Laser
Range Finders

Laser Range Finders

- Laser Range Finders are perhaps the most accurate sensors for measuring distances.

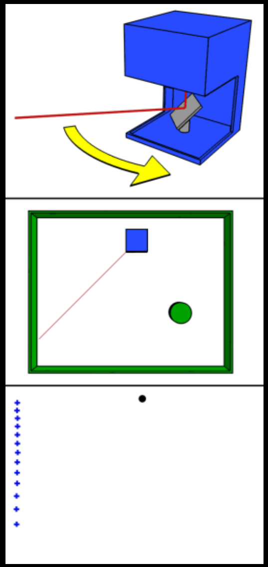
- Similar concept to IR distances sensors in that IR light is emitted and detected.

- These sensors are ***LiDAR*** (Light Detection and Ranging) systems

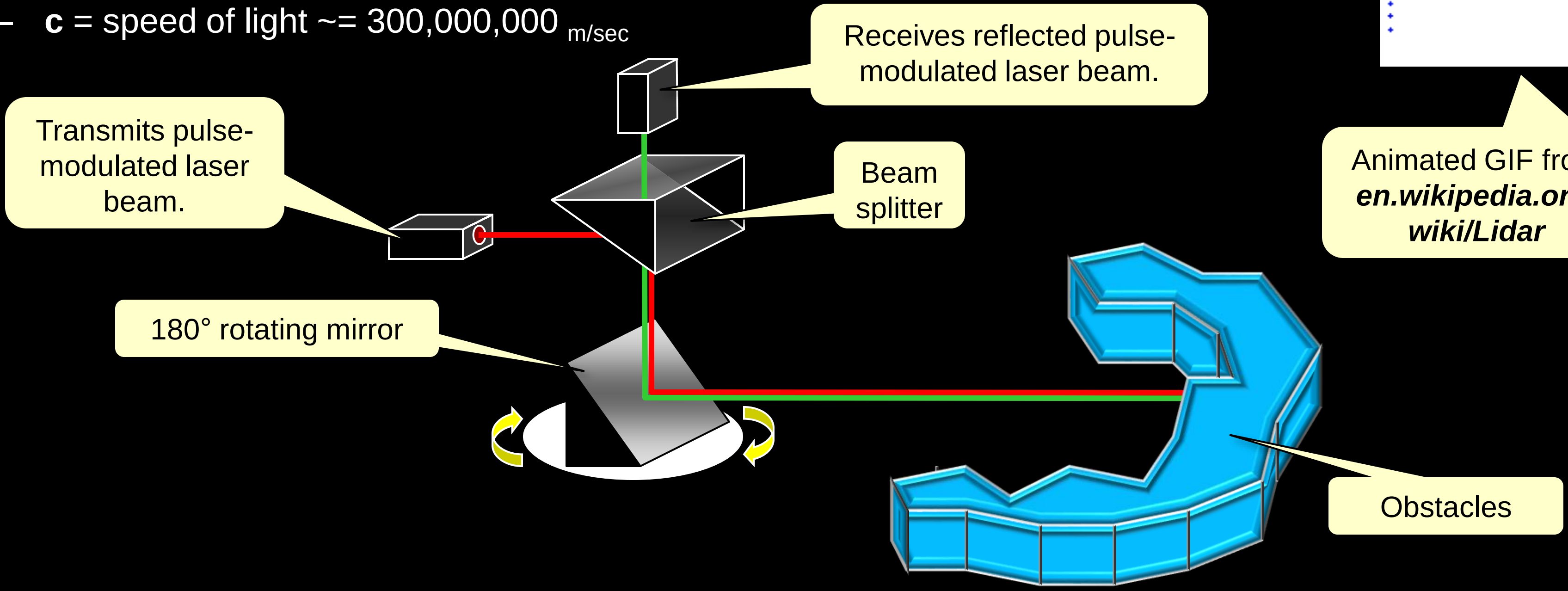
- Lidar systems use one of three techniques:
  - *Pulsed Modulation*
  - *Amplitude Modulation Continuous Wave* (AMCW)
  - *Frequency Modulation Continuous Wave* (FMCW)

Pulsed Modulation Lidar

- Emits a pulsed laser light beam
  - Reflected light is returned to detector
  - Rotating mirrors used to direct outgoing and incoming light to perform up to 240° scan
- Range calculated as **$r = t \times c / 2$**
  - **t** = time taken for light to return from when it was sent
  - **c** = speed of light ~= 300,000,000 $_{m/sec}$

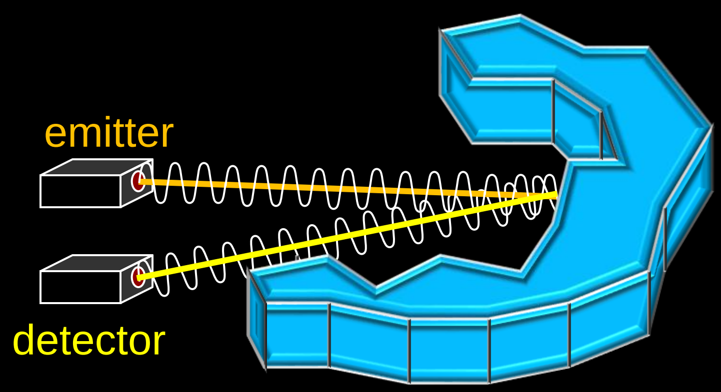


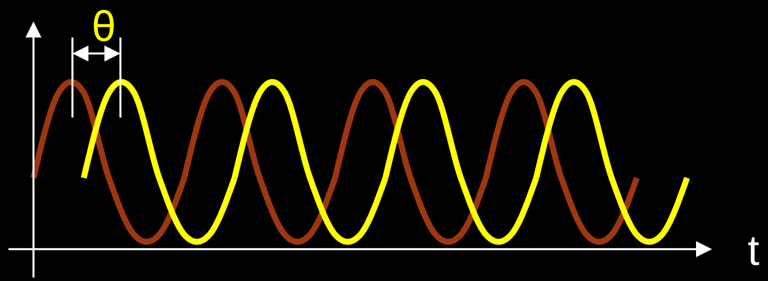
Animated GIF from ***en.wikipedia.org/wiki/Lidar***



Amp. Mod. Cont. Wave Lidar

- AMCW sensors
  - emitter sends out a continuous modulated laser signal (i.e., intensity of beam is modulated using some wave pattern (e.g., sin wave).
  - detected light has same amplitude but is phase-shifted
  - difference in phase shift indicates range

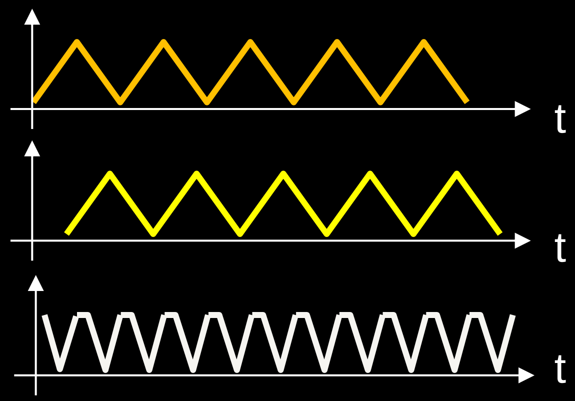


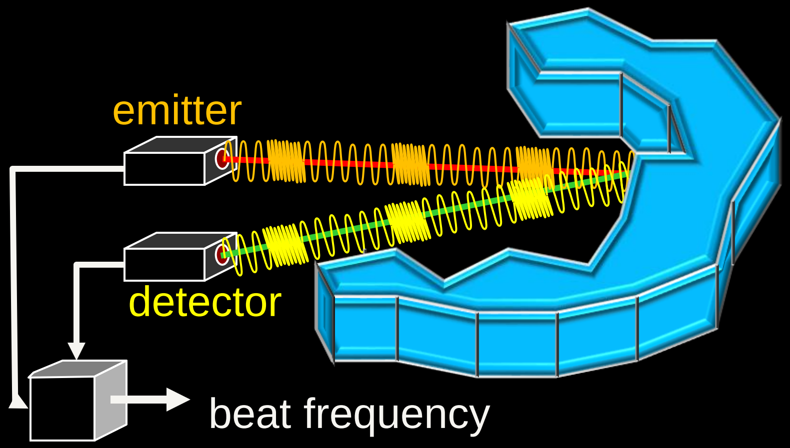


Range calculated as **r = θc/4πf** where
θ = phase shift
f = frequency of modulated signal

Freq. Mod. Cont. Wave Lidar

- FMCW technique is simpler and hence lower cost
- Resolution is limited by modulating frequency
- FMCW sensors similarly emit a continuous laser beam, but modulated now by frequency.
  - emitted signal is mixed with reflected signal.
  - result is difference in frequency





Many Scanners Available … e.g. …

- Sick LMS-291
  - 180° field of view with 0.5° resolution
  - accuracy ±1.5$_{cm}$ in short range (1$_m$ - 8$_m$)
  - and ±4$_{cm}$ in long range (8$_m$ - 20$_m$)

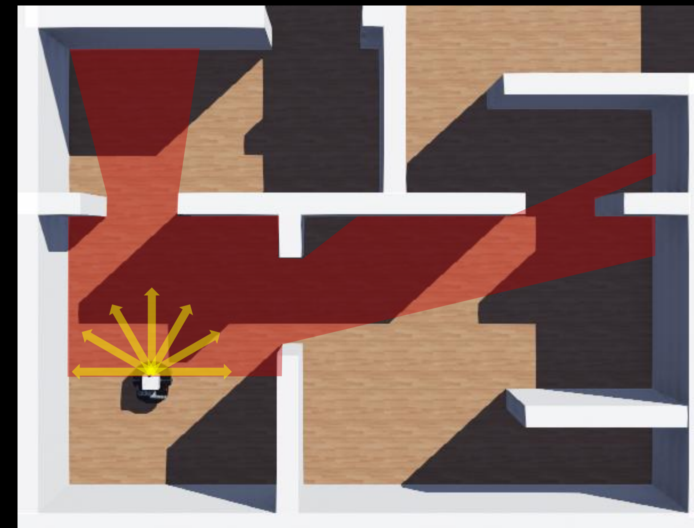
- Hokuyo URG-04LX
  - 240° field of view with 0.36° resolution
  - accuracy ±1$_{cm}$ in range (6$_{cm}$ - 4$_m$)

- Velodyne HDL 32E
  - 360° field of view with 0.33° resolution
  - accuracy ±2 $_{cm}$

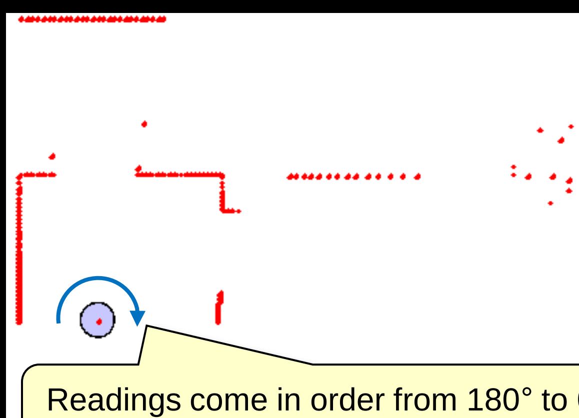
- Velodyne VLP-16
  - 360° field of view with 0.4° resolution
  - accuracy ±3 $_{cm}$

Laser Range Finder 2D Scan

- Consider a robot with a 180° sick LMS-291 lidar sensor in a 2D environment:

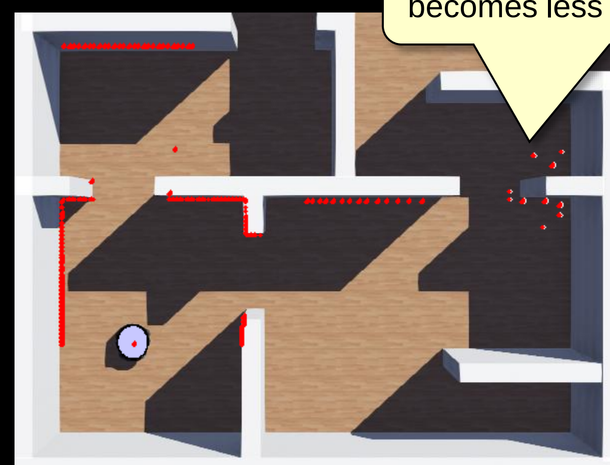


- Scanning a single set of data produces a set of points in 2D:

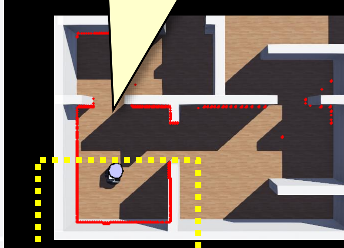


Readings come in order from 180° to 0° (i.e., clockwise scan from left side of sensor).

As readings get further away, data becomes less accurate.



Need just two sets of scans to cover a full 360°.



The Webots Lidar Class

- We use the Lidar class in Webots to represent a Lidar sensor:

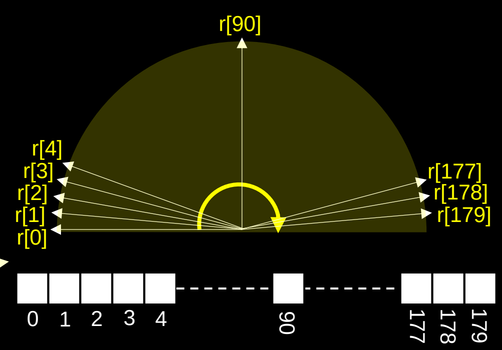
```
import com.cyberbotics.webots.controller.Lidar;

Lidar lidar = new Lidar("Sick LMS 291");
lidar.enable(TimeStep);

int fieldOfView = lidar.getFov()*180/Math.PI          // 180 degrees
int lidarWidth = lidar.getHorizontalResolution();     // 180 degrees
int lidarLayers = lidar.getNumberOfLayers();          // 1 layer
double maxRange = lidar.getMaxRange();                // 80 meters

float r[] = null;

while (robot.step(TimeStep) != -1) {
   r = lidar.getRangeImage();
   ...
}
```

Can be one of these:
- "Ibeo Lux"
- "Hokuyo URG-04LX"
- "Hokuyo URG-04LX-UG01"
- "Hokuyo UTM-30LX"
- "LDS-01"
- "Sick LMS 291"
- "Sick LD-MRS"
- "Velodyne VLP-16"
- "Velodyne HDL-32E"
- "Velodyne HDL-32E"

Various methods are available to get sensor specifications.

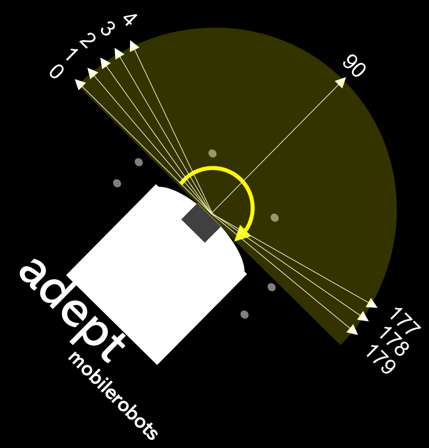
This function reads the sensor and returns an array of the distances for each of the angles. In our case, 1 reading for each degree angle … so 180 readings that cover the 180° range.

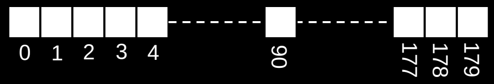
Each range reading is stored in an array at a specific index corresponding to the angle it was obtained at (i.e., from 0 to 179)



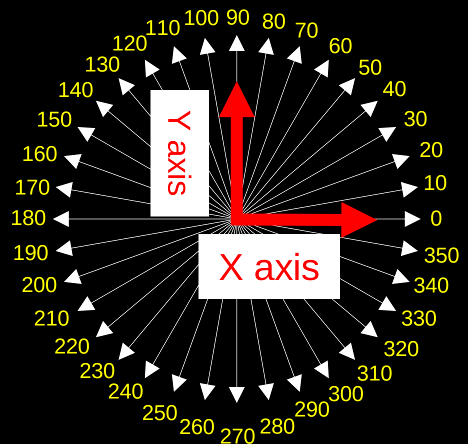
Converting Coordinate Systems

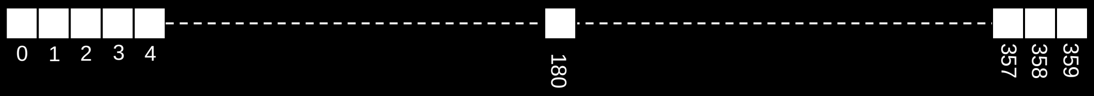
- As mentioned, robot takes range readings which are stored in an array with indices from 0 - 179 representing each of the degrees from 0° to 179° where 90° is directly in front of the robot.





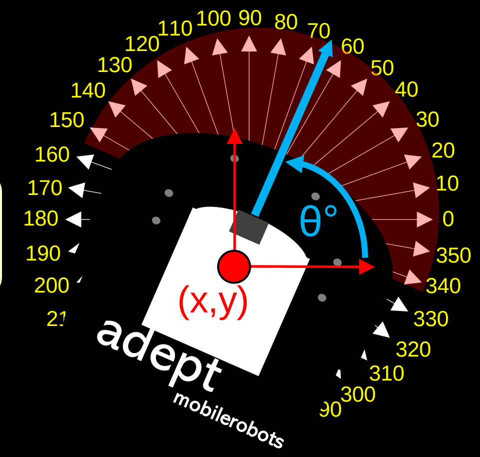
- But we need to apply the readings to the world coordinate system by storing them in the world coordinate array that has indices from 0 - 359 representing each of the degrees from 0° to 359° where 0° is the horizontal in the world coordinate system.





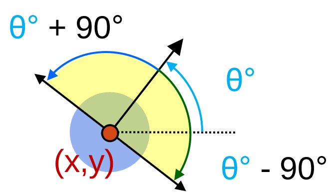
Converting to World Coordinates

- Must transfer current readings from robot array to world array by “rotating the readings”.
  - Just need to flip the readings around and place them at different indices in the world array.



This is the array of lidar readings in the range from 0 to 179 degrees according to the robot’s coordinate system.

```
θ = robot’s angle

FOR angle a from 0 to 179 DO {
    rangeData[θ + 90° - a] = newRanges[a] * 100;
}
```



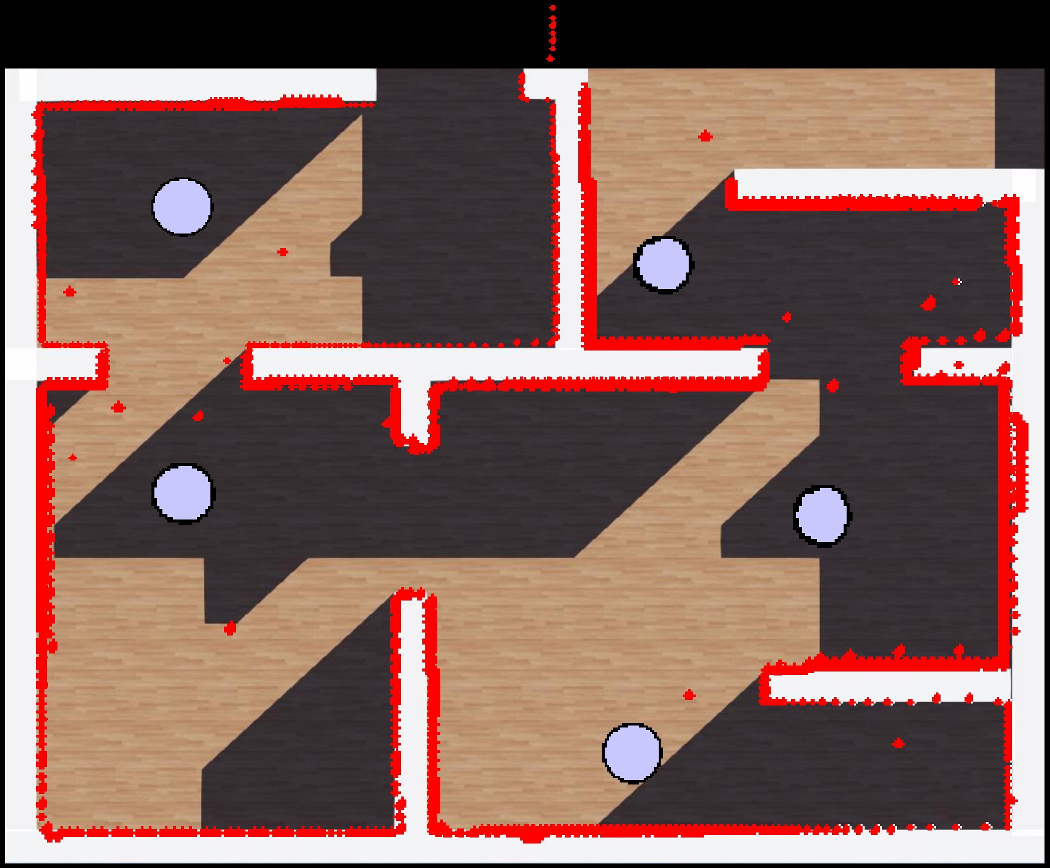
Convert to cm from meters.

This is the array of lidar readings in the range from 0 to 359 degrees according to the world’s coordinate system.

You will need to add code to ensure that this is always in the range from 0 to 359 (i.e., not negative nor above 359).

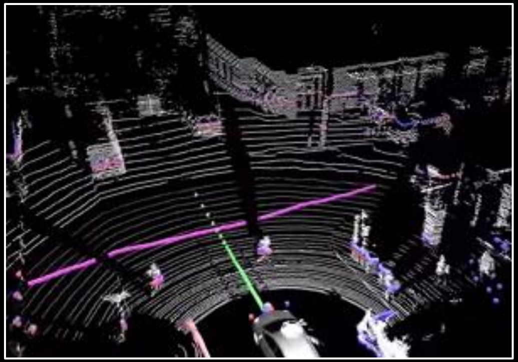
More Accurate Mapping

- Can make accurate maps by moving robot to just a few positions and doing a 360° scan:

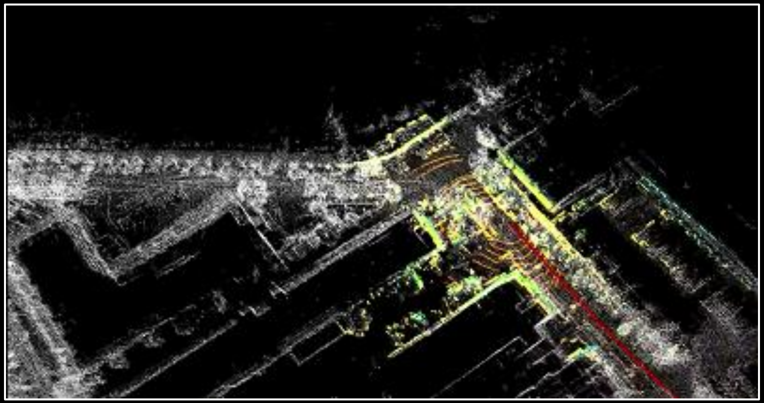


3D Scene Representation

- Taking scans at multiple vertical layers produces 3D scene:
- https://www.youtube.com/watch?v=nXlqv_k4P8Q



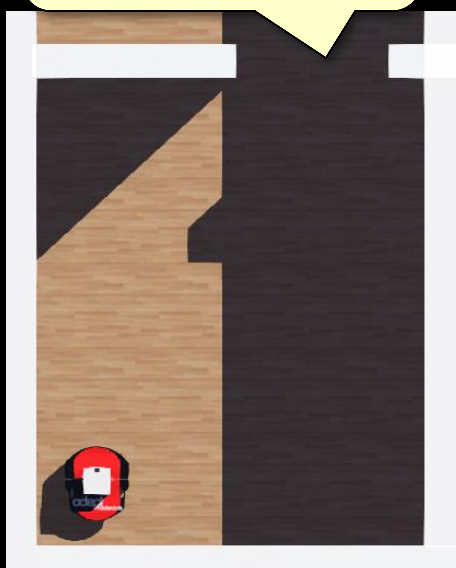
- https://www.youtube.com/watch?v=KmuICcnbQ1U&t=25s



Doorway Identification

- Since there is a lot of data available from a single scan, we can identify features in the environment, such as doorways:

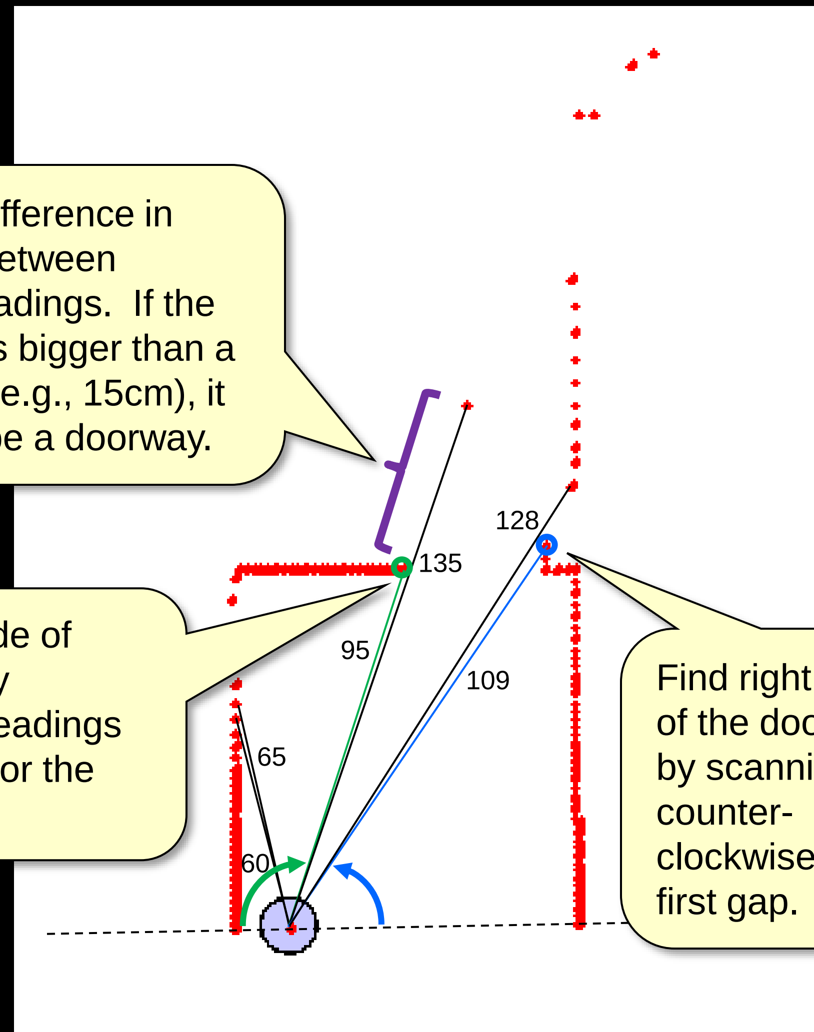
How can we identify this doorway from a single scan ?



Compare difference in distances between adjacent readings. If the difference is bigger than a **threshold** (e.g., 15cm), it may likely be a doorway.

Find left side of doorway by scanning readings clockwise for the first gap.

Find right side of the doorway by scanning counter-clockwise for first gap.



Doorway Identification

- Just go through the rangeReadings array from the left side and then from the right side, looking for the first “big” gap:

82	84	85	87	88	92	91	90	92	94	95	135	215	219	226	....	85	87	128	109	108	106	108	102	100	98	95	94	92
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14		167	168	169	170	171	172	173	174	175	176	177	178	179

First “big” gap from left side is from index 10 to 11. So, the **left doorway** point is at index **10**.

First “big” gap from right side is from index 170 to 169. So, the **right doorway** point is at index **170**.

- Then calculate the coordinates represented by the doorway point. Here is how to calculate the left one:

Angle that robot is currently facing

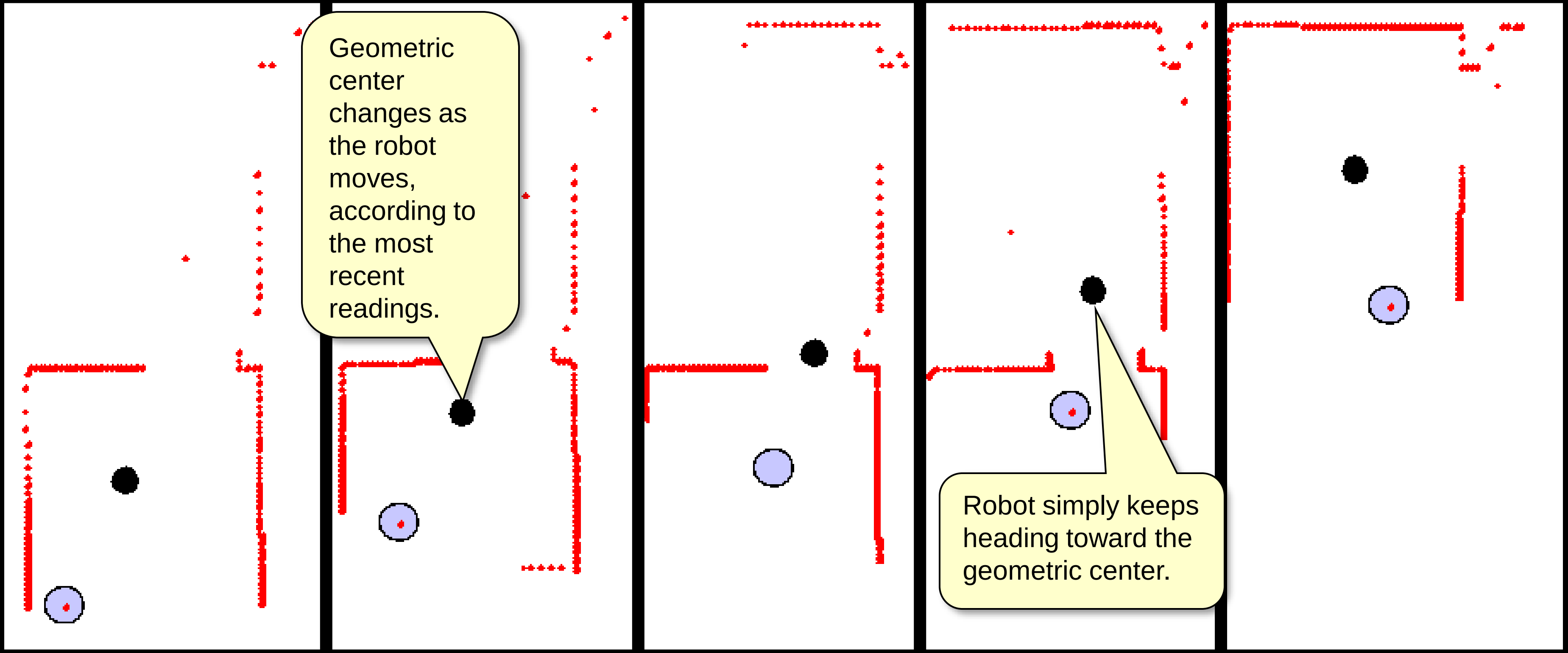
Found from array as shown above

```
leftDoorwayAngle  = θ + 90° - leftDoorwayIndex
leftDoorwayPointX = rangeReadings[leftDoorwayIndex] * cos(leftDoorwayAngle)
leftDoorwayPointY = rangeReadings[leftDoorwayIndex] * sin(leftDoorwayAngle)
```

You will also need to convert from meters to cm here.

Open-Area-Directed Navigation

- We can even navigate through a doorway by travelling towards the open areas.
  - In an empty room, we can find the geometric center (a.k.a. centroid) of the room by averaging all the coordinates:



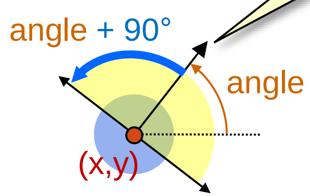
Computing Geometric Center

- Compute geometric center by <u>averaging</u> all computed lidar points based on the robot's current location and orientation:

```
(x, y) = get robot's position
angle = get robot's angle

// This will be the geometric center point
centerX = 0
centery = 0

FOR lidarAngle FROM 0 TO 179 DO {
  d = lidar reading at lidarAngle * 100
  px = d * cos(angle + 90 - lidarAngle)
  py = d * sin(angle + 90 - lidarAngle)
  add px to centerX and py to centerY
}
Divide both centerX and centerY by 180 to get the average

Add x to centerX and y to centerY to translate it to the robot's location
```

Convert to cm from meters.

Robot is facing this way.



This will translate the computed point to be relative to the robot's (and lidar sensor's) location.

Start the Lab ...