GUARANTEED SLAM – PRACTICAL CONSIDERATIONS

Eduard Codres, Mohamed Mustafa, Mario Martinez Guerrero, and Alexandru Stancu

School Of Electrical And Electronic Engineering

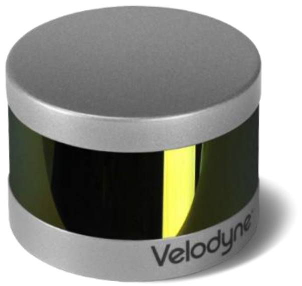
University of Manchester

SWIM 2018

SLAM - *Simultaneous Localisation And Mapping*

- SLAM is used in mobile robots for building reliable maps of unknown environments.
- Building a reliable map of the environment is a critical task firstly because the mobile robot needs to have safe navigation in unknown environments and secondly because in many cases an accurate map is needed for future use when the environment is dangerous or not accessible to humans.
- A mobile robot can build a map of its environment using any of the available exteroceptive sensors such as sonars, digital cameras, structured light sensors, LiDARs etc.

Intel Realsense D435



Velodyne 3D LiDAR

High speed digital camera

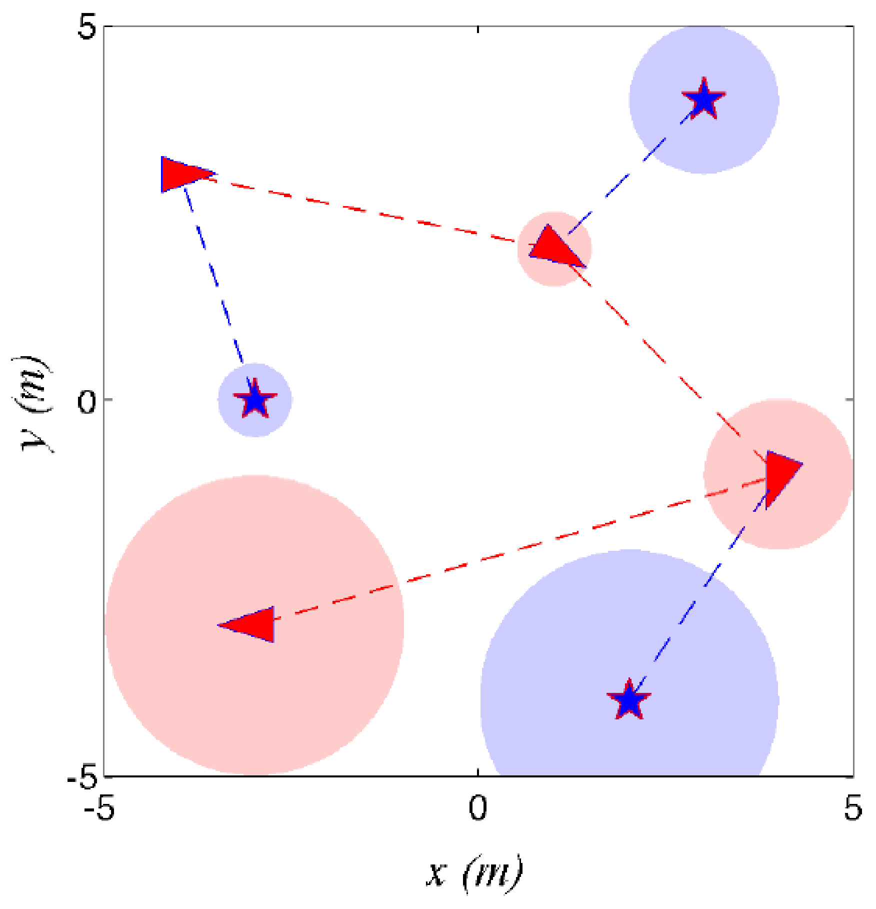
SLAM - *Simultaneous Localisation And Mapping*

- Since the measurements from the exteroceptive sensors are relative to the robot, building the map requires a transformation from the robot frame to the fixed world frame. This transformation is only possible if the robot position is available, which can be found using localisation techniques.

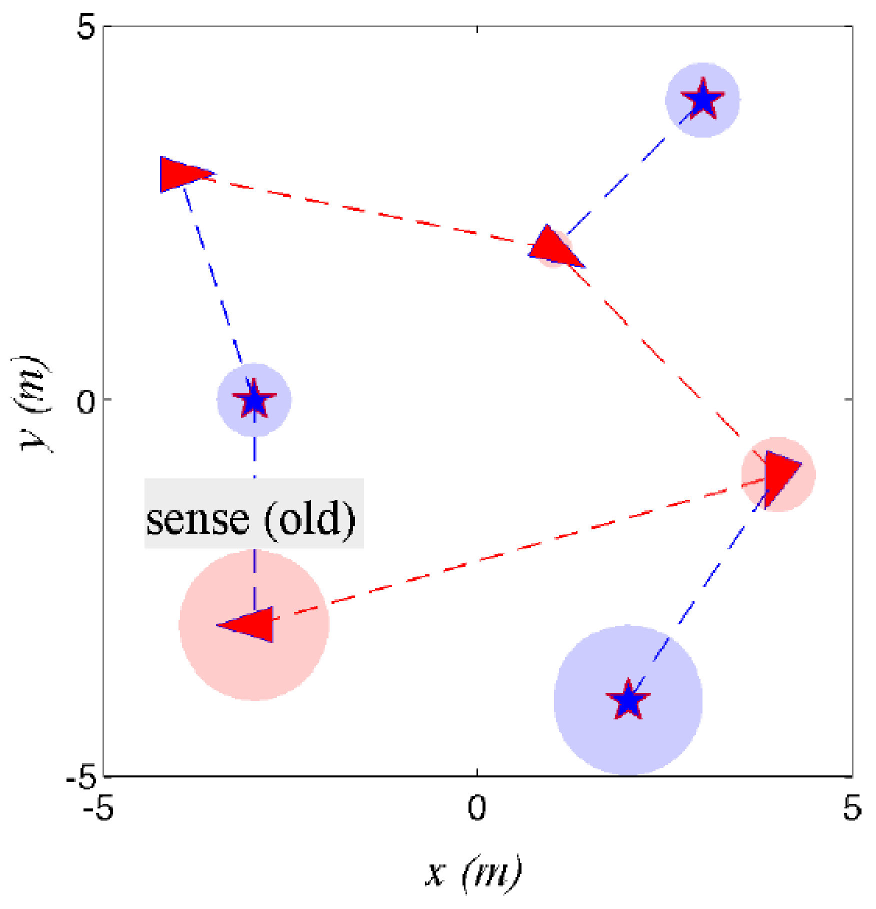
- The easiest localisation can be implemented using proprioceptive sensors such as rotary encoders or inertial measurement units (IMU), but these types of sensors are usually noisy and can lead to uncertainty rapidly accumulating in the robot pose.

- Because mapping problem is always solved in conjunction with the localisation problem and vice-versa, this approach is called *Simultaneous Localisation And Mapping* (SLAM).

SLAM - *Simultaneous Localisation And Mapping*



(a)



(b)

Types of SLAM

- Various SLAM approaches exist, but most of them can be summarised, depending on the way they deal with uncertainty, into two categories:
  - probabilistic SLAM;
  - interval SLAM.

- Most notable probabilistic approaches are based on Kalman filter, extended Kalman filter or a combination of particle filter and extended Kalman filter (FastSLAM).

- Interval SLAM tries to solve the drawbacks of the probabilistic methods (they need a Gaussian noise distribution and model linearisation) in a guaranteed way.

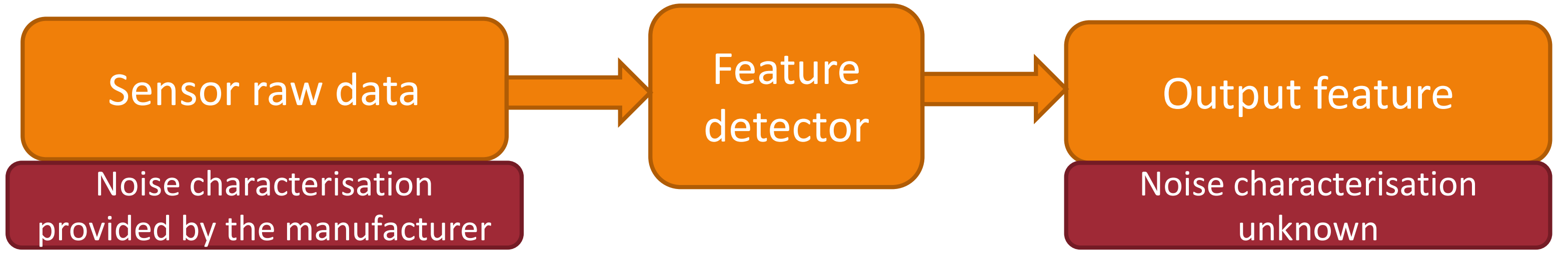
- One of the biggest drawbacks when using interval methods is that, in many cases, they can be pessimistic.

SLAM with feature detection

- One issue with the existing guaranteed SLAM approaches is that they rely on existing methods or algorithms to detect features in the raw data provided by the sensors.

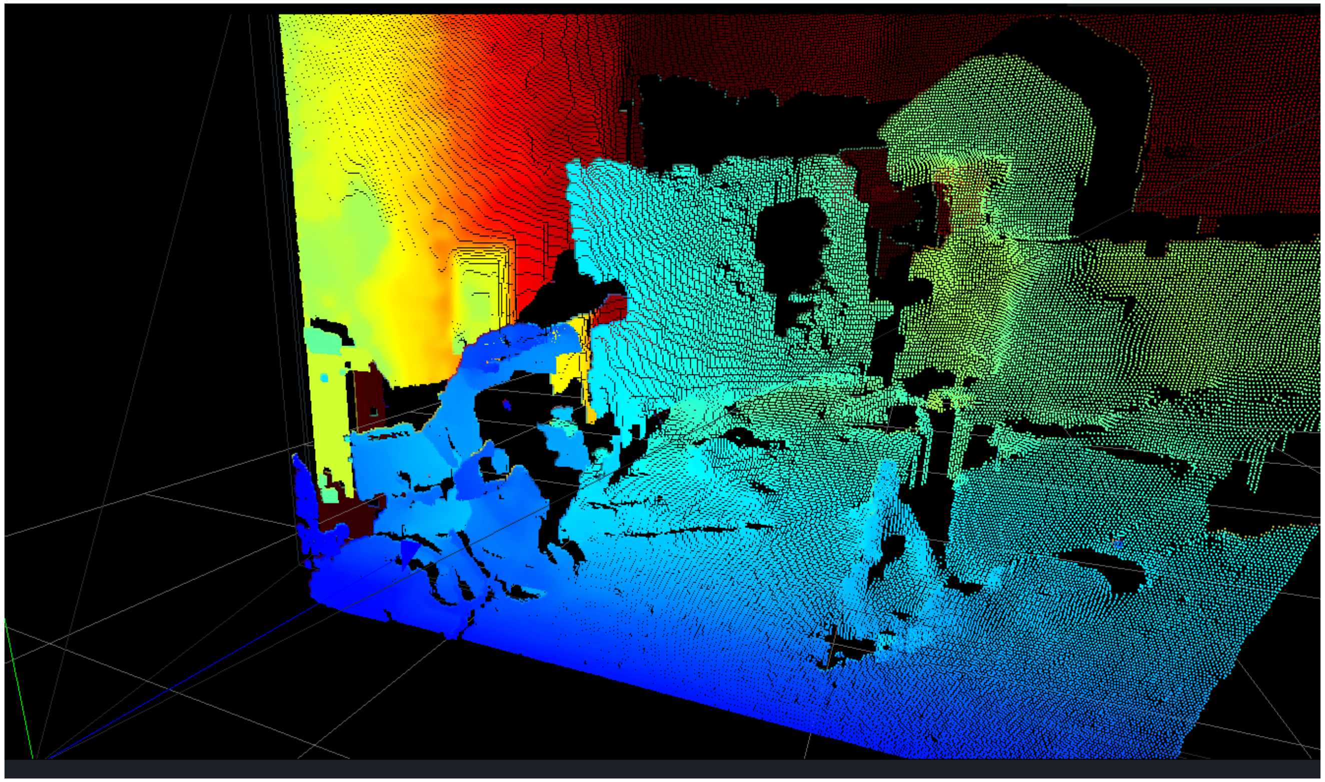
- The image detected features can be: point features, corners, edges etc.

- Geometric features can also be extracted from point cloud data: line features, planar features, cylinders etc.


Sensor raw data
Noise characterisation provided by the manufacturer
Feature detector
Output feature
Noise characterisation unknown

Problem formulation

- Because interval methods can be pessimistic it is better to use all available sensor data.


Example of a dense point cloud generated by a structured light sensor.

Problem formulation

- To address the drawbacks presented before, a SLAM method which takes into account all sensor measurements and generates a CSP (Constraint Satisfaction Problem) is proposed.

- A non-holonomic robot with a 2D Lidar sensor is used as an example.

- In this approach two main problems have to be addressed:
  - The data association problem has to be solved for each sensor measurement in order to be able to contact the CSP;
  - Computation time has to be small enough such that the CSP is contracted in real time. The LiDAR sensor (Hokuyo URG-04lx) generates up to 680 measurements with a frequency of 10 Hz.

Interval SLAM – general aspects

- Generally, the SLAM problem is described by the robot motion model coupled with the sensor model.

- The robot motion model is:

$$\mathbf{s}_k = \boldsymbol{h}(\mathbf{s}_{k-1}, \mathbf{u}_k) + \mathbf{q}_k$$

, where $\mathbf{s}_k$ is the robot pose at step *k*, $\boldsymbol{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)$ is a nonlinear function and $\mathbf{q}_k$ is additive noise associated with the motion.

- The sensor model is given by:

$$\mathbf{z}_{i,k} = \boldsymbol{g}(\mathbf{m}_\mathrm{i}, \mathbf{s}_k) + \mathbf{r}_k$$

, where $\mathbf{z}_{i,k}$ is the measurement of landmark *i* at step *k*, $\boldsymbol{g}(\mathbf{m}_\mathrm{i}, \mathbf{s}_k)$ is a nonlinear function and $\mathbf{r}_k$ is additive sensor noise.

Interval SLAM – general aspects

- The following set of variables is used to solve SLAM as a CSP:

$$\mathbf{x} = \left[\mathbf{s}_0^{\mathrm{T}}, \cdots, \mathbf{s}_{n_f}^{\mathrm{T}}, \mathbf{m}_1^{\mathrm{T}}, \cdots, \mathbf{m}_{n_l}^{\mathrm{T}}\right]$$

, where n_f is the number of time steps and n_l is the number of landmarks.

- The SLAM constraint satisfaction problem is defined by the following set of constraints:

$$\mathbf{s}_k - \boldsymbol{h}(\mathbf{s}_{k-1}, \mathbf{u}_k) \in [\mathbf{q}]$$
$$\mathbf{z}_{i,k} - \boldsymbol{g}(\mathbf{m}_{\mathrm{i}}, \mathbf{s}_k) \in [\mathbf{r}]$$

, where $i \in \{1, \cdots, n_l\}$, $k \in \left\{1, \cdots, n_f\right\}$ and $\mathbf{q}_k$, $\mathbf{r}_k$ are bounded such that $\mathbf{q}_k \in [\mathbf{q}]$, $\mathbf{r}_k \in [\mathbf{r}]$.

- The SLAM CSP can be contracted using a forward-backward contractor.

Interval SLAM – implementation

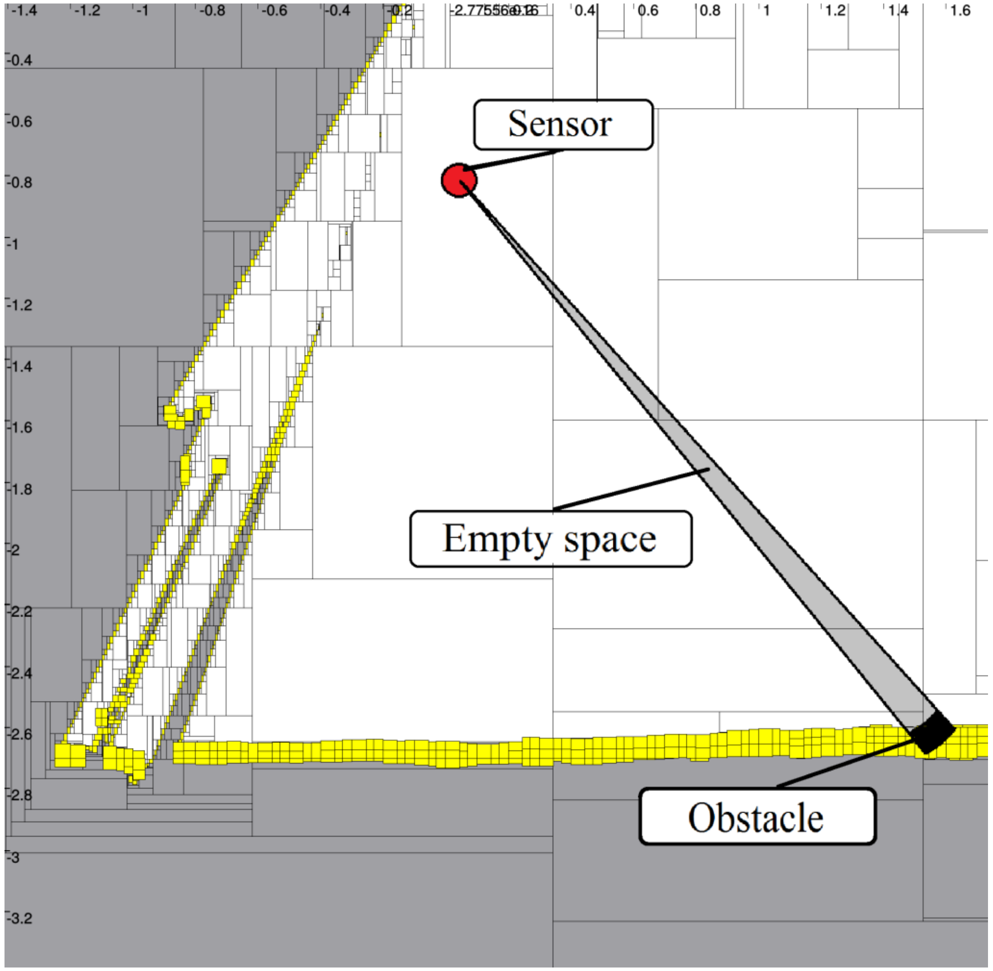
- To solve the data association problem without using a feature detector the map is defined in the following manner:

$$\mathcal{M}:\{\mathcal{P}_{EMPTY},\mathcal{P}_{OBSTACLE}\}$$

- Where:
  - $\mathcal{P}_{EMPTY}$ - defines the area of the map without obstacles;
  - $\mathcal{P}_{OBSTACLE}$ - defines the area of the map containing obstacles.

- Most sensors can be used to generate a map containing the two sets described above.

Interval SLAM – implementation



$\mathcal{P}_{EMPTY}$ is shown in white, $\mathcal{P}_{OBSTACLE}$ is shown in yellow

Interval SLAM – implementation

- The two sets that accumulate all measurements for the empty area and obstacles for each time step are:

$$\mathcal{P}_{EMPTY}^{k} = \bigcup_{i \in \mathcal{Z}_{new}^{k}} \mathcal{P}_{EMPTY}^{i,k}\,; \qquad \mathcal{P}_{OBSTACLE}^{k} = \bigcup_{i \in \mathcal{Z}_{new}^{k}} \mathcal{P}_{OBSTACLE}^{i,k}\,; \qquad \mathcal{P}_{OLD}^{k} = \bigcup_{i \in \mathcal{Z}_{old}^{k}} \mathcal{P}_{OBSTACLE}^{i,k};$$

, where $\mathcal{Z}_{old}^{k}$ and $\mathcal{Z}_{new}^{k}$ are two sets that contain the old and new measurements for each step, respectively. $\mathcal{P}_{EMPTY}^{i,k}$ and $\mathcal{P}_{OBSTACLE}^{i,k}$ are generated using the sensor model.

- The map, containing all measurements for all steps, is then updated by:

$$\mathcal{P}_{EMPTY} = \bigcup_{k=1}^{n_f} \mathcal{P}_{EMPTY}^{k}\,; \qquad \mathcal{P}_{OBSTACLE} = \left(\bigcup_{k=1}^{n_f} \mathcal{P}_{OBSTACLE}^{k} \right) \bigcap \left(\bigcup_{k=1}^{n_f} \mathcal{P}_{OLD}^{k} \right)$$

- A measurement is considered old if it was previously included in the map. This is verified with the following inclusion test:

$$\mathcal{P}_{EMPTY}^{i,k} \subset \mathcal{P}_{EMPTY} \cup \mathcal{P}_{OBSTACLE}$$

Interval SLAM – implementation

- A non-holonomic robot equipped with a 2D LiDAR sensor is used to test the interval SLAM in a practical application.
- The kinematic model is implemented to generate the constraints for the robot motion model:

$$\begin{cases} s_{x,k} = s_{x,k-1} + v_k \cdot \Delta t \cdot \cos(s_{\theta,k-1}) + q_{x,k} \\ s_{y,k} = s_{y,k-1} + v_k \cdot \Delta t \cdot \sin(s_{\theta,k-1}) + q_{y,k} \\ s_{\theta,k} = s_{\theta,k-1} + \omega_k \cdot \Delta t + q_{\theta,k} \end{cases}$$

, where $\boldsymbol{s_k} = [s_{x,k}, s_{y,k}, s_{\theta,k}]^T$ is the robot pose vector, $\boldsymbol{u_k} = [v_k, \omega_k]^T$ is the input vector and $\boldsymbol{q_k} = [q_{x,k}, q_{y,k}, q_{\theta,k}]^T$ represents the motion noise

- The sensor model for the 2D LiDAR is shown below:

$$\begin{cases} z_{\rho,i,k} = \sqrt{(m_{x,i} - s_{x,k})^2 + (m_{y,i} - s_{y,k})^2} + r_{\rho,k} \\ z_{\alpha,i,k} = atan2(m_{y,i} - s_{y,k}, m_{x,i} - s_{x,k}) - s_{\theta,k} + r_{\alpha,k} \end{cases}$$

, where $\boldsymbol{z}_{i,\boldsymbol{k}} = [z_{\rho,i,k}, z_{\alpha,i,k}]^{\boldsymbol{T}}$ is the measurement vector (range and bearing) in robot frame, $\boldsymbol{m}_i = [m_{x,i}, m_{y,i}]^T$ is the landmark being measured and $\boldsymbol{r}_k = [r_{\rho,k}, r_{\alpha,k}]^T$ is the measurement noise.

Interval SLAM – implementation

- For the empty area set the following constraints are defined:

$$\mathcal{P}^{i,k}_{EMPTY}: \begin{cases} \sqrt{\left(x - s_{x,k}\right)^2 + \left(y - s_{y,k}\right)^2} \in \left[0, z_{\rho,i,k} - \overline{r}_{\rho,k}\right] \\ atan2\left(y - s_{y,k}, x - s_{x,k}\right) - s_{\theta,k} \in z_{\alpha,i,k} + \left[r_{\alpha,k}\right] \end{cases}$$

- For the obstacle area set the following constraints are defined:

$$\mathcal{P}^{i,k}_{OBSTACLE}: \begin{cases} \sqrt{\left(x - s_{x,k}\right)^2 + \left(y - s_{y,k}\right)^2} \in z_{\rho,i,k} + \left[r_{\rho,k}\right] \\ atan2\left(y - s_{y,k}, x - s_{x,k}\right) - s_{\theta,k} \in z_{\alpha,i,k} + \left[r_{\alpha,k}\right] \end{cases}$$

, where $i \in \{1, \cdots, n_m\}$, n_m is the number of measurements at step *k*.

- The measurement noise is bounded by intervals such that:

$$r_{\rho,k} \in \left[\underline{r}_{\rho,k}, \overline{r}_{\rho,k}\right]$$
$$r_{\alpha,k} \in \left[\underline{r}_{\alpha,k}, \overline{r}_{\alpha,k}\right]$$

Interval SLAM – computing optimisations

- Recall that the following inclusion test is required for each measurement:

$$\mathcal{P}_{EMPTY}^{i,k} \subset \mathcal{P}_{EMPTY} \cup \mathcal{P}_{OBSTACLE}$$

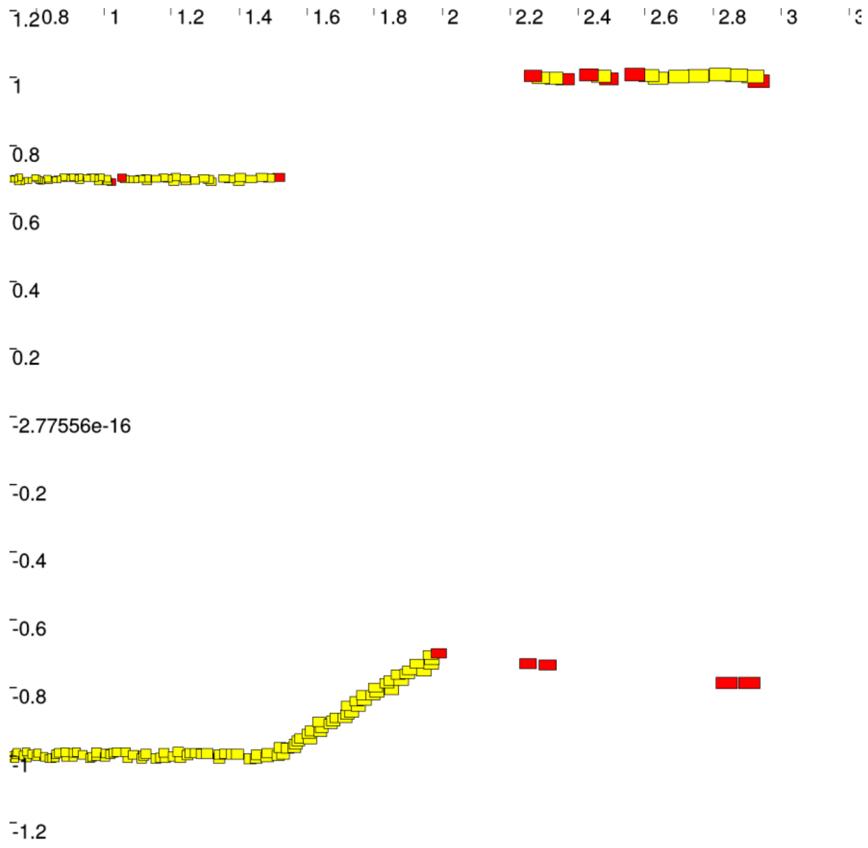
- In practice, this inclusion test can be computationally expensive. Because the sensor generates a compact and ordered set of measurements this can be used to reduce the number of inclusion tests.

- All measurements that are potentially near unmeasured areas are marked on the map. The set of these measurements is denoted by $\mathcal{P}_{UNKNOWN}$.

- With this new information a measurement is considered old if it satisfies the following test, which is easier to implement:

$$\mathcal{P}_{OBSTACLE}^{i,k} \cap \mathcal{P}_{UNKNOWN} = \emptyset$$

Interval SLAM – computing optimisations



Measurements in red are near unmeasured space and included in $\mathcal{P}_{UNKNOWN}$

Interval SLAM – computing optimisations

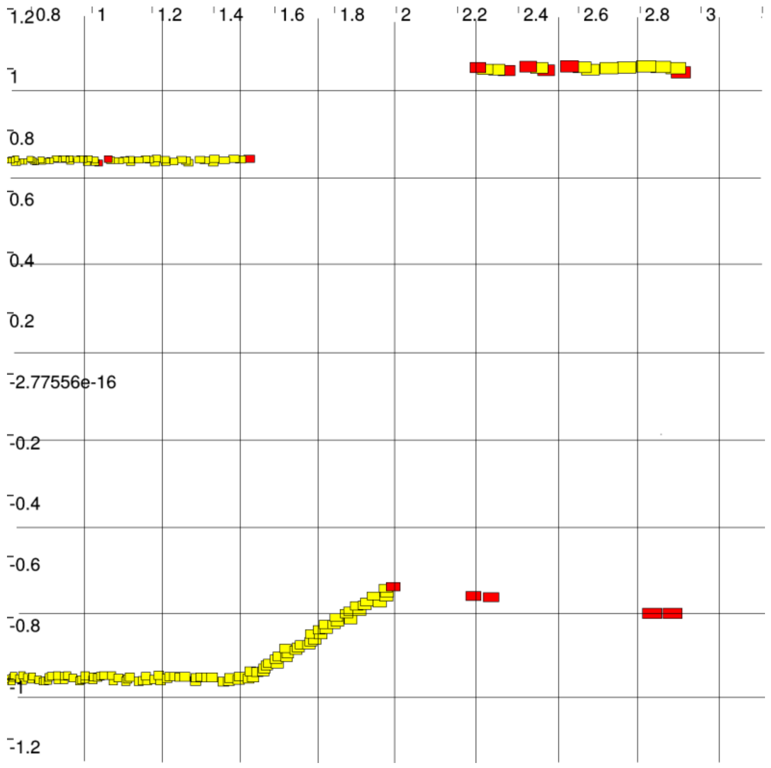
- As the map $\mathcal{M}$ increases, the number of boxes which define the two sets of the map becomes bigger. This, in turn, increases the number of computations required for $\mathcal{P}_{EMPTY}$ and $\mathcal{P}_{OBSTACLE}$.

- C++ containers can be used for data storage as they organise the data in a more efficient way. Some of the container classes are listed below:
  - Multiset;
  - Map;
  - Unordered map.

- These containers use trees to store the data and speed up the search process for accessing the data.

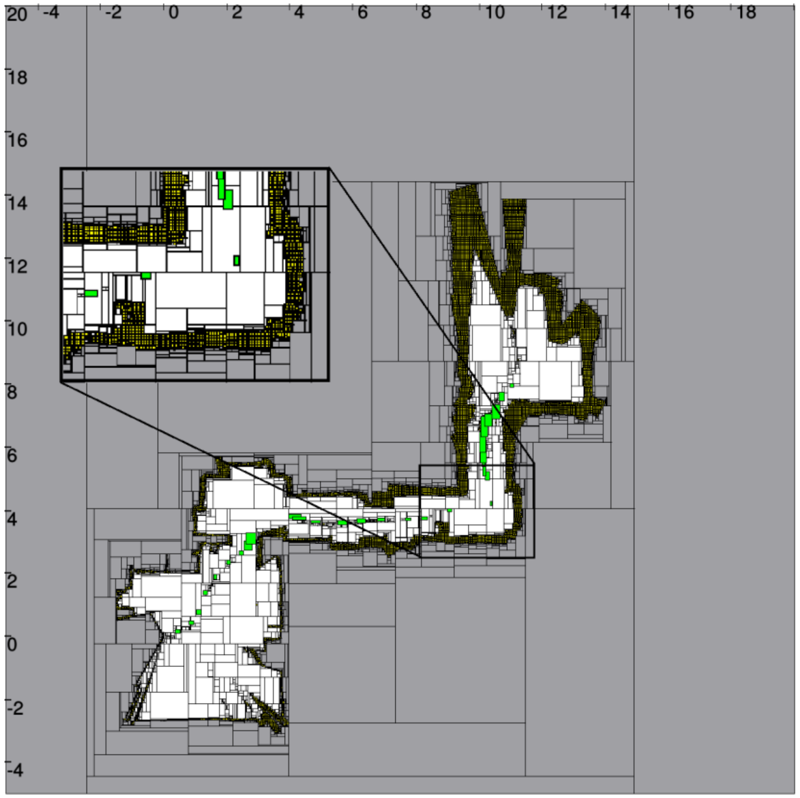
Interval SLAM – computing optimisations

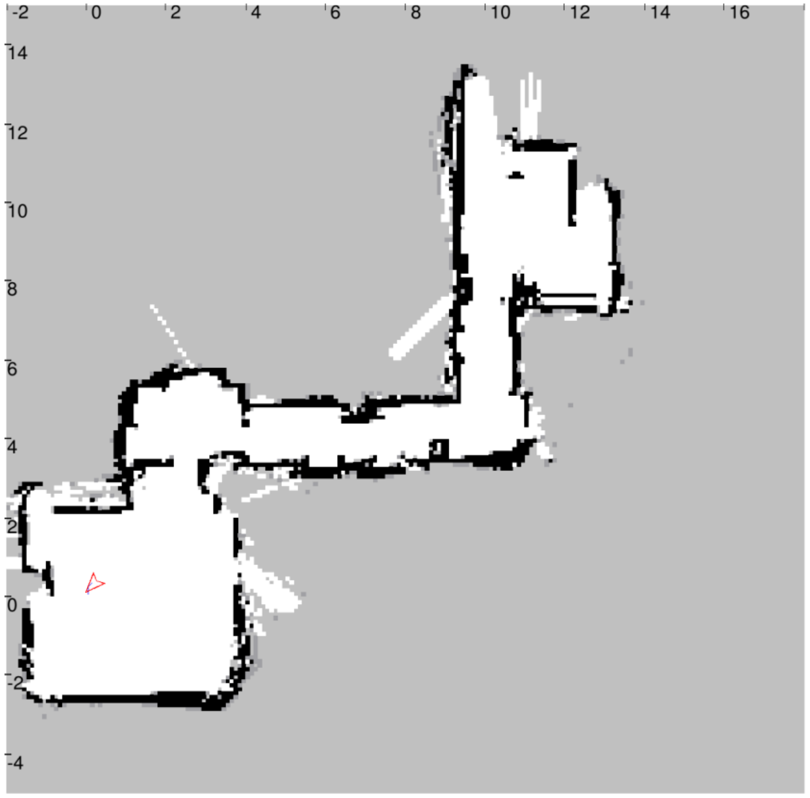
- A better approach is to store the data in a grid structure. The space is partitioned into cells and each box in the map is allocated to its corresponding cell.



Interval SLAM – results

- Comparison between an interval map and a probabilistic map. The interval map accumulates the uncertainty over time, as the robot moves inside the unknown environment.





Conclusions

- A new Interval SLAM approach using all available measurements is developed.
- The method is tested and showed to be computationally feasible for a 2D map built with a LiDAR sensor.
- Future work:
  - Extend the method for 3D environments;
  - Use parallel computing to allow bigger measurement data sets;