.io.read\_point\_cloud(path)

print(pcd\_01)

Sau khi xử lý hay chỉnh sửa, có thể ghi (lưu) dữ liệu Point Cloud sang các định dạng như PLY hay PCD:

o3d.io.write\_point\_cloud("output\_file.ply", ply)

## 2.6. Trực quan hóa dữ liệu Point Cloud

Sử dụng hàm draw\_plotly của Open3D để trực quan hóa điểm đám mây với các tham số điều chỉnh góc nhìn:

o3d.visualization.draw\_plotly([ply],

                                  zoom=0.2,

                                  front = [0.4257, -0.2125, -0.8795],

                                  lookat=[2.6172, 2.0475, 1.532],

                                  up=[-0.0694, -0.9768,0.2024]

                                  )

A chair and a desk in a room

AI-generated content may be incorrect.

points = np.asarray(pcd\_01.points)

center = np.mean(points, axis=0)

o3d.visualization.draw\_plotly([pcd\_01],

                                  zoom=0.7,

                                  front=[0.5439, -0.2333, -0.8060],

                                  lookat=[2.4615, 2.1331, 1.338],

                                  up=[-0.1781, -0.9708, 0.1608])

A red cat on a grid

AI-generated content may be incorrect.

## 2.7. Các thao tác cơ bản trên Point Cloud

### 2.7.1 Tô màu cho point cloud

Dùng phương thức paint\_uniform\_color để gán một màu duy nhất cho tất cả các điểm.

pcd\_01.paint\_uniform\_color([1, 0.706, 0])

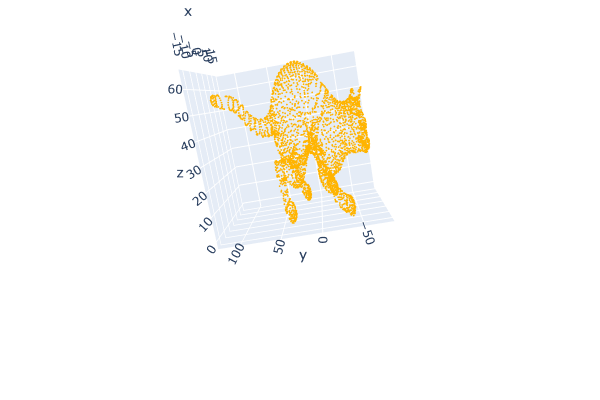
o3d.visualization.draw\_plotly([pcd\_01],

                                  zoom=0.7,

                                  front=[0.5439, -0.2333, -0.8060],

                                  lookat=[2.4615, 2.1331, 1.338],

                                  up=[-0.1781, -0.9708, 0.1608])



### 2.7.2 Kiểm tra thông tin điểm của point cloud

points = np.asarray(ply.points)

print("Số điểm:", len(ply.points))

print("Tọa độ mẫu (5 điểm đầu):", points[:5])

print("Min tọa độ:", np.min(points, axis=0))

print("Max tọa độ:", np.max(points, axis=0))

A screenshot of a computer

AI-generated content may be incorrect.

### 2.7.3 Giảm mẫu (Downsampling) sử dụng Voxel Downsampling

dataset =o3d.data.PCDPointCloud()

pcd=o3d.io.read\_point\_cloud(dataset.path)

print(pcd)

o3d.visualization.draw\_plotly([pcd],

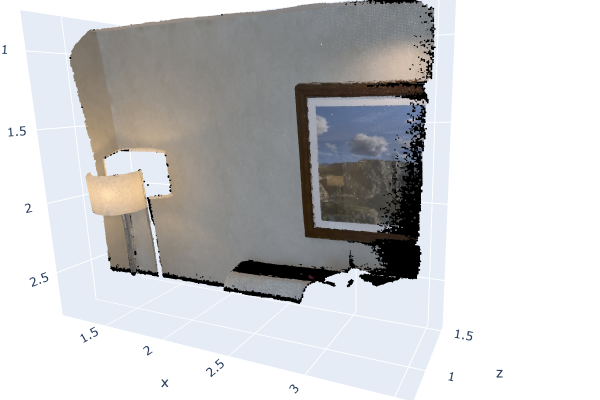
                                  zoom=0.3412,

                                  front = [0.4257, -0.2125, -0.8795],

                                  lookat=[2.6172, 2.0475, 1.532],

                                  up=[-0.0694, -0.9768,0.2024]

)



Dùng phương thức voxel\_down\_sample giảm số lượng điểm trong dữ liệu bằng cách tạo một lưới voxel có kích thước 0.03.

print("Downsample the point cloud with a voxel of 0.03")

downpcd=pcd.voxel\_down\_sample(voxel\_size=0.03)

o3d.visualization.draw\_plotly([downpcd],

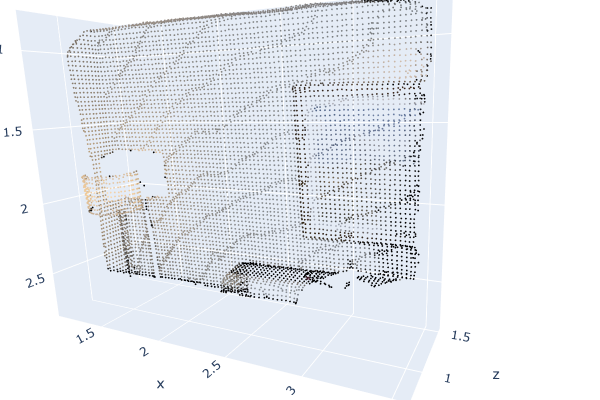
                               zoom=0.3412,

                                  front = [0.4257, -0.2125, -0.8795],

                                  lookat=[2.6172, 2.0475, 1.532],

                                  up=[-0.0694, -0.9768,0.2024]

                              )



### 3.7.4 Cắt dữ liệu (Cropping) và xử lý dữ liệu dựa trên vùng chọn

demo\_crop\_data=o3d.data.DemoCropPointCloud()

vol=o3d.visualization.read\_selection\_polygon\_volume(demo\_crop\_data.cropped\_json\_path)

chair = vol.crop\_point\_cloud(ply)

o3d.visualization.draw\_plotly([chair],

                               zoom=0.3412,

                                  front = [0.4257, -0.2125, -0.8795],

                                  lookat=[2.6172, 2.0475, 1.532],

                                  up=[-0.0694, -0.9768,0.2024]

                              )

A white chair with a wooden base

AI-generated content may be incorrect.

Tính khoảng cách từ từng điểm của toàn bộ dữ liệu với phần “ghế” vừa cắt.

Sau đó, xác định các điểm nằm xa “ghế” (khoảng cách > 0.01) và loại bỏ chúng khỏi tập dữ liệu ban đầu để tạo ra ply\_without\_chair (đám mây điểm không bao gồm ghế).

distance=ply.compute\_point\_cloud\_distance(chair)

dists=np.asarray(distance)

ind=np.where(dists> 0.01)[0]

ply\_without\_chair=ply.select\_by\_index(ind)

o3d.visualization.draw\_plotly([ply\_without\_chair],

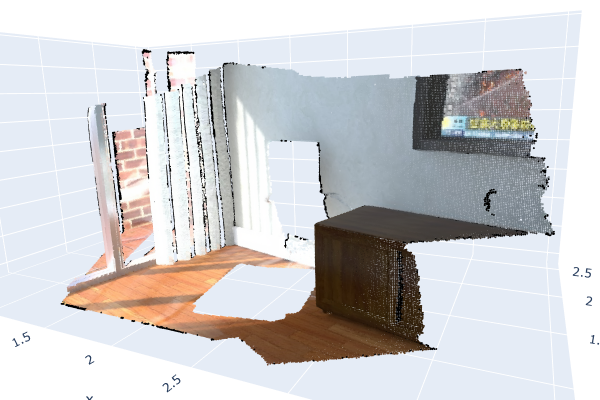
                               zoom=0.3412,

                                  front = [0.4257, -0.2125, -0.8795],

                                  lookat=[2.6172, 2.0475, 1.532],

                                  up=[-0.0694, -0.9768,0.2024]

                              )



### 3.7.5 Tính bao lồi (Convex Hull)

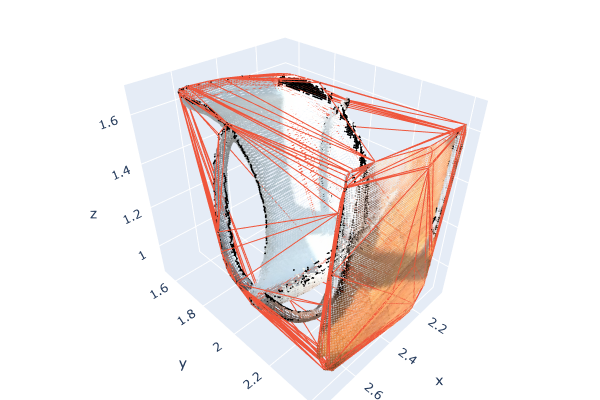
Xác định ranh giới hình học tối thiểu bao quanh dữ liệu đã chọn (bao quanh “ghế”)

hull, \_=chair.compute\_convex\_hull()

hull\_ls=o3d.geometry.LineSet.create\_from\_triangle\_mesh(hull)

hull\_ls.paint\_uniform\_color((0,1,1))

o3d.visualization.draw\_plotly([chair,hull\_ls])



### 3.7.6 Phân cụm (Clustering) với DBSCAN

Nhận diện các nhóm điểm (cluster) dựa trên mật độ điểm.

DBSCAN phân cụm các điểm dựa trên khoảng cách (eps) và số lượng điểm tối thiểu tạo thành một cụm.

with o3d.utility.VerbosityContextManager(

    o3d.utility.VerbosityLevel.Debug) as cm:

  labels = np.array(

      ply.cluster\_dbscan(eps=0.02, min\_points=10, print\_progress=True))

print(labels)

max\_label=labels.max()

print(f"point cloud has {max\_label + 1} clusters")

colors=plt.get\_cmap("tab20")(labels / (max\_label if max\_label > 0 else 1))

colors[labels < 0] = 0

ply.colors=o3d.utility.Vector3dVector(colors[:, :3])

o3d.visualization.draw\_plotly([ply],

                               zoom=0.3412,

                                  front = [0.4257, -0.2125, -0.8795],

                                  lookat=[2.6172, 2.0475, 1.532],

                                  up=[-0.0694, -0.9768,0.2024]

                              )

A screenshot of a computer screen

AI-generated content may be incorrect.