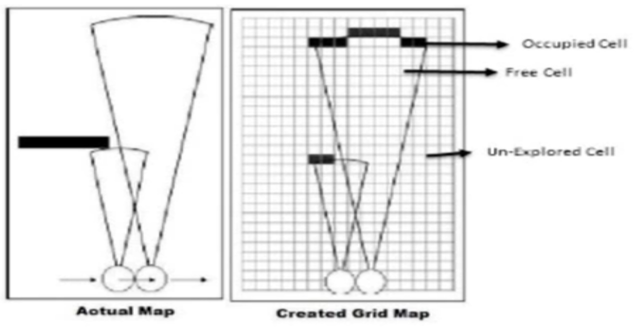
2D - Lidar SLAM algorithms

SLAM - Simultaneous Localization and Mapping
Mapping - Preparing the map of the surrounding environment
Localization - Determine the position and orientation of the robot on the map

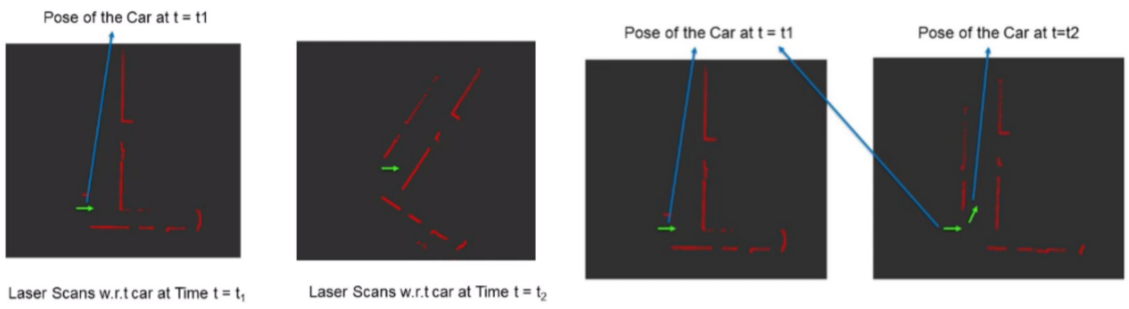
Hector SLAM, **GMapping**, and **Cartographer** are the 3 most used LIDAR-based SLAM algorithms

Hector SLAM:

- This algorithm uses a grid-based approach to mapping and employs a process called "Scan Matching" to estimate the robot's pose.
- The core of Hector SLAM is the occupancy grid map. This grid map divides the environment into a grid of cells. Each cell can have different states, such as "occupied," "free," or "unknown." These states represent the mapping of obstacles and open areas in the environment.
- As the robot moves and obtains new sensor scans, Hector SLAM updates the grid map based on the further sensor information. It marks cells as "occupied" if detected as obstacles in the sensor data.



-
- This algorithm uses a technique called "scan matching" to estimate the robot's pose (position and orientation).
- It compares the current scan of the environment with the previous scan to determine the change in the orientation and position of the robot.

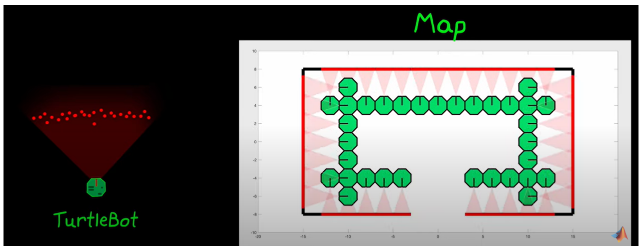

Pose of the Car at t = t1
Laser Scans w.r.t car at Time t = t1
Laser Scans w.r.t car at Time t = t2
Pose of the Car at t = t1
Pose of the Car at t=t2

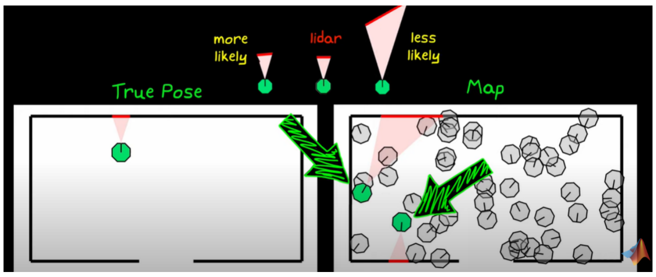
- Hector SLAM doesn’t need any odometry data for localization.
- Hector SLAM is designed for efficiency and real-time performance. It often offers faster mapping and localization results, making it suitable for applications that require rapid responses.

GMapping:

- GMapping stands for Grid-based mapping, and it uses the occupancy grid similar to Hector SLAM to create the map of the environment.
- GMapping employs a particle filter (also known as a Monte Carlo Localization algorithm) to estimate the robot's pose (position and orientation)
- GMapping detects loop closures—instances when the robot revisits a previously mapped area. Loop closure detection helps correct accumulated errors in the map and pose estimation. It's usually achieved by comparing current sensor scans with previously stored ones and identifying similar patterns.
- GMapping can be computationally demanding, especially when maintaining multiple particle filter hypotheses. It may require more computational resources and is sometimes slower, depending on the configuration.

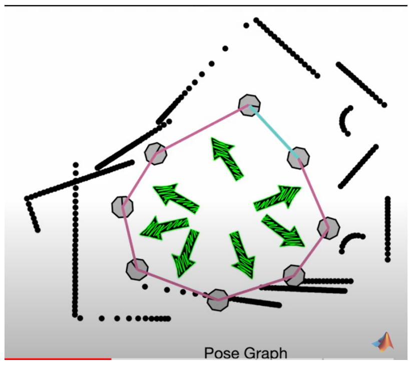
Monte Carlo Localization Algorithm:

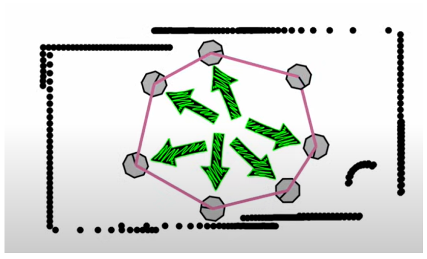
- Initially, the robot has the map of the environment but it doesn't know where exactly it is present on that map.
- This algorithm helps to find the best estimate of the position of the robot.
- Initially, a set of particles is randomly generated across the map, covering the entire space where the robot might be located. These particles have equal weights.
- Here a particle is a hypothetical position and orientation which might be the pose of the robot.
- Now the robot starts moving, so we have the LIDAR data and odometry data now the particles are updated according to this data. (prediction step)
- 

- The particles are also moved in the way the robot is moving and then we compare the lidar scans of the robot with the particles, the particles that are a good match to the robot are kept (their weights are increased) and the others are discarded.
- 

- Now we resample the particles with more particles at the places that are more likely and fewer particles at the places that are less likely to be the robot's position.
- We repeat the last three steps till the particles converge.

Cartographer:

- Cartographer is a graph-based algorithm developed by Google
- Cartographer has two phases, the Local optimization phase and the global optimization phase
- In the local optimization phase, the cartographer uses scan-matching to update the robot's pose and update the submap currently being built.
- However, there will still be errors, and if the environment is large the errors accumulate and have a significant effect.
- Then comes the global phase which uses a pose graph optimization algorithm to improve the accuracy of the map and robot's pose.
- It creates a graph with nodes as poses of the robot as it moves and connects them with edges that represent the constraints between them.
- These constraints can be derived from various sources, such as odometry, visual odometry, loop closures, or other sensor measurements.
- 

- And if it detects a loop closure in the graph i.e. the robot revisits a previously visited location, loop closure constraints are added to align the poses from the current and earlier visits.
- 
- This improves the map also the previous poses of the robot.
- The robot continues moving and adds more poses to the graph and optimizes the map and poses of the robot.
- This algorithm also uses an occupancy grid to represent the map.
- Cartographer is renowned for its ability to create highly accurate and detailed maps of complex indoor environments.

References:

- **Hector SLAM:** Lecture 3 2: Hector Mapping - Simultaneous Localization and Map…
- **Monte Carlo Localization:** Understanding the Particle Filter | | Autonomous Navigation, Part 2
- **Pose Graph Optimization:** Understanding SLAM Using Pose Graph Optimization | Autonomous Navigation, P…
- **Cartographer:** https://research.google/pubs/pub45466/
- https://github.com/robopeak/rplidar_ros/wiki
- https://www.researchgate.net/publication/340113105_RP_Lidar_Sensor_for_Multi-Robot_Localization_using_Leader_Follower_Algorithm