

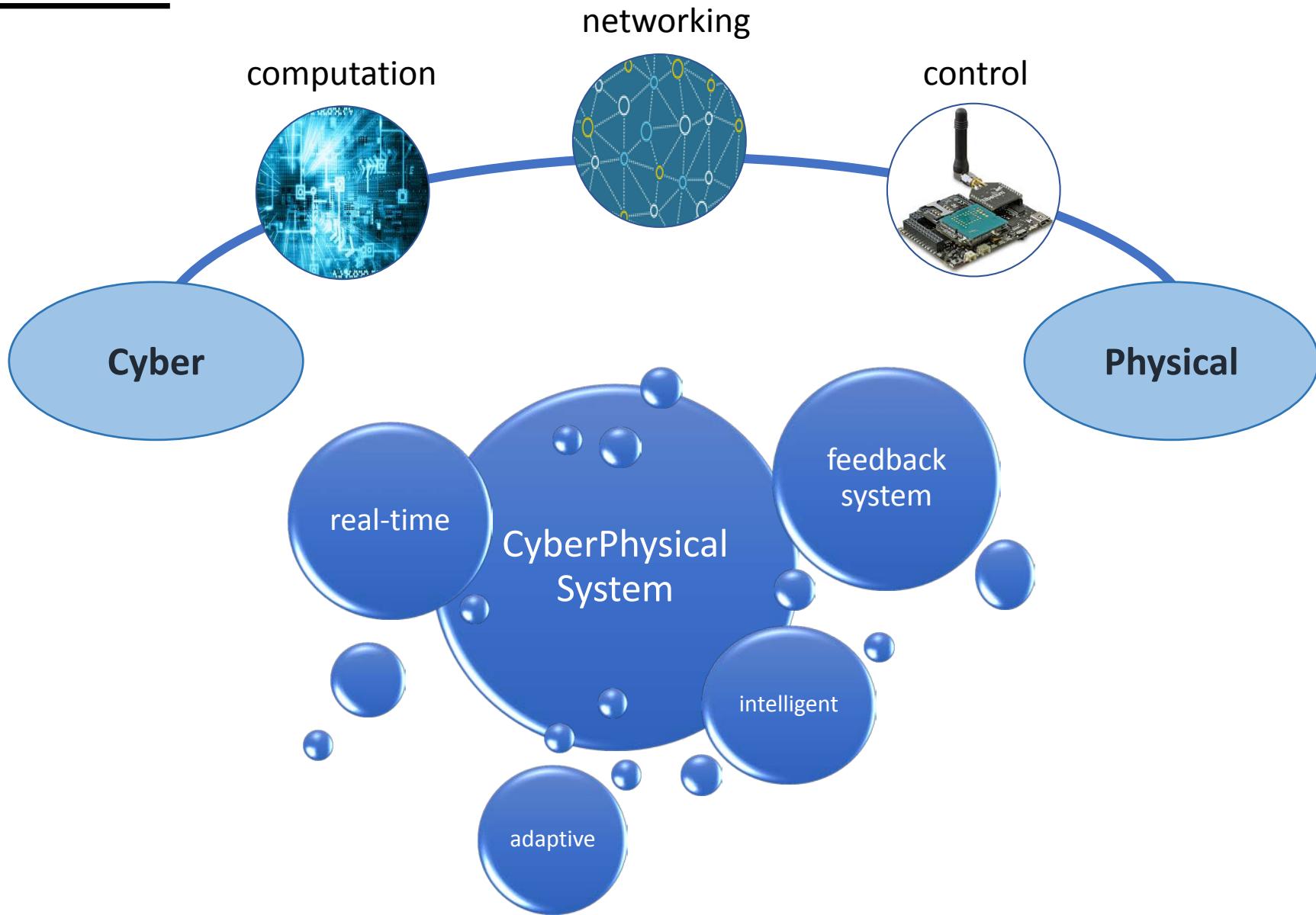


# LiDAR Data Processing For Automated Environments

Hannaneh Najdataei

.: October 2017 :.

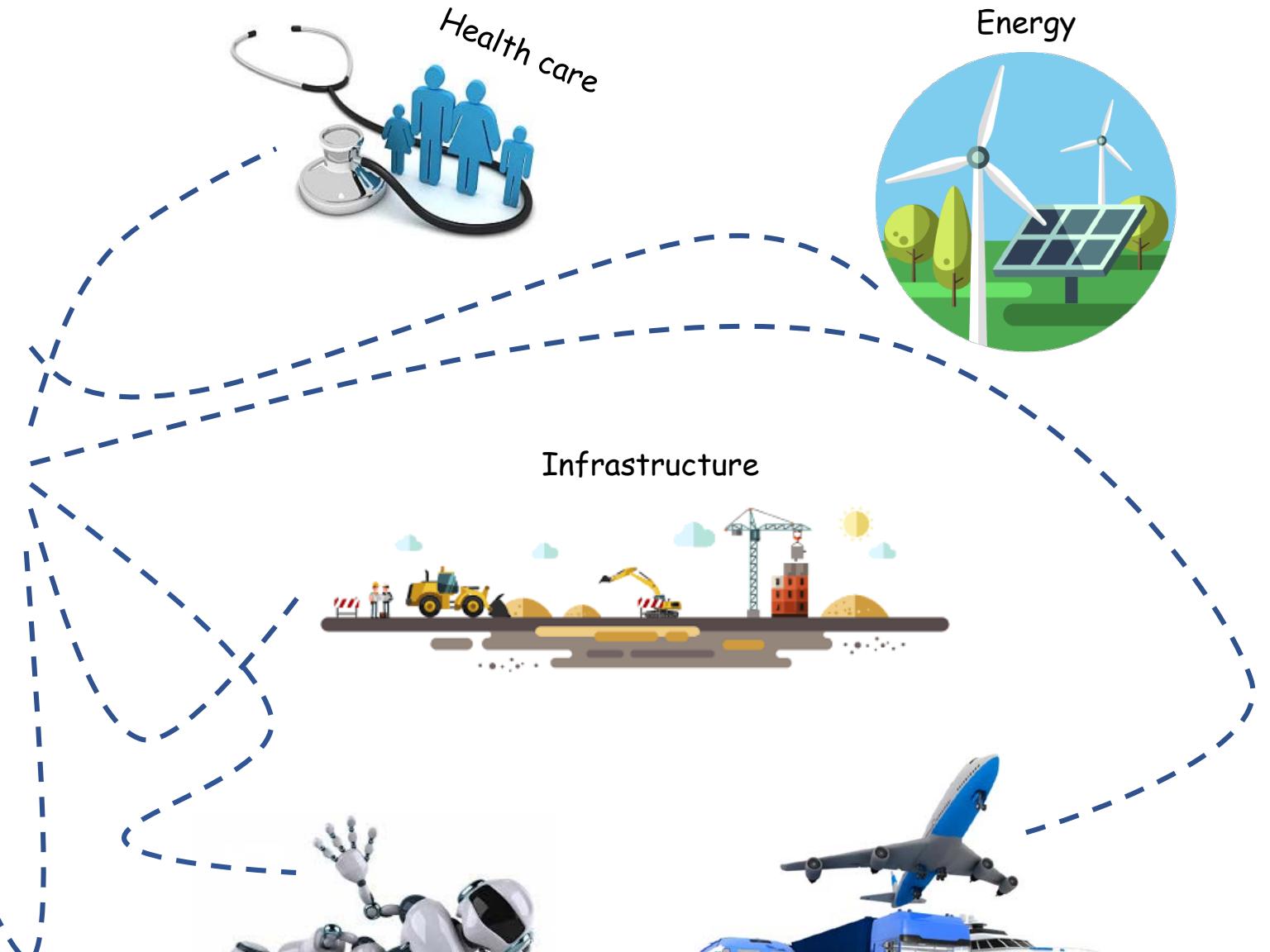
# Introduction



# CPS Applications



Manufacturing



Robotics



Transportation

*Health care*



Energy



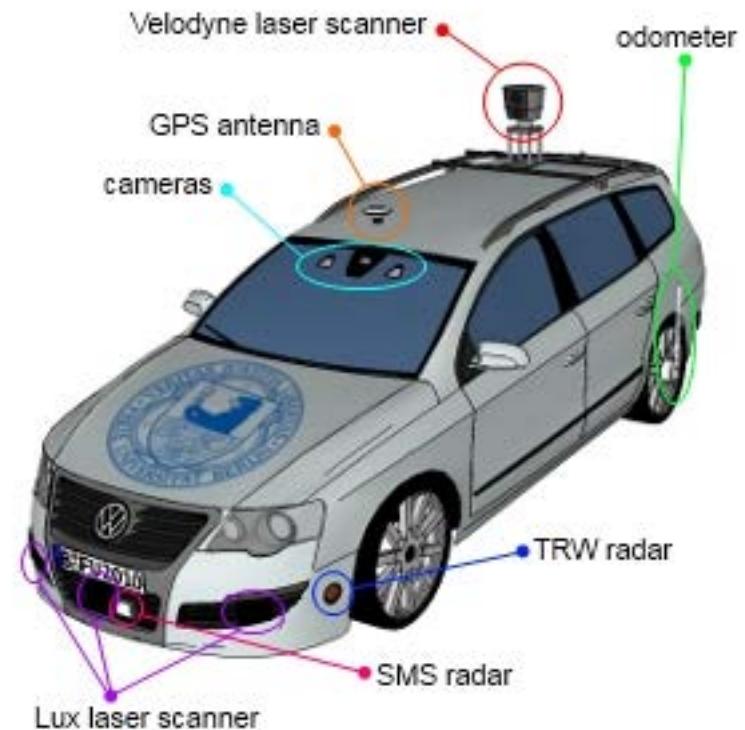
Infrastructure



# Automated Environment



✓ Makes life easier! and safer



Pic: robotland

# Outline

- LiDAR and Point Cloud
- Point Cloud Library
- Webots Simulator
- Use cases

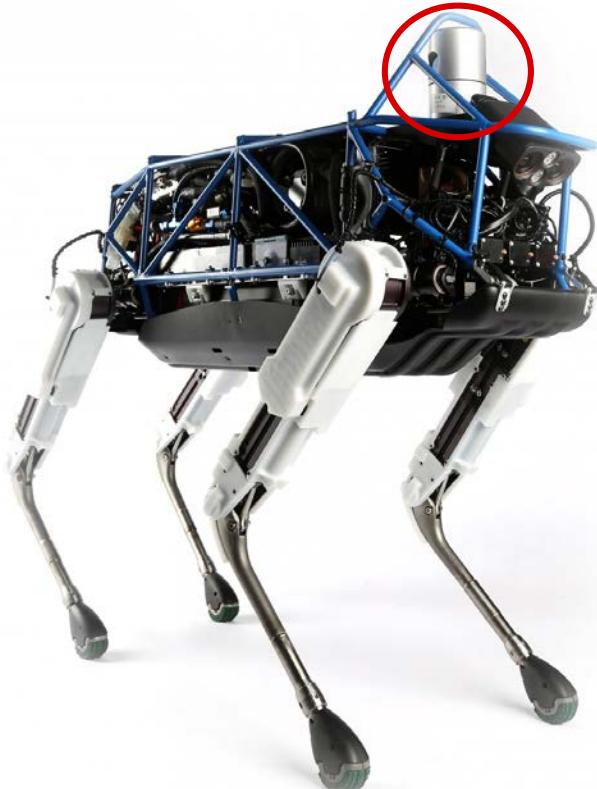


# LiDAR and Point Cloud

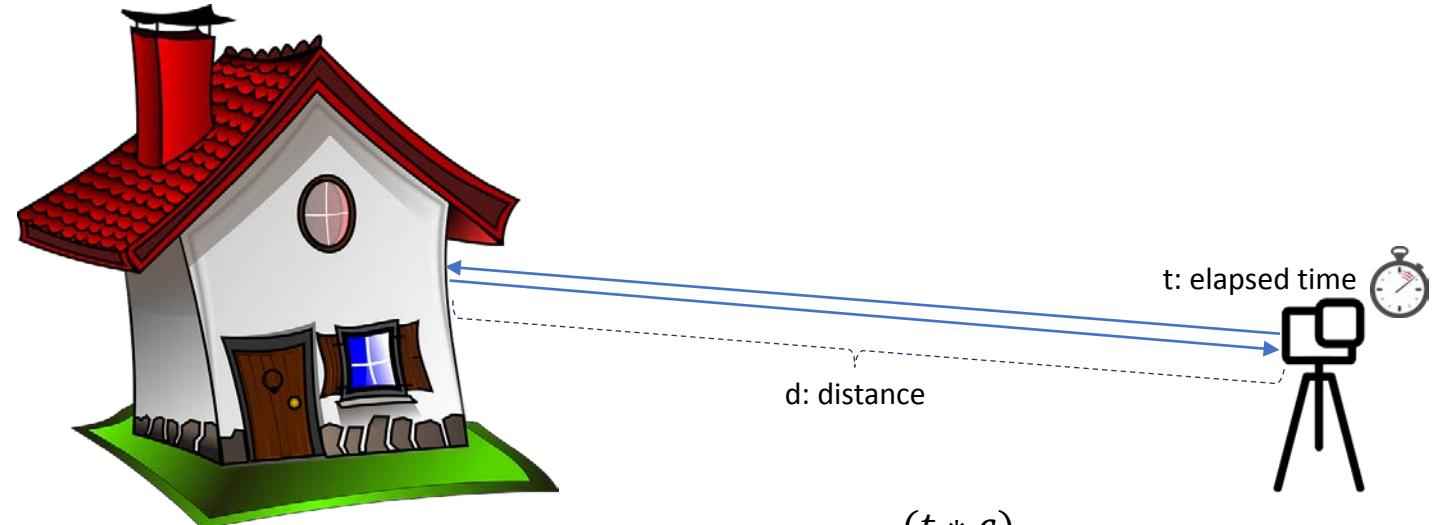
Introducing Spot



# LiDAR

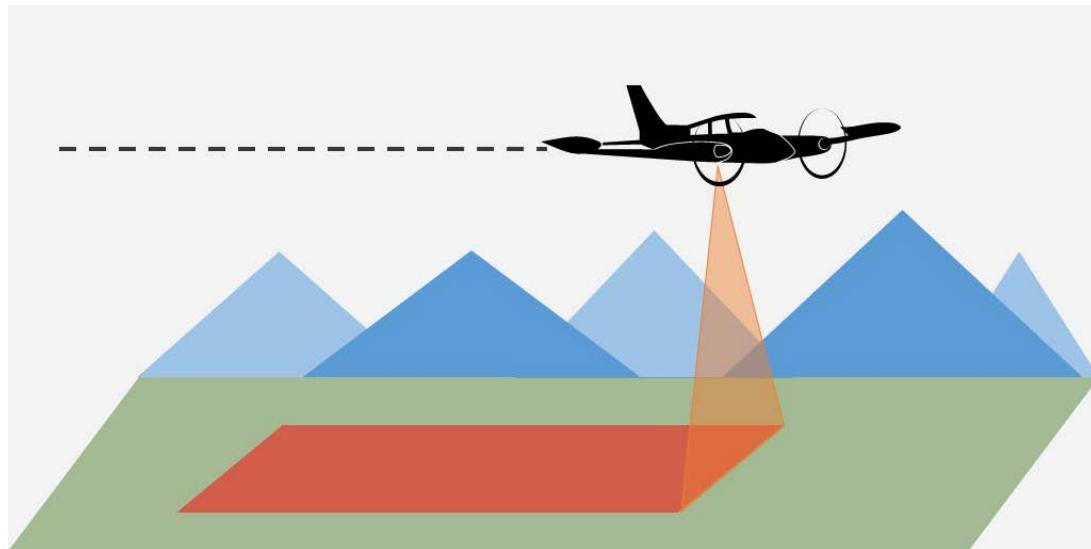


- Portmanteau word of *light* and *radar*
- Stands for Light Detection And Ranging

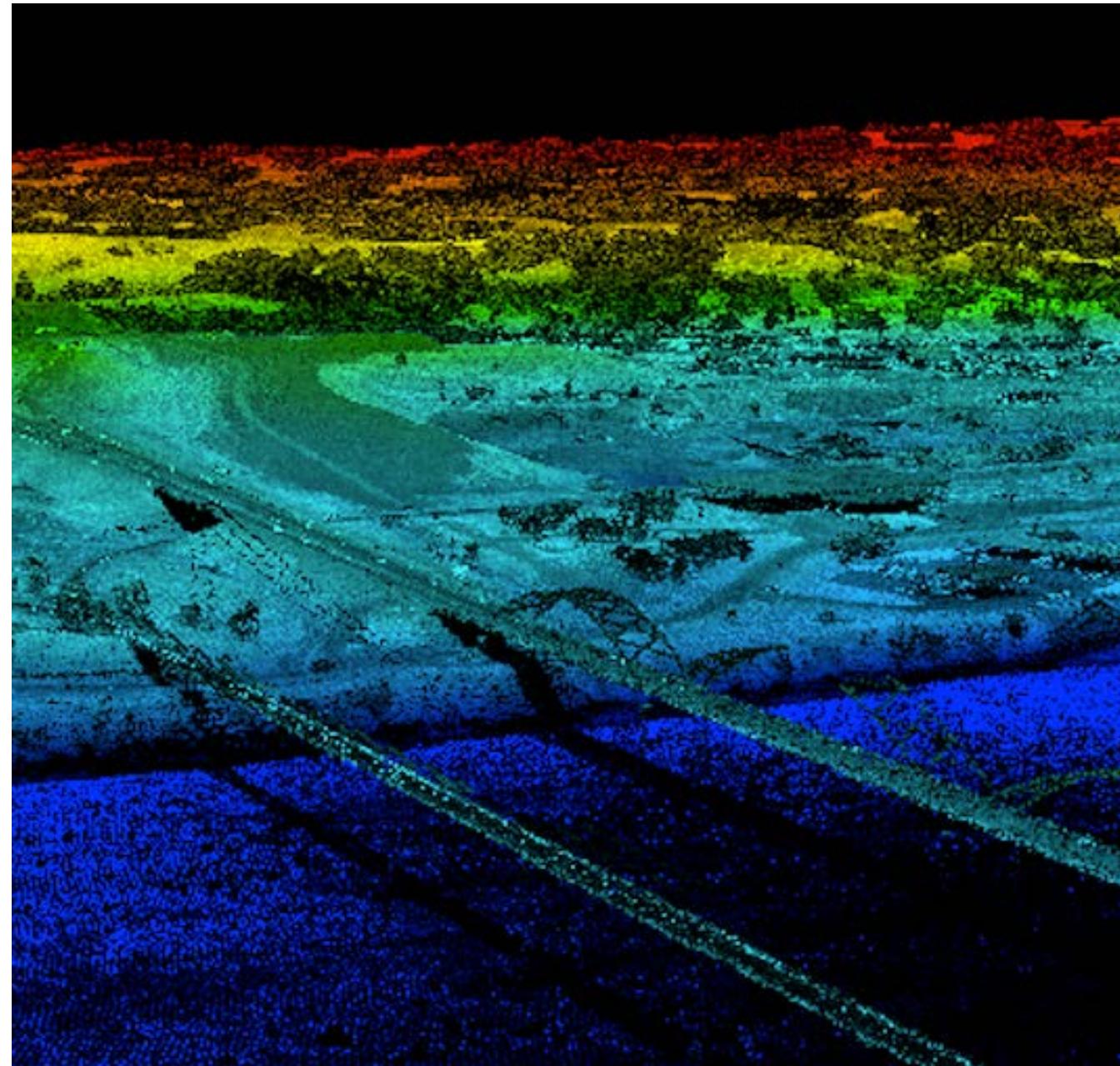
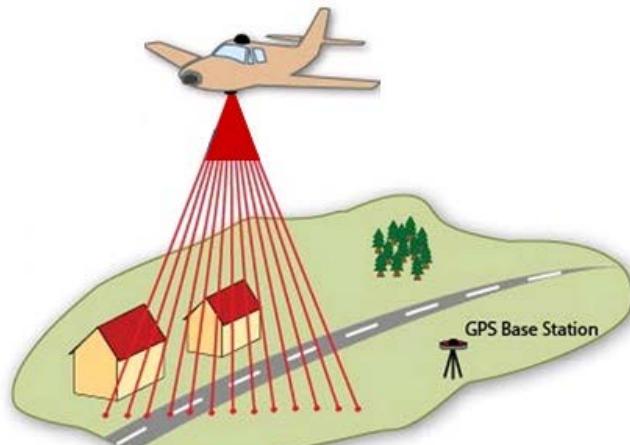


$$d = \frac{(t * c)}{2}$$

# Airborne LiDAR



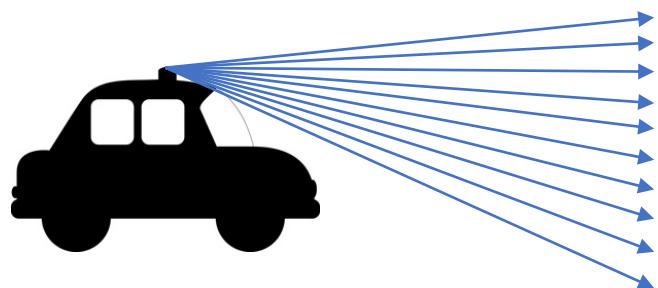
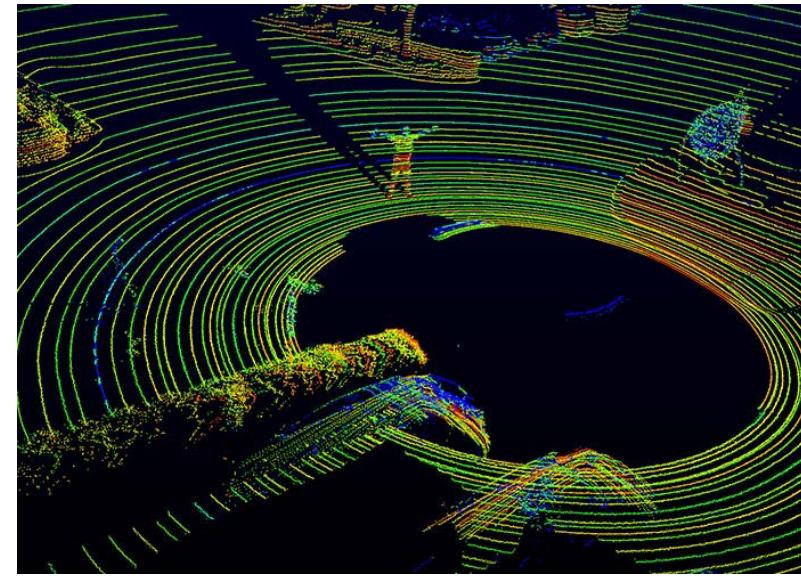
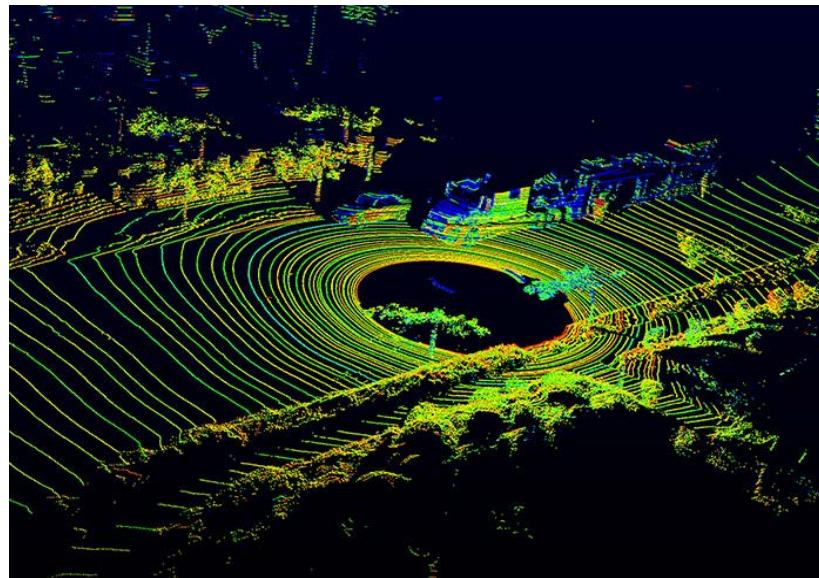
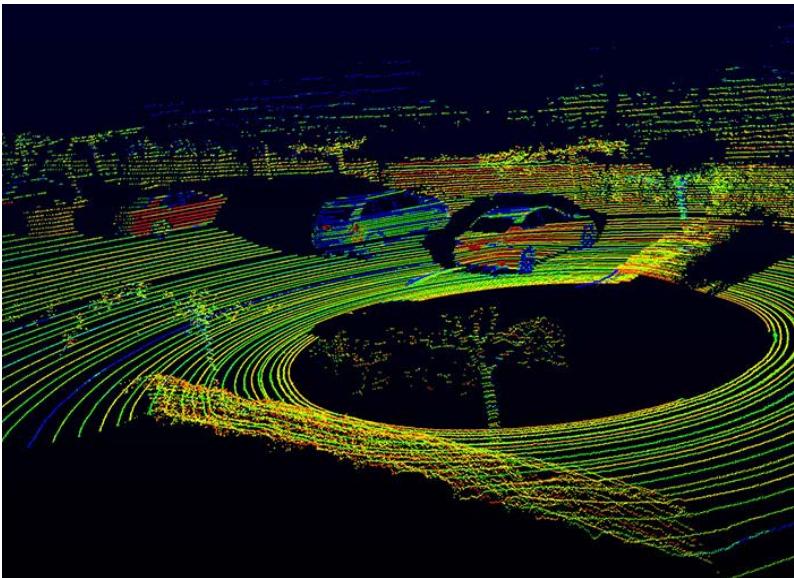
Airborne LiDAR – 3D mapping



Pic: [airborneimaginginc.com](http://airborneimaginginc.com)

# Rotating LiDAR

Pic: point cloud library



# Point Cloud Library



## What is PCL?

Learn more

Point cloud basics

Surface reconstruction

Visual hull

Robot 1: A white robot with blue highlights is kneeling and working on a yellow rectangular object.

Robot 2: A white robot with purple highlights is standing behind Robot 1, pointing towards the yellow object.

Background: A large black board with mathematical equations and diagrams related to point clouds and surface reconstruction.

## PCL features

Learn more

Initial point cloud data

Filtering

Segmentation

Surface reconstruction

Model fitting

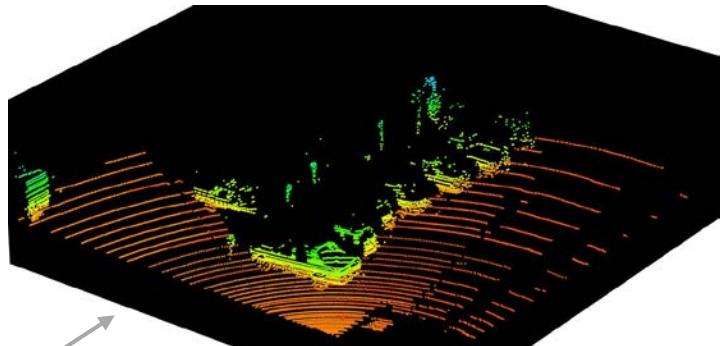
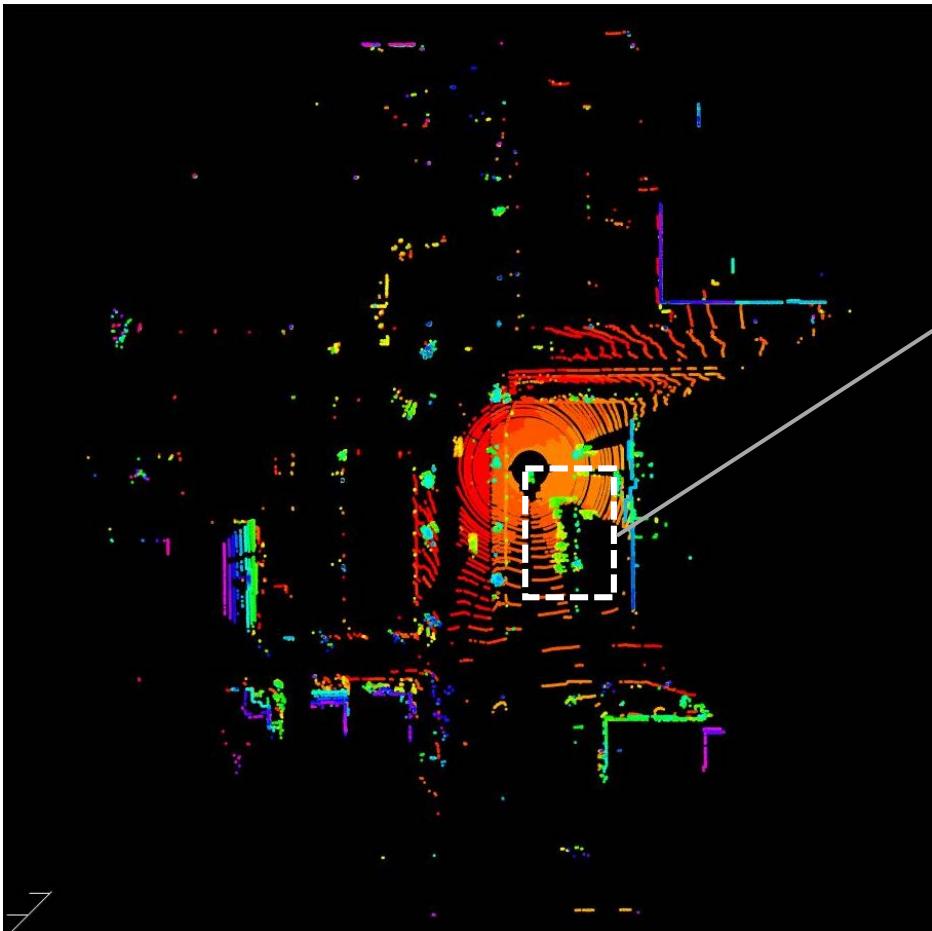
CITY

BUILDINGS

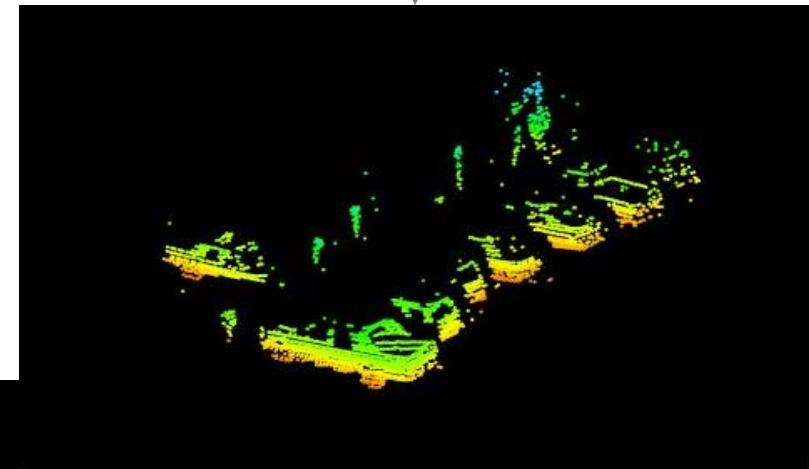
The background shows a dense city skyline represented as a point cloud. The skyline is divided into five vertical sections, each representing a different PCL feature:

- Initial point cloud data:** The leftmost section shows the raw, unprocessed point cloud of the city.
- Filtering:** The second section shows the point cloud after filtering, where certain points have been removed or highlighted.
- Segmentation:** The third section shows the point cloud segmented into individual buildings, each colored differently (e.g., green, blue, red, orange).
- Surface reconstruction:** The fourth section shows the 3D surfaces of the buildings reconstructed from the point cloud.
- Model fitting:** The rightmost section shows a wireframe model of the city buildings, representing a fitted model to the original point cloud.

# PCL - Segmentation



Removing the Ground



Clustering

# Euclidean Cluster Extraction

*PCL – EuclideanClusterExtraction(eps, minPts)*

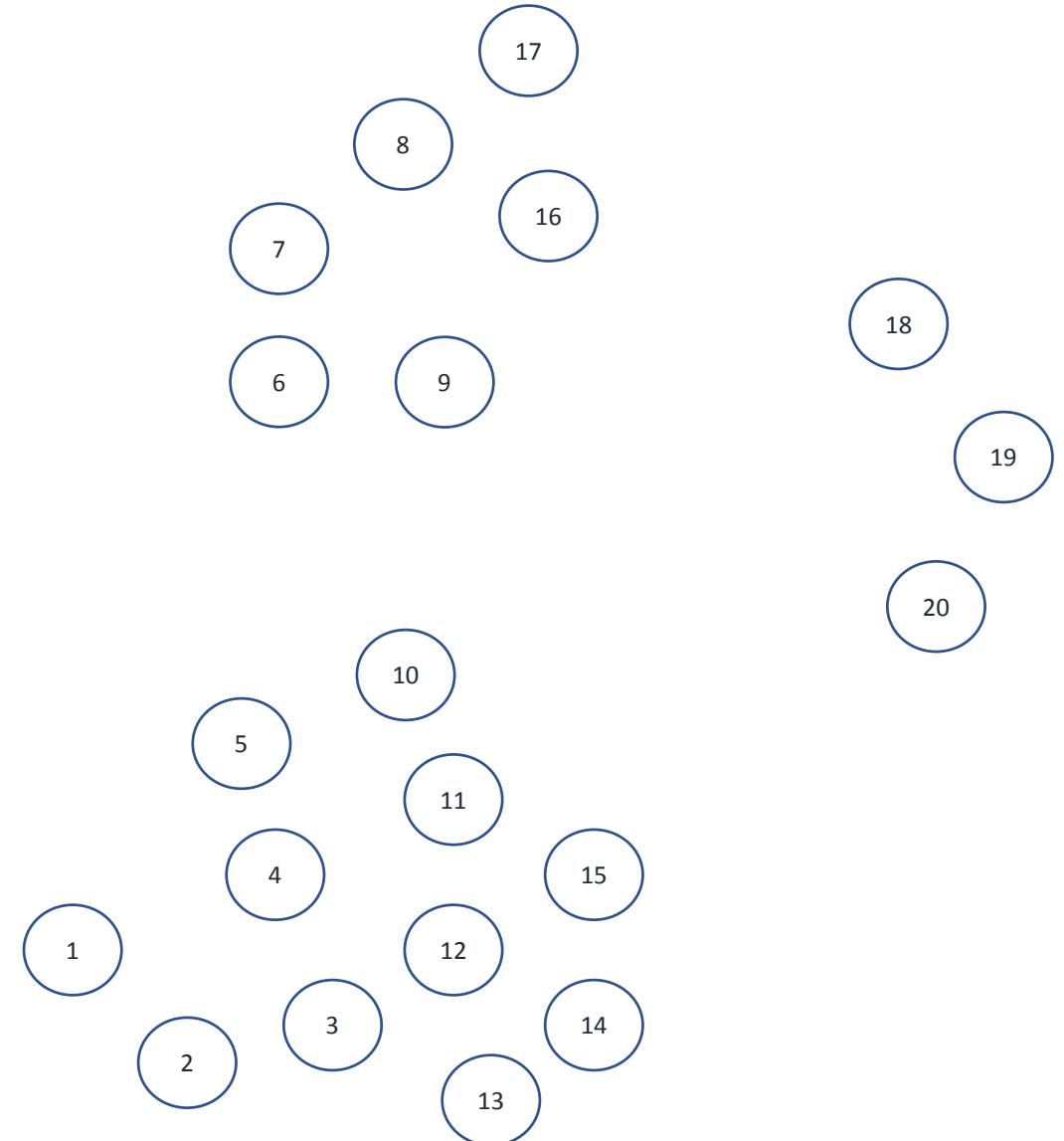
---

```
1: clusters =  $\emptyset$ 
2: for  $p \in P$  do
3:    $Q = \emptyset$ 
4:   if  $p.\text{status} \neq \text{processed}$  then
5:      $Q.\text{add}(p)$ 
6:     for  $q \in Q$  do
7:        $q.\text{status} = \text{'processed'}$ 
8:        $N = \text{GetNeighbors}(q, \epsilon)$ 
9:        $Q.\text{addAll}(N)$ 
10:      if  $\text{size}(Q) \geq \text{minPts}$  then clusters.add(Q)
```

---

$clusters = \{\}$

$Q = \{\}$



# Euclidean Cluster Extraction

*PCL – EuclideanClusterExtraction(eps, minPts)*

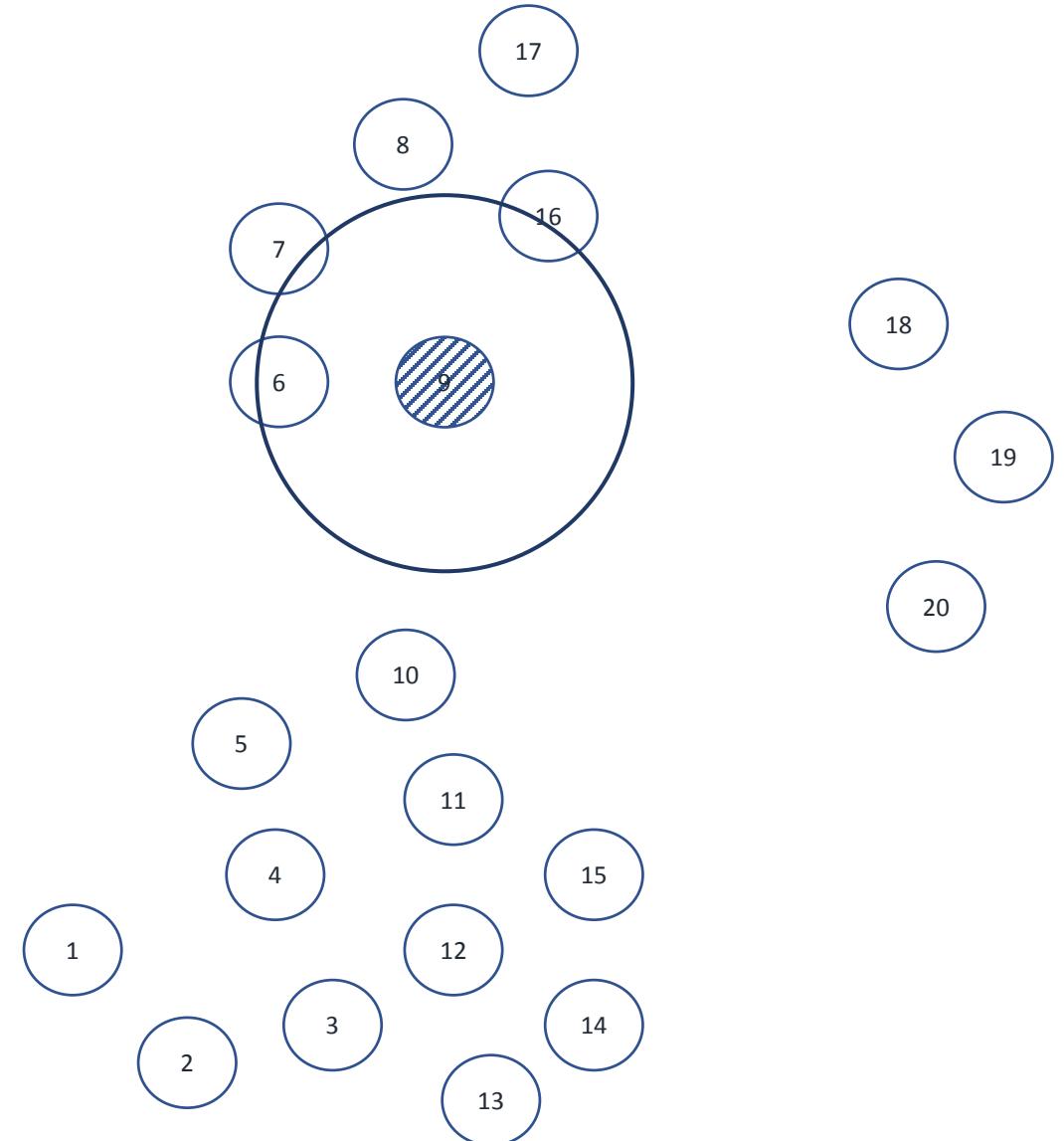
---

```
1: clusters =  $\emptyset$ 
2: for  $p \in P$  do
3:   Q =  $\emptyset$ 
4:   if  $p.\text{status} \neq \text{processed}$  then
5:     Q.add( $p$ )
6:     for  $q \in Q$  do
7:       q.status = 'processed'
8:        $N = \text{GetNeighbors}(q, \epsilon)$ 
9:       Q.addAll( $N$ )
10:      if size(Q)  $\geq \text{minPts}$  then clusters.add(Q)
```

---

$clusters = \{\}$

$Q = \{9\}$



# Euclidean Cluster Extraction

*PCL – EuclideanClusterExtraction(eps, minPts)*

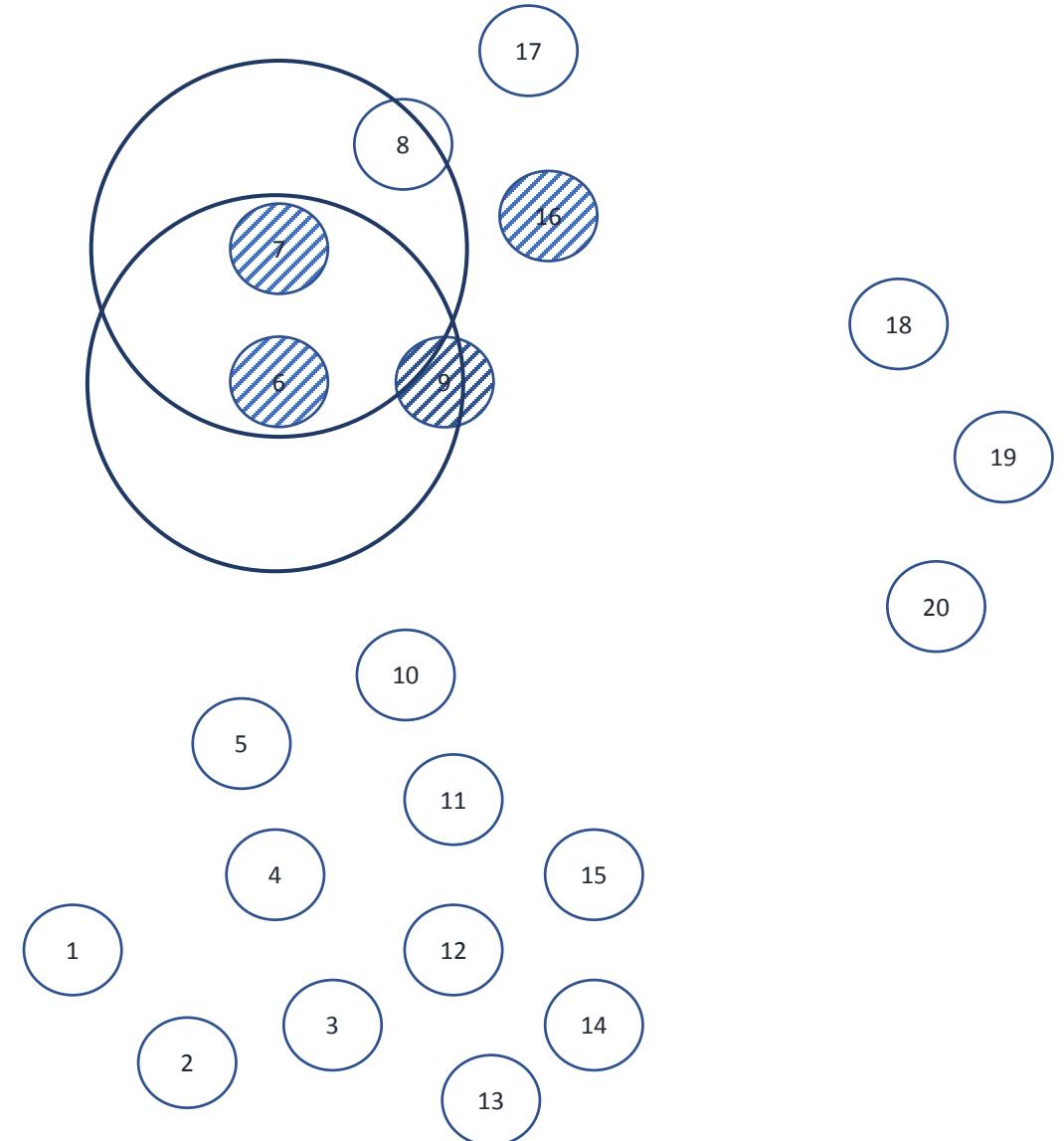
---

```
1: clusters =  $\emptyset$ 
2: for  $p \in P$  do
3:   Q =  $\emptyset$ 
4:   if  $p.\text{status} \neq \text{processed}$  then
5:     Q.add( $p$ )
6:     for  $q \in Q$  do
7:       q.status = 'processed'
8:        $N = \text{GetNeighbors}(q, \epsilon)$ 
9:       Q.addAll( $N$ )
10:    if size(Q)  $\geq \text{minPts}$  then clusters.add(Q)
```

---

$clusters = \{\}$

$Q = \{9, 6, 7, 16\}$



# Euclidean Cluster Extraction

*PCL – EuclideanClusterExtraction(eps, minPts)*

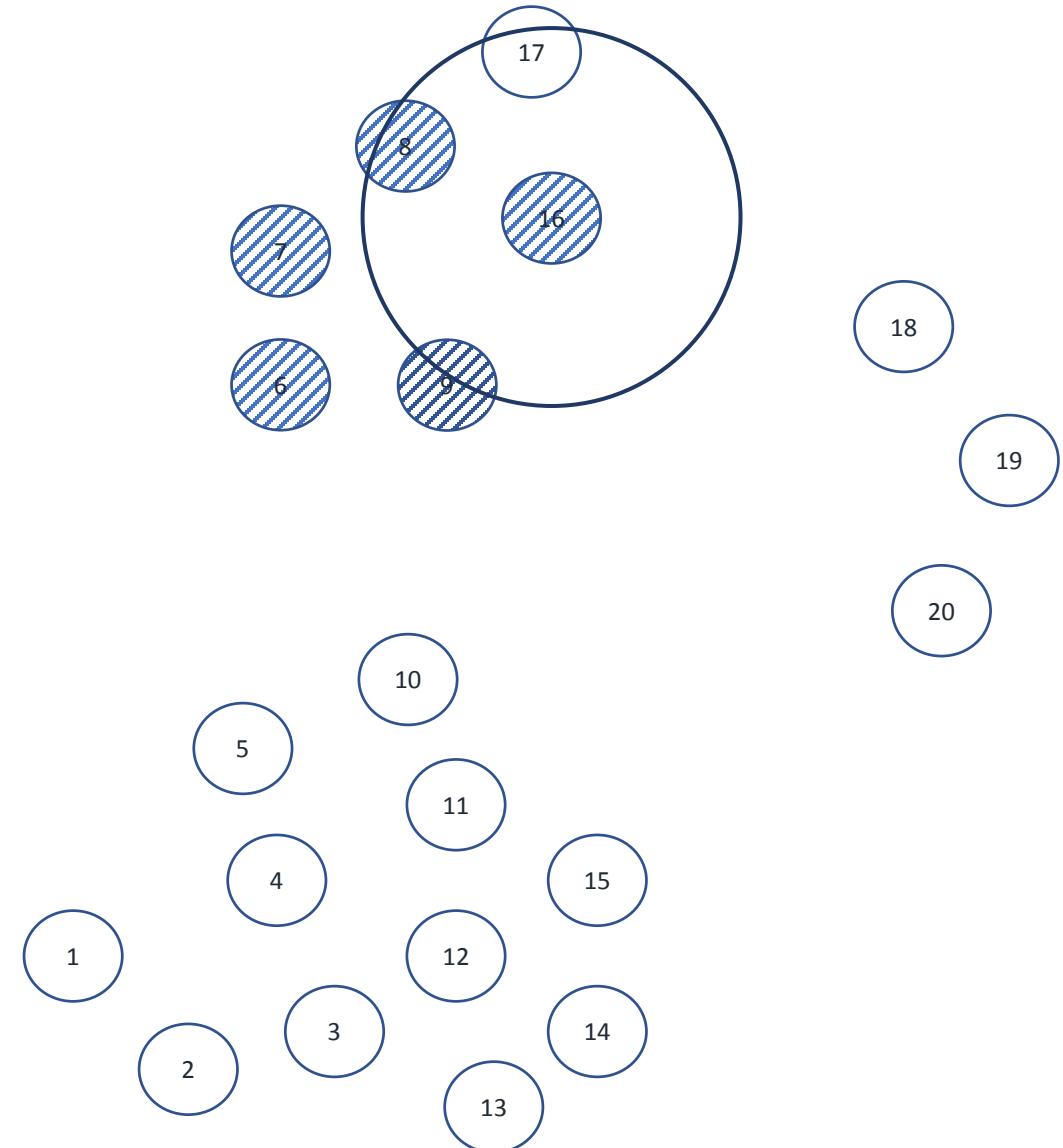
---

```
1: clusters =  $\emptyset$ 
2: for  $p \in P$  do
3:   Q =  $\emptyset$ 
4:   if  $p.\text{status} \neq \text{processed}$  then
5:     Q.add( $p$ )
6:     for  $q \in Q$  do
7:       q.status = 'processed'
8:        $N = \text{GetNeighbors}(q, \epsilon)$ 
9:       Q.addAll( $N$ )
10:    if size(Q)  $\geq \text{minPts}$  then clusters.add(Q)
```

---

$clusters = \{\}$

$Q = \{9, 6, 7, 16, 8\}$



# Euclidean Cluster Extraction

*PCL – EuclideanClusterExtraction(eps, minPts)*

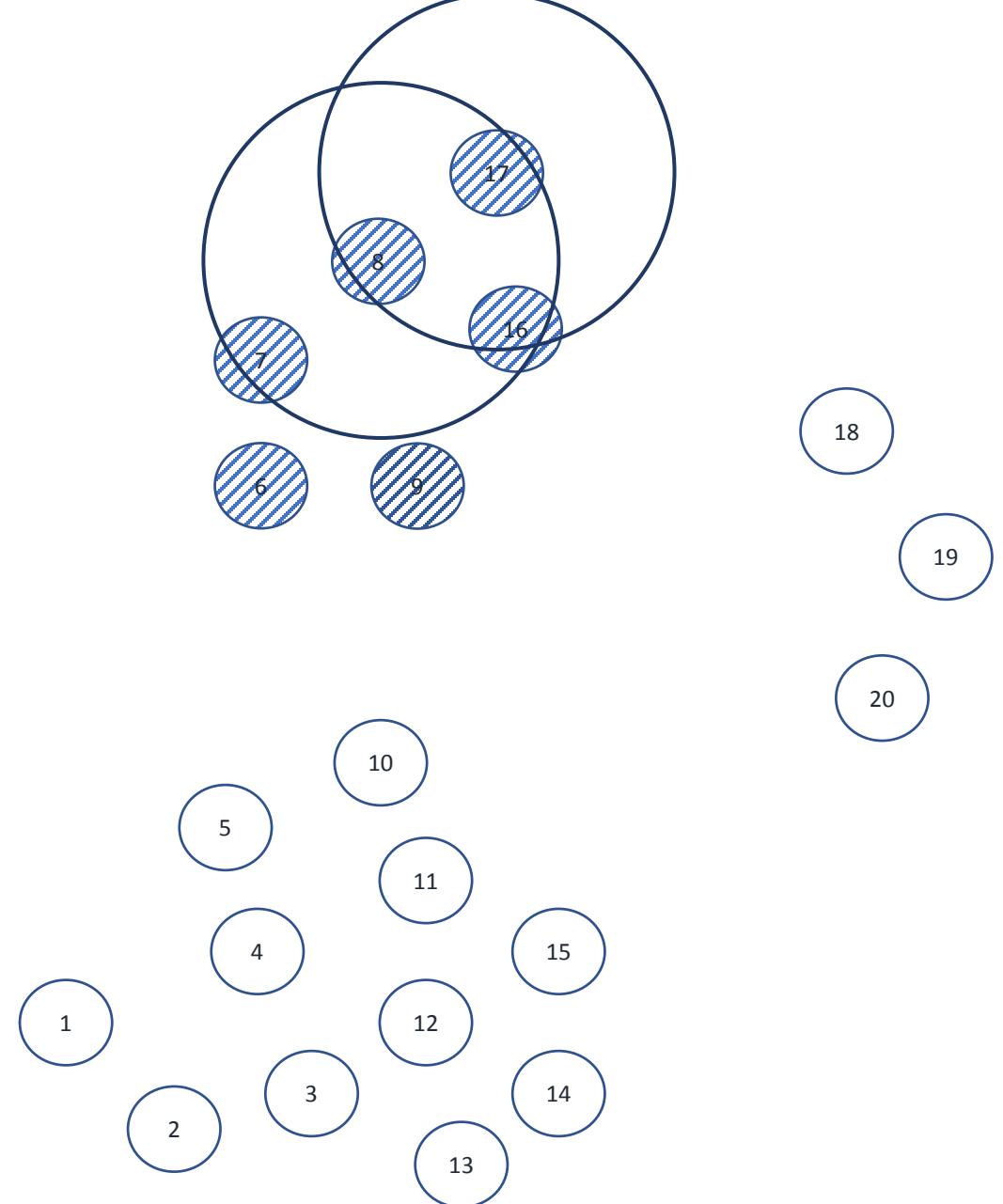
---

```
1: clusters =  $\emptyset$ 
2: for  $p \in P$  do
3:   Q =  $\emptyset$ 
4:   if  $p.\text{status} \neq \text{processed}$  then
5:     Q.add( $p$ )
6:     for  $q \in Q$  do
7:       q.status = 'processed'
8:        $N = \text{GetNeighbors}(q, \epsilon)$ 
9:       Q.addAll( $N$ )
10:    if size(Q)  $\geq minPts$  then clusters.add(Q)
```

---

$clusters = \{\}$

$Q = \{9, 6, 7, 16, 8, 17\}$



# Euclidean Cluster Extraction

*PCL – EuclideanClusterExtraction(eps, minPts)*

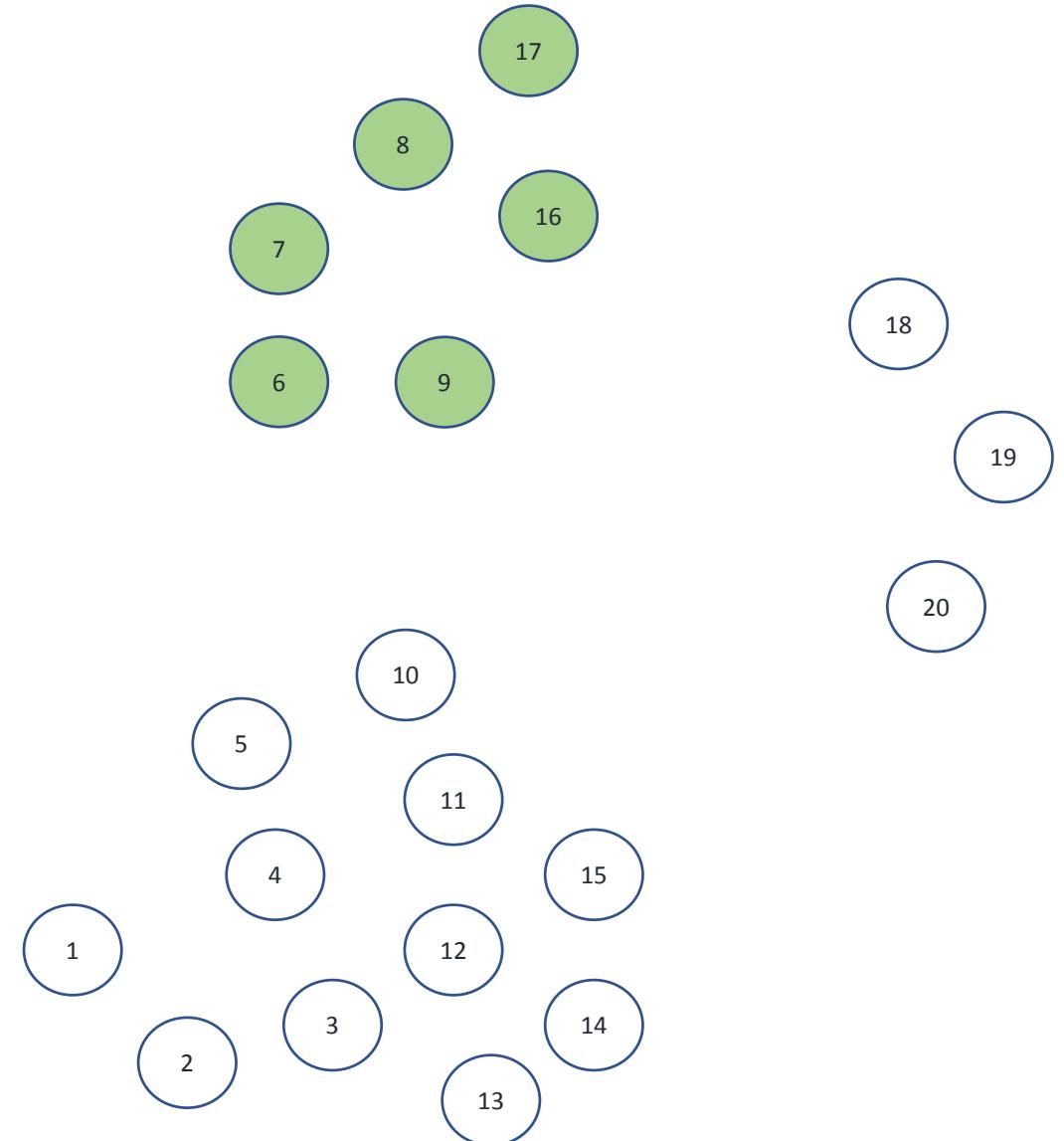
---

```
1: clusters =  $\emptyset$ 
2: for  $p \in P$  do
3:    $Q = \emptyset$ 
4:   if  $p.\text{status} \neq \text{processed}$  then
5:      $Q.\text{add}(p)$ 
6:     for  $q \in Q$  do
7:        $q.\text{status} = \text{'processed'}$ 
8:        $N = \text{GetNeighbors}(q, \epsilon)$ 
9:        $Q.\text{addAll}(N)$ 
10:      if  $\text{size}(Q) \geq \text{minPts}$  then clusters.add(Q)
```

---

$clusters = \{C_1(9,6,7,16,8,17)\}$

$Q = \{\}$



# Euclidean Cluster Extraction

*PCL – EuclideanClusterExtraction(eps, minPts)*

---

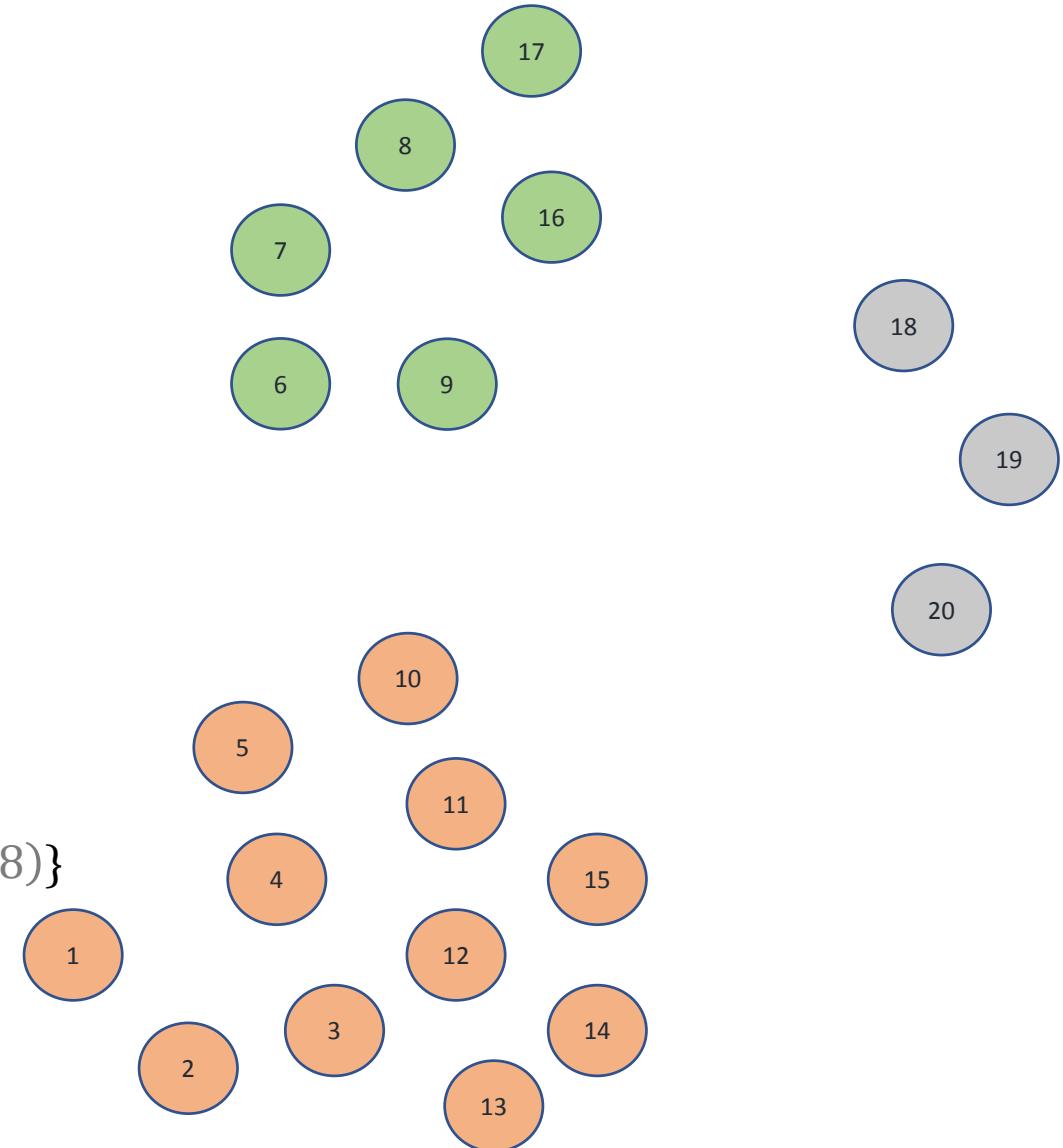
```
1: clusters =  $\emptyset$ 
2: for  $p \in P$  do
3:    $Q = \emptyset$ 
4:   if  $p.\text{status} \neq \text{processed}$  then
5:      $Q.\text{add}(p)$ 
6:     for  $q \in Q$  do
7:        $q.\text{status} = \text{'processed'}$ 
8:        $N = \text{GetNeighbors}(q, \epsilon)$ 
9:        $Q.\text{addAll}(N)$ 
10:      if  $\text{size}(Q) \geq \text{minPts}$  then clusters.add( $Q$ )
```

---

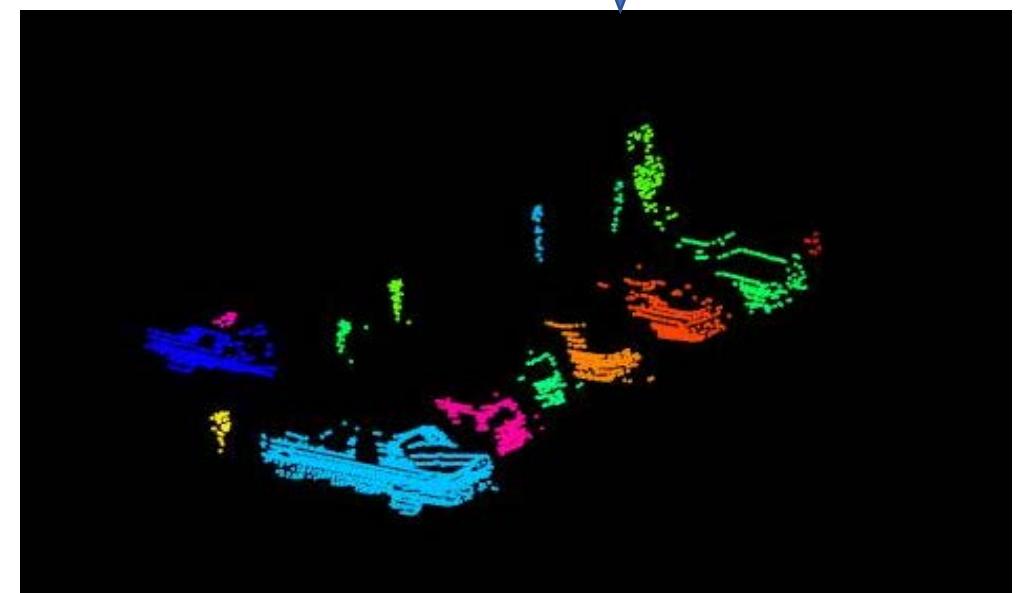
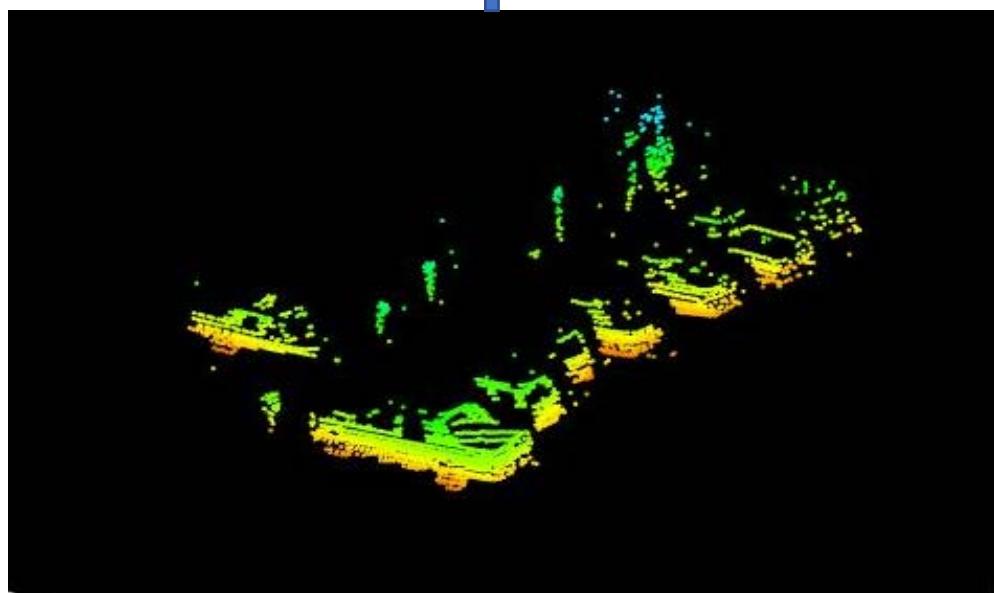
*clusters*

$= \{C_1(9, 6, 7, 16, 8, 17), C_2(4, 5, 3, 11, 12, 2, 13, 10, 15, 14, 1), C_3(19, 20, 18)\}$

*Q = {}*

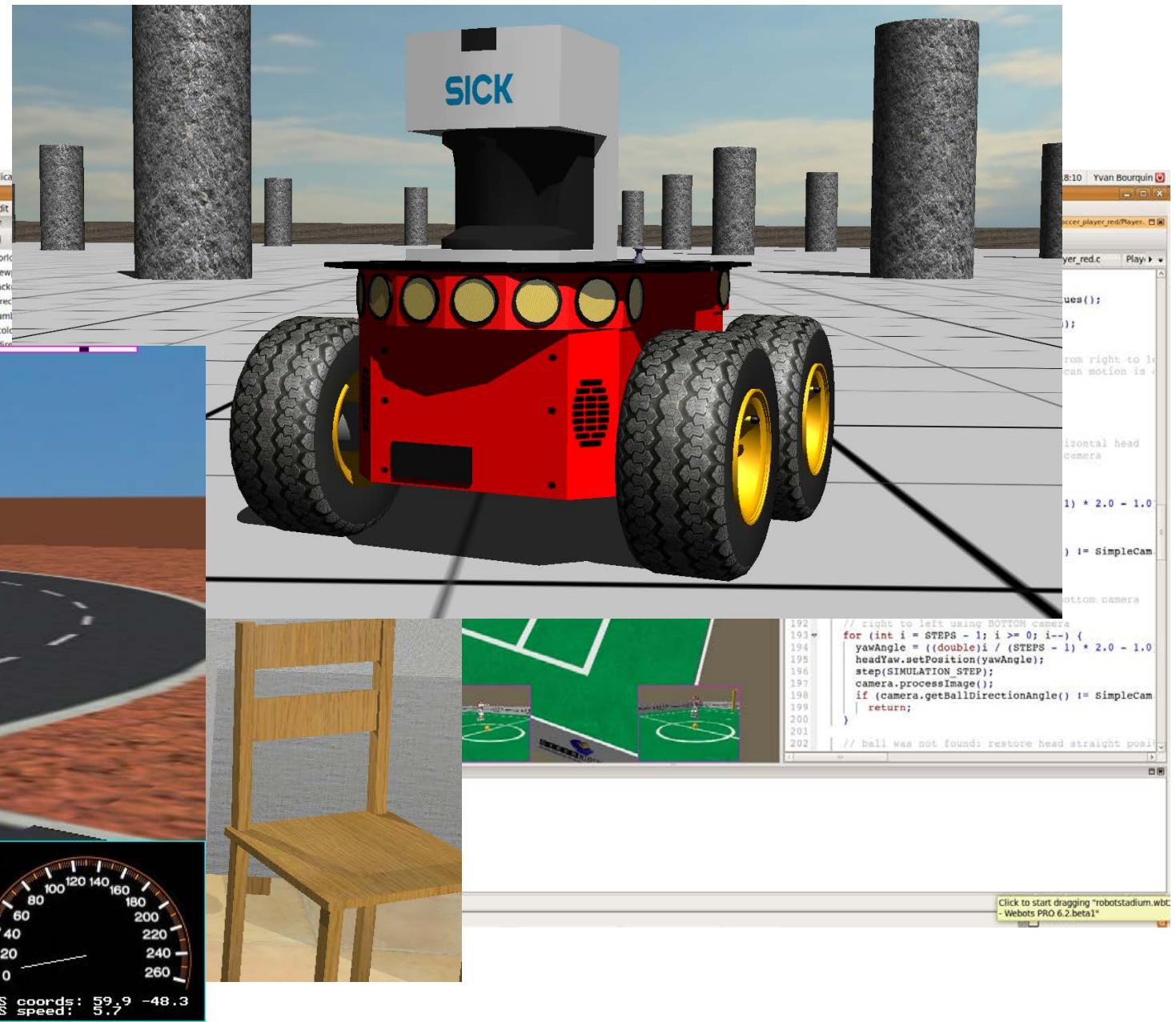


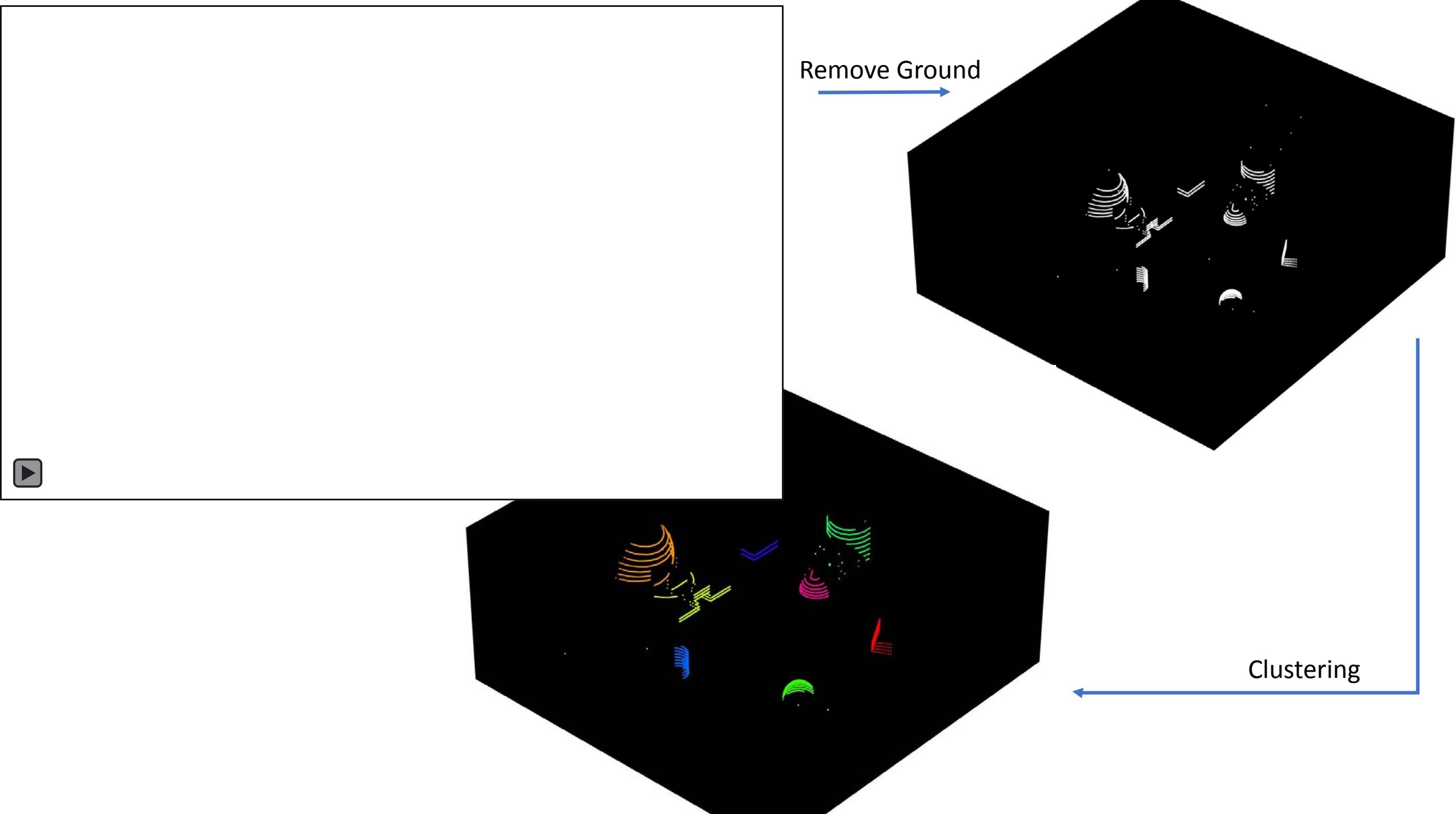
## Euclidean Cluster Extraction



# Webots Simulator

# Webots Simulator





# Use Cases

# Use cases

- Autonomous vehicles
  - Collision avoidance
  - Object detections
- Manufacturing
  - Protect a prohibited area
  - Alert an emergency situation