# Robot documentation

## Global logic

1. Get target
2. Create particles
3. Loop Localization till x% probability reached
4. Compute global path
   1. Compute next step
   2. Move next step (robot & particles
   3. Adjust localization
      1. If probability < y% go to 2
   4. Check target reached
      1. Yes go to 1
      2. No go to 4.a

## Path management

Use cartography taking into account weight as defined in localization documentation

A A-star search algorithm finds the best path with combined objective to reduce motion cost (minimize distance and rotations) and cartography weight to reduce risk to get in obstacles.

## HMI

Notes

Interface java octave

javaaddpath('C:\Users\jean\Documents\Donnees\eclipse\RobotServer\bin')

javaaddpath('C:\Program Files (x86)\MySQL\MySQL Connector J\src\com\mysql\jdbc') ??

javaaddpath('C:\tmp\robot.jar')

javaclasspath

robot=javaObject("RobotMainServer")

ret= robot.GetLastScanID()

echo = javaObject("EchoRobot")

ret=java\_get(echo,"pendingEcho")

java\_set(echo,"pendingEcho",3)

javamethods(robot)

robot.GetCurrentPosition("1")

### Folders

#### Octave folders

Cd C:\Users\jean\Documents\Donnees\octave\echoLocalization

setupPath.m set the paths

C:\Users\jean\Documents\Donnees\octave\echoLocalization

Contains cartography files and…

Src : contains the octave source codes

@robotJava: define the java object

ToAnalyse: contains file to analyse

Training: contains file and matrix for the training phase

## Object command list

robot=robotJava

robot.SetTraceFileOn(1);

robot.LaunchBatch()

robot.StartTimeoutManagement

robot.NorthAlign(0)

robot.GetRetcode(6,1,2)

robot.GetRetcode(1,1,2)

robot.GetNorthOrientation()

robot.LaunchBatch()

robot.UpdateHardRobotLocation()