System Application Report

SWARM SLAM

by:

Elouan Autret

Under the supervision of:

Luc Jaulin
Fabrice Le Bars

Option Systèmes Perception Information Décision

Abstract

During underwater missions, autonomous robots have not access to a GPS positioning and must rely on other sensors to evaluate its position. The method used is a range only (with landmarks) SLAM with Interval Analysis this method allow a guaranteed estimation of the errors.

When sweeping an unknown and hazardous area it is important to known for sure the position of the robot, to avoid obstacles or keep from damaging the environment. The method used to control the robot is the potential field method which is adapted to an unknown environment. The implementation uses the ROS framework for communication and visualization.

The SLAM and control show promising results but need more tests and some tweaking for a more complete simulation.

Contents

Acknowledgement

First of all, I would like to thank my supervisor during the project,Luc Jaulin, who were available for all my interrogations and ensured that I was not overwhelmed by my pretensions.

I also would like to thanks Simon Rohou and Thomas Le Mézo for supporting my incessant interruptions for information requests on Ibex and discussing my work.

And finally, I would like to thank Jordan Ninin for his help in improving the efficiency of my program.

List of Figures

Introduction

In recent years the world has seen a multiplication of robots (we will consider a robot as a machine which does not need human interactions during its runtime in the doing of its task – for example an autonomous robot instead of a remotely operated vehicles-) but they stay mainly in known places it is sure for robots that does not need to move but, even for mobile robots such as factory robot that have to move products, they generally do not go in unknown places. This is because a robot needs a good localization to carry out its tasks. In outdoor and open environment this localization can be given by a GPS (Global Positioning System) like for the Google Car, an autonomous vehicle that can go into traffic. But in some environments this information is not available, underwater and in covert environments (indoor, dense forest). In such places the robot has to ask itself two questions:

- What is my environment?

- Where am I?

Those questions correspond to the SLAM problem, SLAM is an acronym for Simultaneous Localization and Mapping, this corresponds to the situation of the chicken and the egg because to know where is the robot must know the map and to map the environment it needs to know it.

The context of our problem is the localization of underwater vehicles in order to explore the environment (to detect mines for example), their localization would be done with the help of beacons with unknown positions (dropped on the zone before the manoeuvre). This problem can be considered as a SLAM problem.
More than one robot would be deployed, this means that they can collaborate in order to be more efficient, for faster exploration and more adaptable. Using multiple robots can be viewed as using a swarm of robots, and for the swarm to be efficient, it can be useful for the robots to share the information and their mapping data (this can be used to avoid robots exploring the same area) this initiates the problem of the map merging.

This report will explain different methods of SLAM and map merging with the goal of choosing one that will be more appropriate for the task.Then it will present the chosen method and its implementation.

Chapter 1

Status Report

1.1 Representation

In general a slam output will give either a topological map or an Occupancy Grid

1.1.1 Topological Map

If the algorithm for the SLAM uses landmarks there is chance that it uses a topological map. This type of map will put in relation different landmarks as a graph, the nodes will be the landmarks and the arc can be the path, the distance between the landmarks, for example an underground map is a topological map with stations such as the landmarks and relations are the connections between the stations:

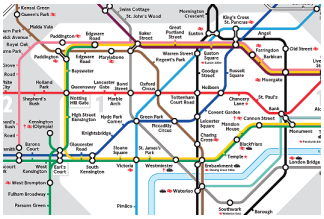


Figure 1.1: Example of a topological map.

This type of map will be made when sensors can distinguish features in the environment. This type of map has a higher abstraction level, it does not know the metrics of the environment but it can consider objects in it:

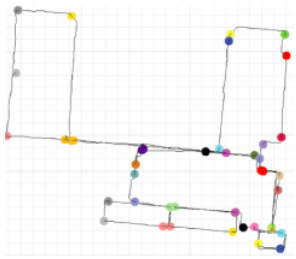


Figure 1.2: Example of a SLAM topological map [1].

In figure 1.2 the landmarks are corners and intersections.

1.1.2 Occupancy Grid

An occupancy grid will represent the continuous space divided in a 2 or 3D grid where each square or box contains the probability of being occupied, in figure 3 the squares are in a trinary state (empty , occupied or unknown):

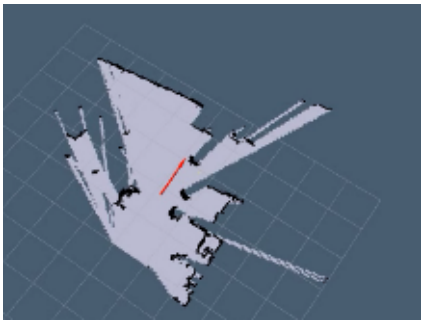


Figure 1.3: 2D Occupancy Grid of the Robotic Club in ENSTA Bretagne with Hector Slam [2].

1.2 Classical methods for SLAM

In general when using a SLAM the state equation of the robot is known and helps the localization problem, the state equation models the behaviour of the robot in function of input parameters:

$$\dot{x} = v * cos(\theta) \tag{1.1}$$
$$\dot{y} = v * sin(\theta) \tag{1.2}$$
$$\dot{\theta} = u_1 \tag{1.3}$$
$$\dot{v} = u_2 \tag{1.4}$$

Above is a state equation modelling a car (x and y are the position, θ is the heading and v the speed), with input u_1 and u_2 which are known (the robot computes them). They are the most used and studied method and many use probability and decision theory to estimate the pose of the robots and the map such as the EKF Slam.

1.2.1 EKF SLAM

An EKF SLAM is a slam algorithm that uses the Extended Kalman filter to accomplish the problem, it is a features based slam.

Kalman Filter

The Kalman Filter is used to estimate the linear state equation variable over time keeping track of the uncertainties of the variables, this filter permits data fusion:

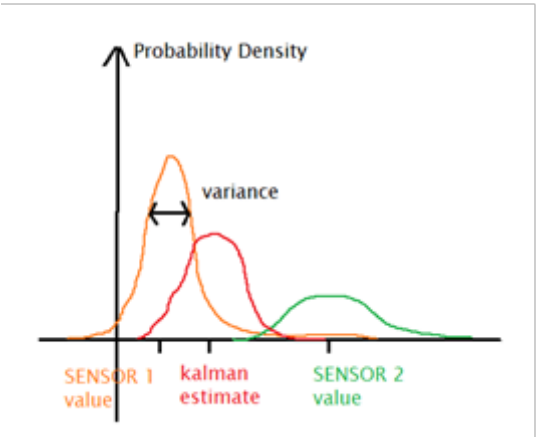


Figure 1.4: Simple example of data fusion.

If we have two sensors that measure the same variable the Kalman filter will combine them in function of their variances (a sensor with great variance means that it is less precise so it is less trustworthy whereas a sensor with little variance will be more precise), in figure 1.4 we combine two sensors with a Gaussian noise, this is a simple example of what the Kalman filter can do, it can also combine information between the state variables, (a state variable can be the position, the speed, angular speed) and if we have relations between those variables (position and speed) the Kalman filter will be able to combine the estimates.

Therefore in what way is the Kalman filter used for the slam? The landmarks detected are put in the state equation in order to estimate their position and use them to improve the robot localisation:

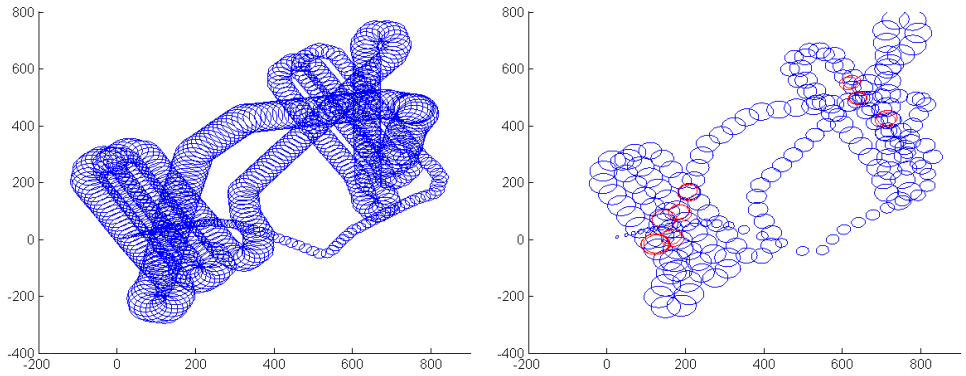


Figure 1.5: Estimate of position of an underwater vehicle without and with landmarks using a Kalman Filter.

In figure 1.5 the Kalman filter only uses the model of the underwater vehicle to estimate its position therefore the variance (represented by an ellipse) always increases whereas when landmarks are available it can slow the uncertainty propagation. The extended Kalman filter is used to work on nonlinear problems such as the SLAM problem, thus causing a bit of approximation.

1.2.2 FastSLAM

The FastSLAM algorithm [7] uses a particle filter and a Kalman filter to solve the SLAM problem, so this is also a landmark based algorithm.
The FastSLAM algorithm creates many particles that represent hypothesis on the position of the robot, the particles are randomly placed over the possible position of the robot. Then the landmarks are estimated for each particle using the Extended Kalman filter (supposing the position of the robot known for the particle).

After that the likeliness of each particle being the right hypothesis is computed. Some particles will be more likely to be right than others, therefore a resampling is done near the most likely particles found just before, the particles are now with the same likeliness (weight). The particles are kept through time and are updated (with each particle having different movements and new weights depending of the movement likeliness). Through time, the particles are supposed to regroup around the correct position of the robot [8].

1.3 SLAM via Interval Analysis

The methods seen approach the problem from a probabilistic point of view but, are not guaranteed, it relies on the probability of the robot to not be an outlier, and when we need a sure position of the robot it could lead to a failure (in a harmful environment or in interaction with humans it would be hazardous). The interval analysis means to guarantee the result, and is not subject to approximation because it does not need to linearise the problem. Interval Analysis has grown to be a competent alternative to the probabilistic methods.

1.3.1 Interval Analysis

In the Intervals theory a real number is replaced by an interval with an uncertainty represented by its width that includes the right number, for a known x the corresponding interval could be $[x, -\epsilon, x + \epsilon]$ with a width of 2ϵ.

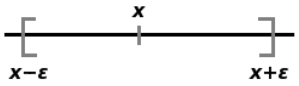


Figure 1.6: Representation of an interval.

This representation has advantages, to easily manage uncertainties by enclosing them in an interval, but the intervals can be contracted around the feasible value thus reducing the uncertainty.

This representation have advantages, to easily manage uncertainties by enclosing them in an interval, but the intervals can be contracted around the feasible value thus reducing the uncertainty.
For example considering the sets $X = [1, 7]$ and $Y = [-1, 5]$ and if Y and X are constraint by the equation $y = x^2$ then we can contract the set Y:

$$Y = Y \cap X^2 = [1, 5]$$

But X can also be contracted by going backwards:

$$X = X \cap \sqrt{Y} = [0, \sqrt{5}]$$

This operation is a forward-backward contractor [9] and [10].

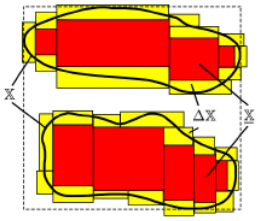


Figure 1.7: Sub paving of a 2 dimensional set [3].

The set inversion [5], finding X such as $f([X])$ is in S, is easy to compute with interval arithmetic, if X is the solution set (see above, the red boxes are included in the set, yellow boxes are on the border), the minimal size of the boxes correspond to the precision for the set inversion.
One of the first papers to address the SLAM problem via interval analysis was in [4], in this paper they proposed an algorithm for the mapping and localization using landmarks.

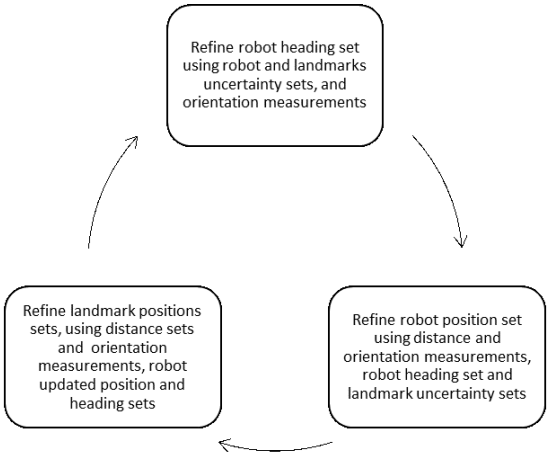


Figure 1.8: Algorithm process for the SLAM [4].

In their case, figure 7, they considered receiving bearing and distance of landmarks, by using constraints propagation (contractors) we can refine the sets [11]

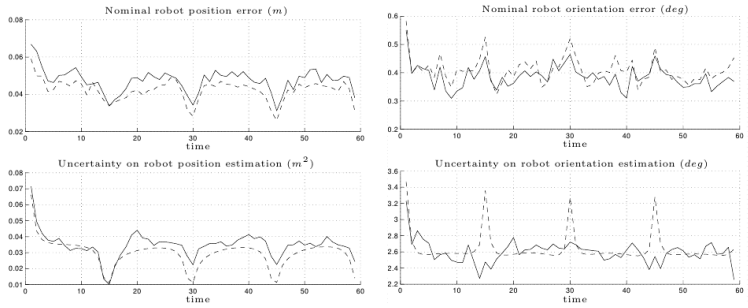


Figure 1.9: Error differences between EKF-SLAM and Interval SLAM in [4].

The interval method can be as precise as the EKF-SLAM but with having a guaranteed uncertainty.

1.4 With Multiple Robots

By using multiple robots the possibilities are expanded but under certain conditions, such as, they do not interfere with each other and do not redo the work of others. For exploration it means that they have to know where the others are or have been, in order to do that the robots need to merge their map to identify their common or uncommon environments.
This problem can be represented as a graph where the nodes are the positions of the robots at a certain time and the arcs are the transformation matrix from a node to another:

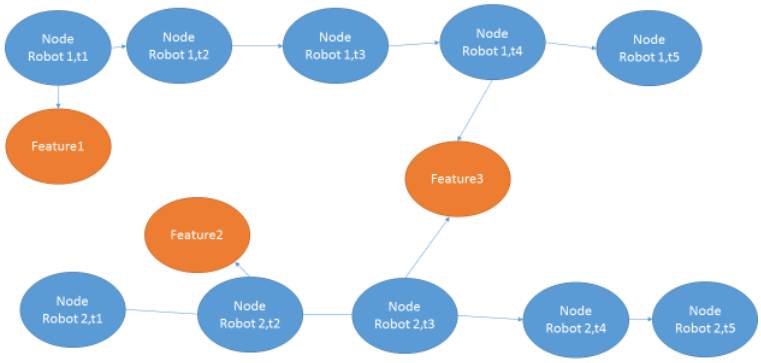


Figure 1.10: Graph representation of the Multi-robot SLAM problem.

The goal of this problem is to make the link between robot 1 and robot 2 (with the feature 3 in figure 9) of course in some contexts a sole link between the graphs would not be sufficient (with sensors getting only range), and would need two or three links two associate the graph. Another method would be to iterate over each feature from the graph of a robot and try to match it to the features of another graph, it is the same reasoning as the ICP (Iterative Closest Point) method use for matching frames for a 3D mapping.

1.5 Direction

Considering the context of the mission, a swarm of underwater vehicles which can communicate between themselves and can know their distances to beacons placed in the water before, in order to do an area scanning. The interval method will be used as it is easier to deal with non-linear problems with it. The map merging problem will be easily dismissed by linking the graph at the start of the mission, it can assumed the robot will have a GPS information when they are at the surface, so the robots will be able to share their estimations in a same frame.

Chapter 2

Theoretic

If the context of the mission is established, the property of the robots are not :

- The robots will have a maximum range of intercommunication;
- The robots will have a maximum range of beacons detection;
- The robots have access to their heading and speed;
- The beacons are recognizable;
- The beacons do not move during the mission;
- The beacons can ping themselves and send information to the robots;
- The distance information and communication are not available continuously.

2.1 SLAM

As seen in 1.3 the SLAM with recognizable landmarks can be done (and has been done) with constraint propagation.

2.1.1 One robot

Contractors

For the SLAM two contractors are needed (see 1.3.1), a distance contractor between the robot and a beacon position, for the beacons between themselves (as they can know their distances), a robot state contractor to spread the constraint over the time.

Distance Contractor The Distance contractor is based on a simple function, the Euclidean distance:

$$(R_x - B_x)^2 + (R_y - B_y)^2 = d_{RB} \tag{2.1}$$

Where R is the box containing the robot and B the box for the beacon and d_{RB} the interval of the measure. This contractor with the Euclidean distance is not very efficient, indeed if the pose of the robot is known but the beacon has not been estimated yet, the output will be the box containing the circle with the measured distance as a rayon surrounding the robot.To improve the computation the contracted box can be cut

into a grid and then the different cases would be fed to the contractor and the remaining non-empty boxes would be kept.

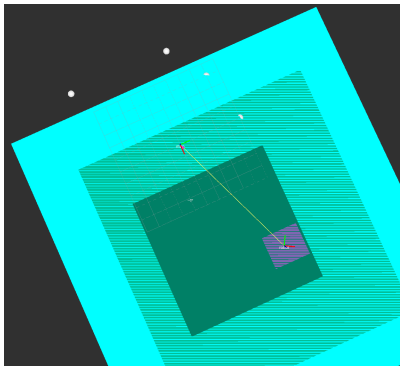


Figure 2.1: Beacons Boxes computed without paving.

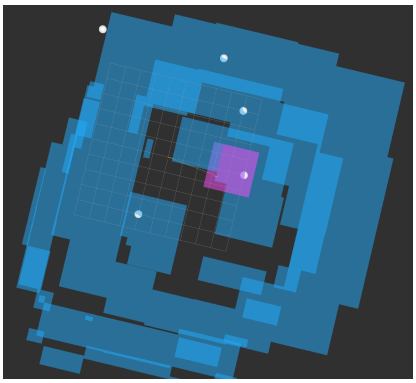


Figure 2.2: Beacons Box computed with paving with SIVIA [5].

In figure 2.1 the position will be difficult to improve and in 2.2 the estimates of the beacons are better as the space between the robot and a beacon is not considered as a possible position.

Robot State Contractor The interest of this contractor is to use a the contraction made by a beacon a time t_1 to improve the estimate the robot over the time which can then improve the estimate of a beacon via a measure at a time t_2.

The contractor is made by knowing how the robot can move therefore the state equation of the robot is needed, for the report the state equation of a car is used:

$$\dot{x} = v * cos(\theta) \tag{2.2}$$
$$\dot{y} = v * sin(\theta) \tag{2.3}$$
$$\dot{\theta} = u_1 \tag{2.4}$$
$$\dot{v} = u_2 \tag{2.5}$$

Then considering u the input of the robot, dt the incrementation duration, the time t_1 and the time $t_2 = t_1 + dt$ and $X(t)$ the state vector at the time t the contractor is based on the following equation:

$$X_0(t_2) = X_0(t_1) + X_3(t_1) * \cos(X_2(t_1)) * dt \tag{2.6}$$
$$X_1(t_2) = X_1(t_1) + X_3(t_1) * \sin(X_2(t_1)) * dt \tag{2.7}$$
$$X_2(t_2) = X_2(t_1) + u_1 * dt \tag{2.8}$$
$$X_3(t_2) = X_3(t_1) + u_2 * dt \tag{2.9}$$

Combining the information

The slam is supposed to be online (done in real time) this mean as for the control of a robot an estimate of the position is needed continuously (> 10 Hz). At each iteration the algorithm 1 is processed.

Algorithm 1 Process information for an iteration

Require: $v(t)$,$\theta(t)$,$u(t)$
X : (State vector from the precedent iteration)
$[\,dt]$:(interval containing the duration from the last iteration)
$measure_{beacons}$: (set of measures between two iteration)
$Past$ (vector recording of robot estimate and measure)
n(size of Past)
$distC$: contractor of distance
$robotStateC$: contractor of robot estimate
$update$: function which update the estimate of X by taking into account the proprioceptive data.
$box_{beacons}$: Set of the estimate of the beacons

$X \leftarrow update(X, [\,dt]\, v(t), \theta(t), u(t))$
$Past \leftarrow Past + \{(X, [\,dt]\,, measure_{beacons})\}$
$n \leftarrow n + 1$
if $measure_{beacons} \neq \emptyset$ **then**
  for $i = n-1; i > 1; i--$ **do**
    $(X_i, [\,dt]\,_i, measure_{beacons,i}) \leftarrow Past(i)$
    $(X_{i+1}, [\,dt]\,_{i+1}, measure_{beacons,i+1}) \leftarrow Past(i+1)$
    for $measure_{beacon,i} \in measure_{beacons,i}$ **do**
      $(X_i, box_{beacon}, measure_{beacon,i}) \leftarrow distC(X_i, box_{beacon}, measure_{beacon,i})$
    end for
    $(X_i, X_{i+1}, [\,dt]\,_{i+1}) \leftarrow robotStateC(X_i, X_{i+1}, [\,dt]\,_{i+1})$
  end for
  for $i = 0; i < n-1; i++$ **do**
    $(X_i, [\,dt]\,_i, measure_{beacons,i}) \leftarrow Past(i)$
    $(X_{i+1}, [\,dt]\,_{i+1}, measure_{beacons,i+1}) \leftarrow Past(i+1)$
    for $measure_{beacon,i} \in measure_{beacons,i}$ **do**
      $(X_i, measure_{beacon,i}) \leftarrow distC(X_i, measure_{beacon,i})$
    end for
    $(X_i, X_{i+1}, [\,dt]\,_{i+1}) \leftarrow robotStateC(X_i, X_{i+1}, [\,dt]\,_{i+1})$
  end for
end if

This algorithm does not use all the information available when using one robots (distance between beacons) and does not use a paving for the estimation of the beacons:

When receiving data from a beacon a contractor over the distance between beacon can be computed :

Algorithm 2 Process distance between beacons

Require: $measure_{beacons}$: (set of measures between two iteration)
$box_{beacons}$: Set of the estimate of the beacons
$Bsender$: beacon which has transmitted the data

for $measure_{beacon} \in measure_{beacons}$ **do**
  $(box_{Bsender}, box_{beacon}, measure_{beacon}) \leftarrow distC(box_{Bsender}, box_{beacon}, measure_{beacon})$
end for

But as written in 2.1.1 the estimate box for the beacons can be paved, that change the precedents algorithms when treating with a beacon estimate (8,16,1). An algorithm for the distance constraint between two set of boxes must be executed:

Algorithm 3 Distance Constraint Application on two set of boxes

Require: d: interval of the distance
$\mathbb{B}_1$: A set of boxes
$\mathbb{B}_2$: A second set of boxes
$\mathbb{T}_1 \leftarrow$ Vector of empty boxes of the same size of $\mathbb{B}_1$
$\mathbb{T}_2 \leftarrow$ Vector of empty boxes of the same size of $\mathbb{B}_2$
for $box_1 \in \mathbb{B}_1$ **do**
  for $box_2 \in \mathbb{B}_2$ **do**
    $(box_{1t}, box_{2t}, d) \leftarrow distC(box_1, box_2, d)$
    $\mathbb{T}_1(box_1) \leftarrow \mathbb{T}_1(box_1) \cup box_{1t}$
    $\mathbb{T}_2(box_1) \leftarrow \mathbb{T}_2(box_2) \cup box_{2t}$
  end for
end for
$\mathbb{B}_1 \leftarrow \mathbb{T}_1 \setminus \{box \in \mathbb{T}_1 \mid box = \emptyset\}$
$\mathbb{B}_2 \leftarrow \mathbb{T}_2 \setminus \{box \in \mathbb{T}_2 \mid box = \emptyset\}$

With the algorithms 1, 2, 3 it is possible to do a range only slam with a unique robot (see chapter 4 for more details).

2.1.2 With more robots

When using multiple robots, each robot can get the estimations of the map from the other robots (if there is transmission), in the context of the mission the first position is known (the box for the pose estimate has a little width depending on the GPS data) therefore the position of the boxes is sure between the robots. When receiving data from another robots, one robot will fuse the estimates of the beacons from the other robots with its own estimates:

Algorithm 4 Fusion of beacon estimate between robots

Require: b: a beacon
$\mathbb{B}_1$: set of boxes estimating the beacon b by the robot receiving the data
$\mathbb{B}_2$: set of boxes estimating the beacon b by the robot sending the data
$\mathbb{T}_1 \leftarrow$ Vector of empty boxes of the same size of $\mathbb{B}_1$
for $box_1 \in \mathbb{B}_1$ **do**
  for $box_2 \in \mathbb{B}_2$ **do**
    $\mathbb{T}_1(box_1) \leftarrow \mathbb{T}_1(box_1) \cup (box_1 \cap box_2)$
  end for
end for
$\mathbb{B}_1 \leftarrow \mathbb{T}_1 \setminus \{box \in \mathbb{T}_1 \mid box = \emptyset\}$

This contraction will supposedly greatly improve the estimation of the beacons as

the robots do not start the mission at the same position they will see the world differently, therefore the intersection of their vision will be efficient.

2.2 Controlling the robots

The mission followed by the robots is to scan an area, with multiple robots the area can be divided between the robots (at least one robot by area).

2.2.1 Path Generation

The area is divided in a decided number of cases:

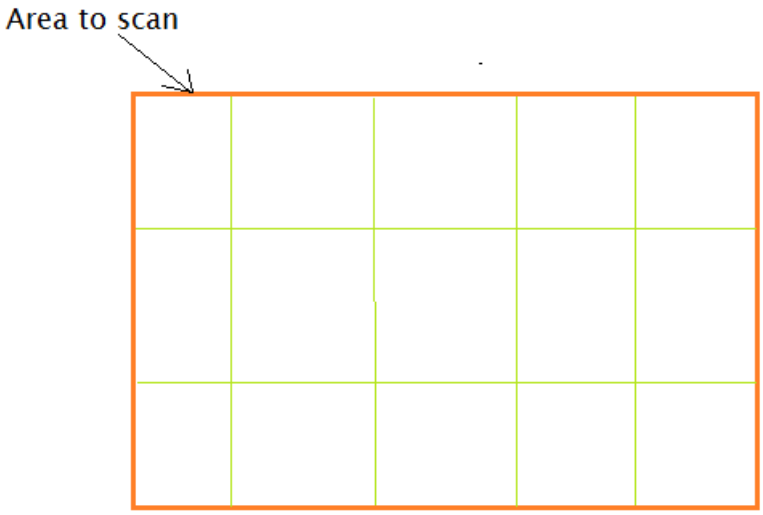


Figure 2.3: Paving of an area to scan.

A robot will be ordered to scan a chosen case, it will make a first pass to acquire the positions of the beacons then a more thorough pass to make scan the case.

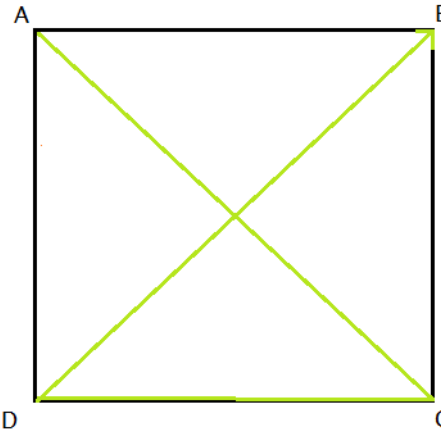


Figure 2.4: Path of the first pass to detect beacons.

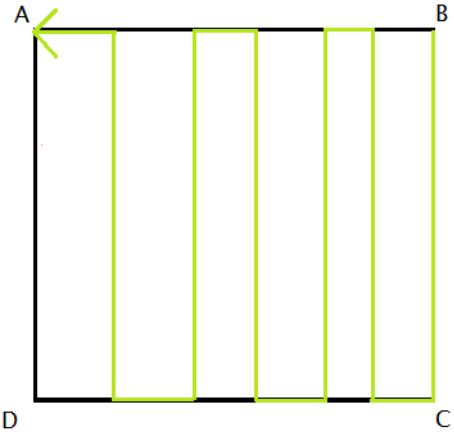


Figure 2.5: Path of the second pass.

This last pass is used very often in area scanning with underwater vehicles.

2.2.2 Following the path and considering the other robots

The method chosen to control the robots for the path following and the obstacles avoidance is the potential field method. It has been chosen for its easy implementation and manipulation as it permit to manage movable obstacles such as the other robots. The robot is seen as an electric particle in an electric field and respond to it with the relation:
$\mathbf{f} = -\text{grad}(V(\mathbf{p}))$ (page 77, [12])

Where $\mathbf{p}$ is the position of the robot and V its potential and $\mathbf{f}$ the force applied to the robot. If the robot need to go to a chosen point,the point will apply an attractive force on the robot and if another robot is approaching it will apply a repulsive force.

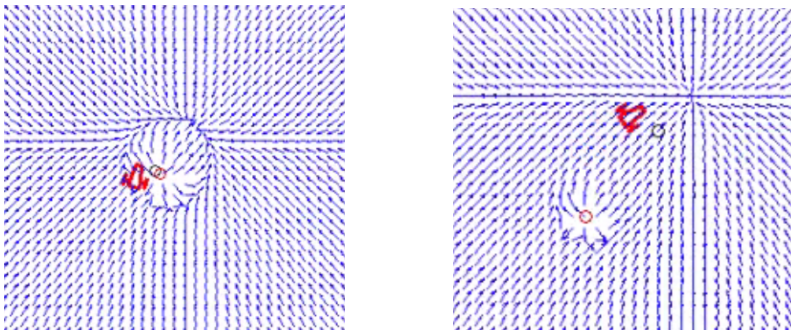


Figure 2.6: Car following an attractive point with potential field method with presence of a repulsive point

Chapter 3

Implementation

3.1 ROS

ROS (Robotic Operating System) is an open source framework made toward robotic application. The goal of ROS is to propose means to facilitate communication between processes,managing possibilities and make available working algorithms for everyone [13].

This framework will be use in this project for communication and visualisation.

3.1.1 How it works

ROS is functioning with nodes, a master and slaves,the master is a ROS process and nodes can be programmed by anyone:

- The framework is available on many OS (operating system)such as Linux (CPU architecture: ARM x86 x64), Android , Windows, OSX;
- The programming of the nodes can be done in multiple language : C++, Python, Java, Lua, Lisp, C#, Go, R, Ruby (from the most supported to the least)
- Nodes written in different languages can function together.

The communication with ROS works with topics,subscriber, and publisher, a publishing node will announce to the master that it is publishing over a topic and the subscribing node will say to the master that it want to listen to a topic. Then if the publisher and the subscriber are on the same topic, the master will transfer data in order for the two other node to communicate directly via TCP/IP or UDP (see figure 3.1).

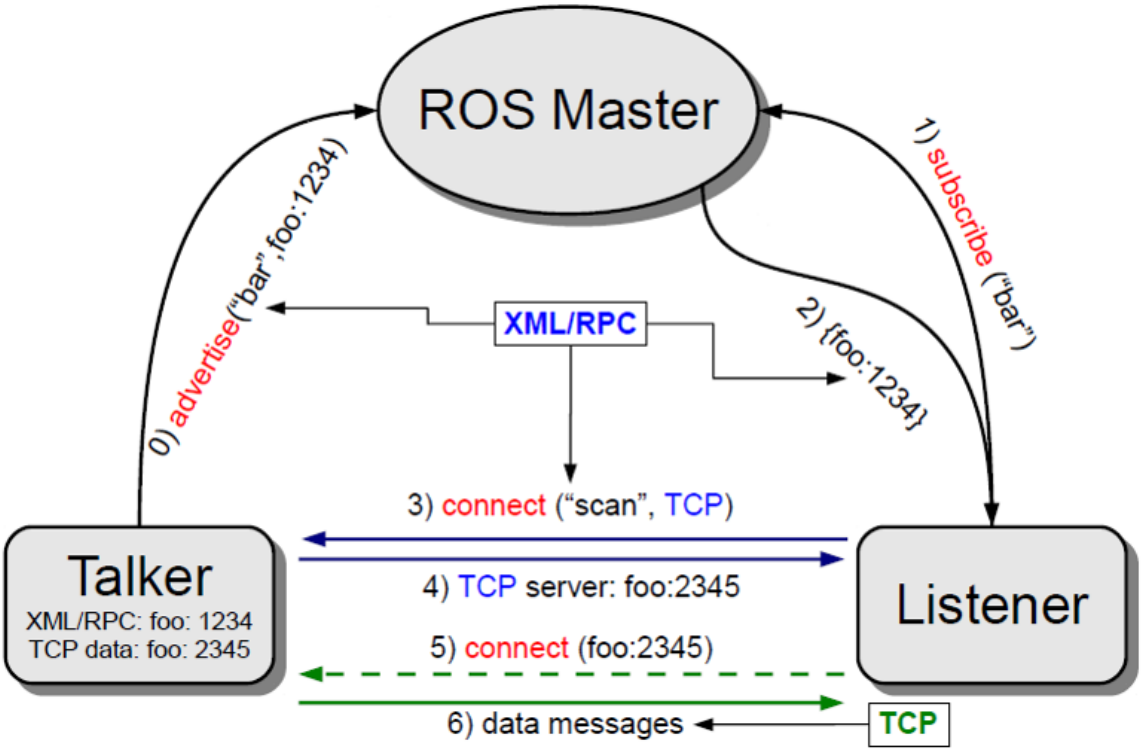


Figure 3.1: Connection of nodes and topics [6].

ROS also facilitate the serialization of messages (conversion of the message into bytes):

std_msgs/Header header

uint32 seq

time stamp

string frame_id

geometry_msgs/Quaternion orientation

float64 x

float64 y

float64 z

float64 w

As it can be seen in the message example, a message can reference another message, in order to create message capable of transmitting complex data (an image with all its meta data). This functionality of ROS permits a big modularity for the programmer, an easy passage from simulation to real via only changing topic names.

3.1.2 ROS in the project

For the simulation of the swarm slam project multiple nodes have been created in order to represent accurate communication and sensing.

IA_MSGS

ia_msgs is a utility package to regroup created message to help to the communication of interval vector (in 2 and 3 dimension):

```
ia_msgs/StampedInterval
  std_msgs/Header header
      uint32 seq
      time stamp
      string frame_id
  ia_msgs/IdInterval[] data
      uint8 id
      ia_msgs/Interv[] data
          geometry_msgs/Point position
              float64 x
              float64 y
              float64 z
          float64 width
          float64 height
```

The above message allows to send with a time stamp and a reference frame a list of an identified list of two-dimensional boxes,it is used to send the estimates of the beacons.

RVIZ_IA_PLUGIN

The rviz plug-in has been done to ease the visualization of the interval, rviz is on openGL based graphic visualizer incorporated in ROS. rviz can receive message and interpret them.

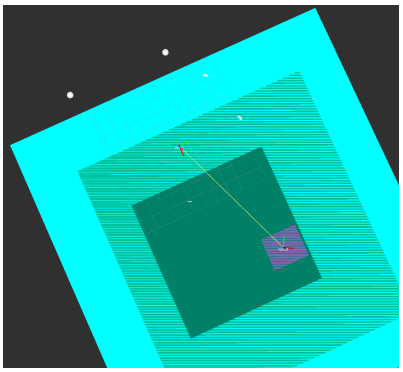


Figure 3.2: Example of rviz utilisation.

In figure 3.2 rviz interpret the StampedInterval message (the blue rectangle), two frame, the real pose of the robot and the map frame seen as an ensemble of blue, green and red arrow, the point cloud of beacons printed as spheres and the Interval message for the pose estimate of the robot. What has been done is the interpretation of the interval related messages.

IA_SLAM

It is the core of the project this node receive the sensor data (distance from the beacons), the information from other robots and the proprioceptive data such as the heading,speed motors order, then it estimate the position of the robot and the beacons by using the algorithm from the section 2.1.It has been implemented in C++ as it create an heavy load on the CPU.

The arithmetic of intervals is done with the help of the IBEX library a C++ library towards intervals applications [14]

ROBOT_SIMU

This node simulates the robot by using the state equation (a car model for the project) its send the pose information to the beacon_simu node and the proprioceptive information to the ia_slam node.This node and all the other simulation nodes have been implemented in Python for easier manipulation.

BEACON_SIMU

This node simulates the sensor of the beacons,it knows the real pose of the beacons and the pose of the robot and considering the distance from the beacons it send or not the information every second (ia_slam received a beacon data every tenth of a second).

TALK_SIMU

This node works on the same principle as the beacon_simu node but for the discussion between robots, if the robots are close enough data is transferred with a limitation in time,a transfer can only happened three seconds after another to model the bandwidth limitation with underwater modem, the time was chosen arbitrarily but can be quickly changed.

IA_CONTROLLER

This node is receiving the position of the robot and the other robots, in order to compute the motors orders to follow the path and avoid obstacles. The orders are then sent to the robot_simu node.

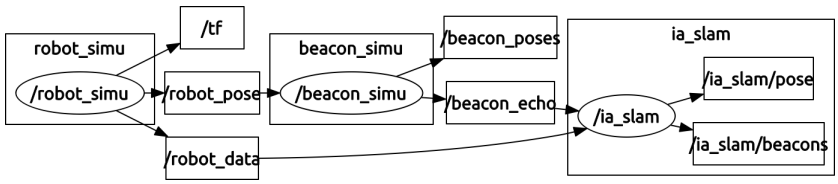


Figure 3.3: Relation between nodes with one robot and no controller node.

In figure 3.3 is represented the relation between nodes with only one robots with the help of rqt_graph, a ROS program. A similar graph can be found in the appendix (figure 5.1).

3.2 Computational problems

3.2.1 SLAM Heavy Load

The complexity of the algorithm 1 for an iteration is $\sim \mathcal{O}(n.(nbDiv))$ where n is the number of iteration passed and $nbDiv$ the maximum number of division of the estimate of the beacons.

Therefore a such an algorithm is not practical for an online slam. The use of the precedent algorithm generates increasing slow-downs over the time.

To prevent those slow-downs the number of iterations can be limited by using the algorithm 1 on the last n_chosen iterations. It induces a loss of information but it is absorbable.

To cut again the load, the propagation constraint can be done with a reduce rate, not following the incoming sensor data. But the rate cannot be too reduce be can then measure of distance would be forgotten without being processed.

3.2.2 Message Delay

A message may be received but with ROS, it can have a important delay depending on the network and CPU load. Thus when receiving a distance to a beacon the robot can have travel a few decimetre and then measure of distance will be wrong when computed.

To address this issue the interval of the measure is inflated of the distance travelled by the robots between the measure and the computation.

Chapter 4

Results

For the tests the positions of beacons are fixed and the initial poses too.

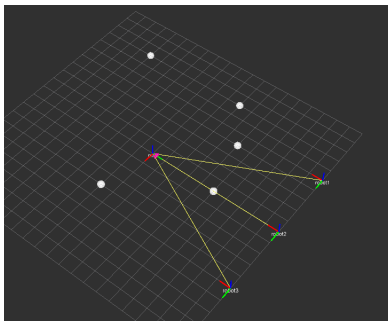


Figure 4.1: Initial Poses of the robots.

4.1 One Robot

For the test, one robot is used with a sensor precision of 10cm and a precision for the starting position of 10cm. A case is 1mx1m.

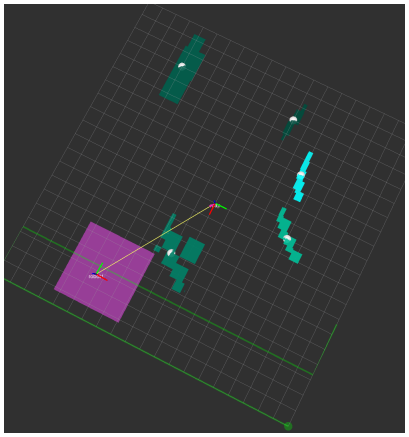


Figure 4.2: Visualization with one robot scanning the area (start).

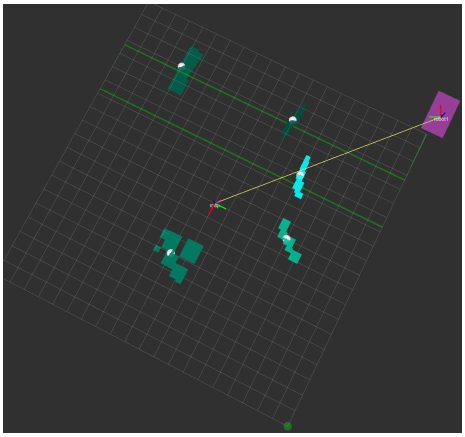


Figure 4.3: Visualization with one robot scanning the area (end).

The images 4.2 and 4.3 are obtained with the robot going directly to the scanning without doing the first pass (see subsection 2.2.1).

The estimate of the beacons has not a good quality,and the estimate boxes for one beacon are spread out around it therefore the estimate of the robot is bad and consequently the controller can't correctly handle the robot and in figure 4.2 the robot does not follow exactly the path (green line) and has an offset.

4.2 Two Robots

For this test, one robot is used with a sensor precision of 10cm and a precision for the starting position of 70cm. In this test the second robot follows the same path as the first one.

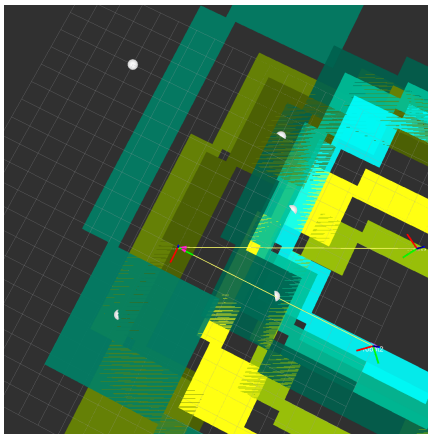


Figure 4.4: Visualization of start of the slam without discussion between robots.

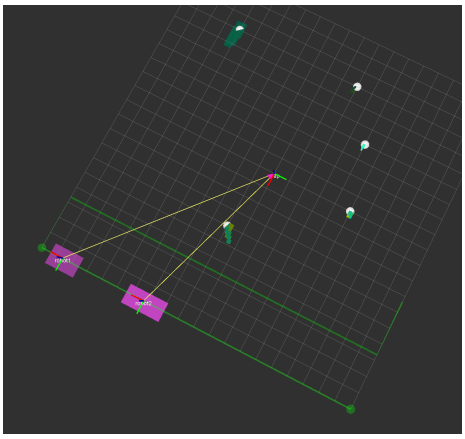


Figure 4.5: Visualization of the two robots following the scanning path.

With two robots communicating the estimations become more precise and allow a better path following(figure 4.5. From them the it could continue to more robots but some errors have happened. Indeed sometime the estimation for a beacon is wrong, the boxes do not include the beacon.

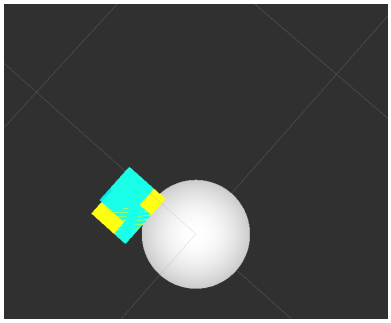


Figure 4.6: Error on the beacon estimate.

The interval method is supposed to give a guaranteed results therefore the problem has to come from the implementation,either there is wrong data in the input,delay

in communication is too important(see subsection 3.2.2), or the state equation used is wrong(but it is working for the robot alone). By repeating the test it can be seen that the problem originates from the start of the test and there is wrong calculation on the pose estimate.

The avoidance algorithm with potential field method is not working smoothly, the position of the robot is supposed to be the middle of the box and the potential is compute with the nearest corner of the other robot box. Then the distances are changing overtime and the chosen point from the other box can also change thus creating an unstable potential field. To address this problem the computation could do the sum of each corner of the two boxes, which would then be the equivalent to compute the potential with the positions being the middle of the box for both robots.

Conclusion

The most needing part in this project was the programming of the slam node but the theory behind is approachable , the state equation linking together the poses over the time, and the measurement linking the beacon to the pose.The implemented version of the algorithm is a bit different from the original, with some safeguards and with limitations due to capacity of the current lining of PC (and embeddable PC).

The scan of an area can be done with the created product but not with a good precision for now due to the remaining errors (in programming), therefore the program is more pessimist on the position of the robots and beacons to compensate those errors.

What could be done in this project in the future:

- The correction of the programming errors crippling the program;
- A re-factoring of the code for a better comprehension from other users
- An optimization of the code to allow remembering more iteration
- Consider three dimension
- Add more path to the controller
- Make experiment on real robot

Chapter 5

Appendix 1

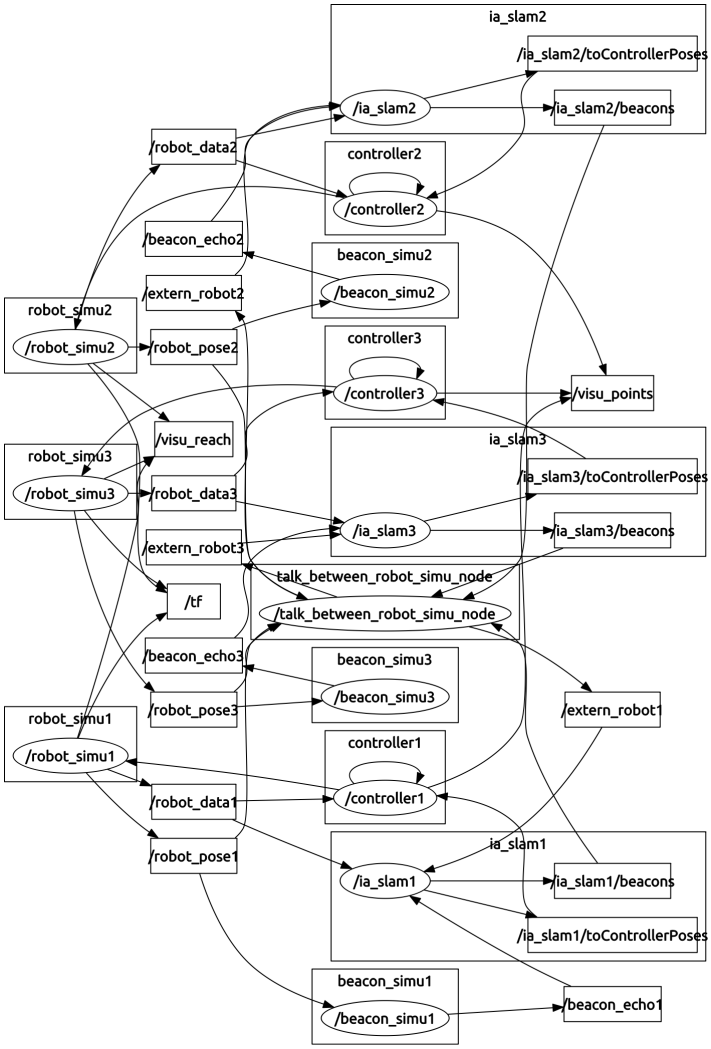
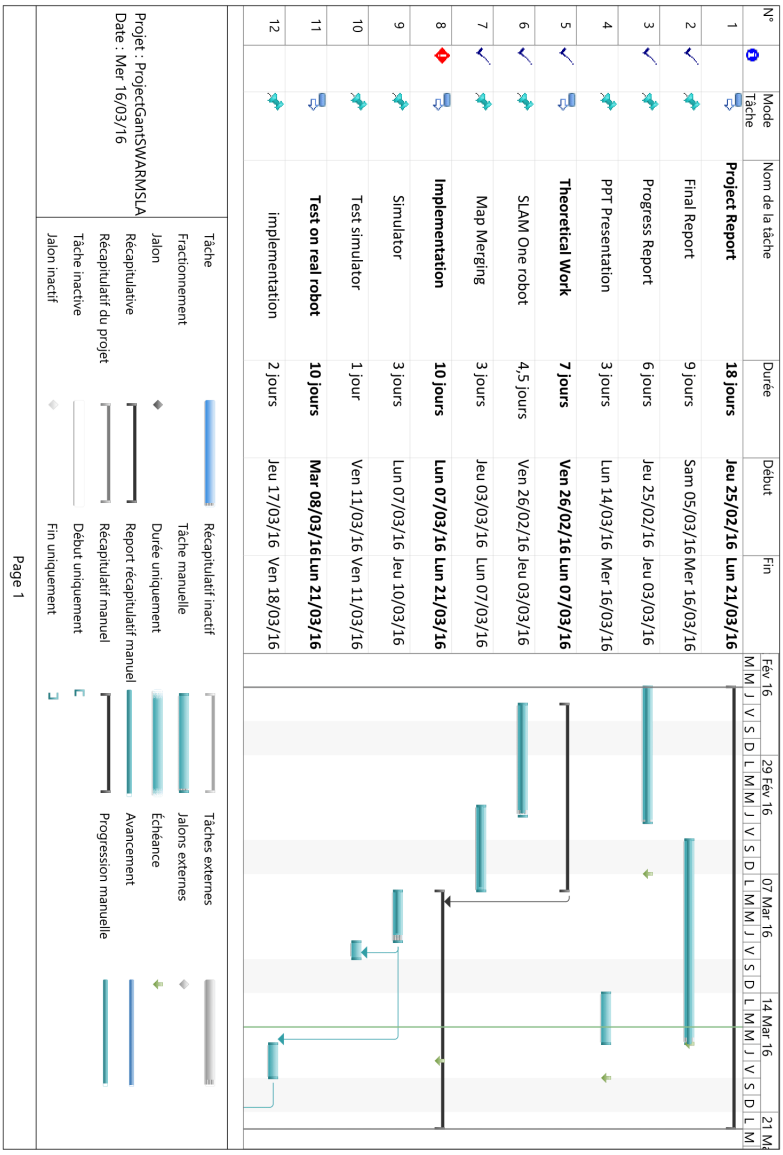


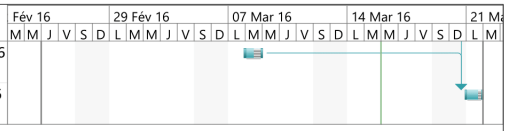
Figure 5.1: Relation between node with a simulation of three robots

Chapter 6

Appendix 2



N°		Mode Tâche	Nom de la tâche	Durée	Début	Fin
13			robots preparation	1 jour	Mar 08/03/16	Mar 08/03/16
14			test	1 jour	Lun 21/03/16	Lun 21/03/16



Projet : ProjectGantSWARMSLA
Date : Mer 16/03/16

Tâche	Récapitulatif inactif	Tâches externes
Fractionnement	Tâche manuelle	Jalons externes
Jalon	Durée uniquement	Échéance
Récapitulative	Report récapitulatif manuel	Avancement
Récapitulatif du projet	Récapitulatif manuel	Progression manuelle
Tâche inactive	Début uniquement	
Jalon inactif	Fin uniquement	

Chapter 7

Appendix 3

How to run the code

Download the Ibex Library

Install ROS

Follow ROS tutorial on how to create packages

In your catkin source directories :

```
$ git clone https://github.com/Elessog/ia_ros.git
$ cd ..
$ catkin_make
$ source devel/setup.bash
```

If Ibex error set IBEX_ROOT in ia_ros/ia_slam/CMakeList.txt

To launch three robots (on different terminal):

```
$ roslaunch beacon\_simu ia\_simu\_control.launch
$ roslaunch ia\_slam ia\_slam\_triple.launch
$ rviz
```

Then set-up rviz via add/byTopic or via the rviz config file and run serviceRob.bash to start the robots (files in ia_ros/).

Bibliography

[1] A. Ranganathan and F. Dellaert, "Online probabilistic topological mapping," *IJRR*, 2010. [Online]. Available: http://frank.dellaert.com/pub/Ranganathan10ijrr.pdf

[2] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 155–160.

[3] A. Bethencourt, "Interval analysis for swarm localization. application to underwater," Ph.D. dissertation, ENSTA Bretagne, 2014.

[4] M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino, "A set theoretic approach to dynamic robot localization and mapping," *Autonomous robots*, vol. 16, no. 1, pp. 23–47, 2004.

[5] L. Jaulin and E. Walter, "Set inversion via interval analysis for nonlinear bounded-error estimation," *Automatica*, vol. 29, no. 4, pp. 1053–1064, 1993.

[6] R. B. Rusu. Robot operating system. Willow GarageT. [Online]. Available: http://wiki.ros.org/Events/CoTeSys-ROS-School?action=AttachFile&do=view&target=ros_tutorial.pdf

[7] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Aaai/iaai*, 2002, pp. 593–598.

[8] A. Svensson. Particle filter explained without equations. UPPSALA UNIVERSITET. [Online]. Available: https://www.youtube.com/watch?v=aUkBa1zMKv4

[9] L. Jaulin, *Applied interval analysis: with examples in parameter and state estimation, robust control and robotics*. Springer Science & Business Media, 2001, vol. 1.

[10] G. Chabert and L. Jaulin, "Contractor programming," *Artificial Intelligence*, vol. 173, no. 11, pp. 1079–1100, 2009.

[11] L. Jaulin and G. Chabert, "Resolution of nonlinear interval problems using symbolic interval arithmetic," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 6, pp. 1035–1040, 2010.

[12] L. Jaulin, *Mobile Robotics*. Elsevier Science, 2015.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[14] Ibex: a c++ numerical library based on interval arithmetic and constraint programming. [Online]. Available: http://www.ibex-lib.org/

[15] M. D. M. A. Garulli and A. G. A. Vicino, "Simultaneous localization and map building for a team of cooperating robots: a set membership approach," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 2, pp. 238–249, Apr 2003.

[16] C. Drocourt, L. Delahoche, E. Brassart, B. Marhic, and A. Clérentin, "Incremental construction of the robot's environmental map using interval analysis," in *Global Optimization and Constraint Satisfaction*. Springer, 2003, pp. 127–141.