Multi-robot patrol simulation

Test environment

Ubuntu 12.04

Boost 1.46 (Debian's default Boost version, installed with Ubuntu 12.04 already)

ROS Hydro and package: stage_ros

LCM v1.0.0 (Lightweight Communications and Marshalling)

Python 2.7 and packages: Scanf, PIL(Python Imaging Library)

Quick Start

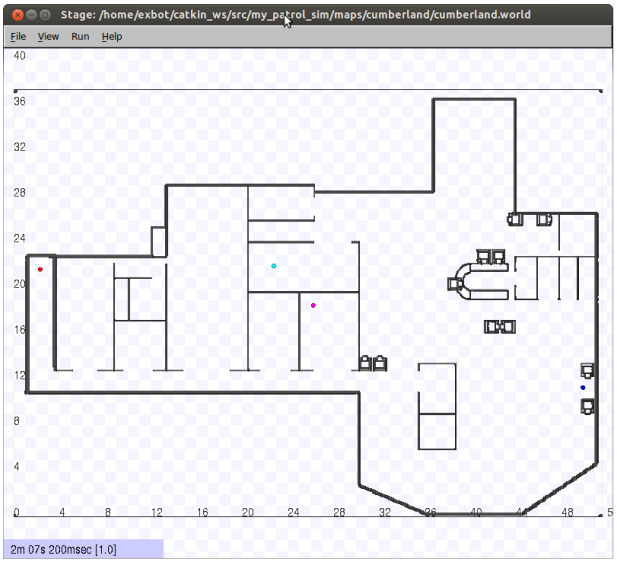
1. Create a ROS Workspace : "~/catkin_ws/src"
2. Move the source folder to "~/catkin_ws/src" and rename it as "my_patrol_sim"
3. Build the workspace:

   $ cd ~/catkin_ws/

   $ catkin_make
4. Open a terminal and run:

   $ source ~/catkin_ws/devel/setup.bash

   $ roslaunch my_patrol_sim navigation_multi_robot.launch



   (Bringup the Stage simulator and ROS navigation module)
5. Open a new terminal and run:

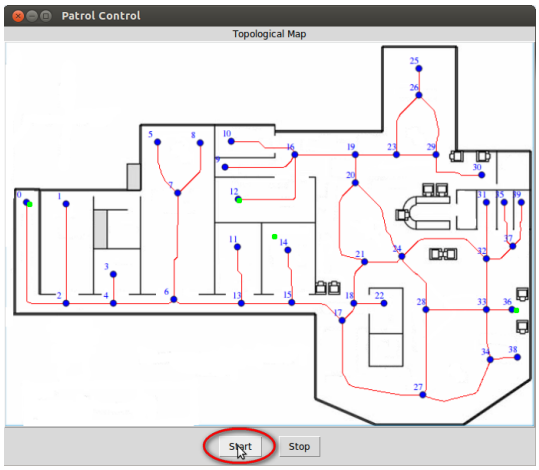
   $ source ~/catkin_ws/devel/setup.bash

   $ roslaunch my_patrol_sim patrol_multi_robot.launch

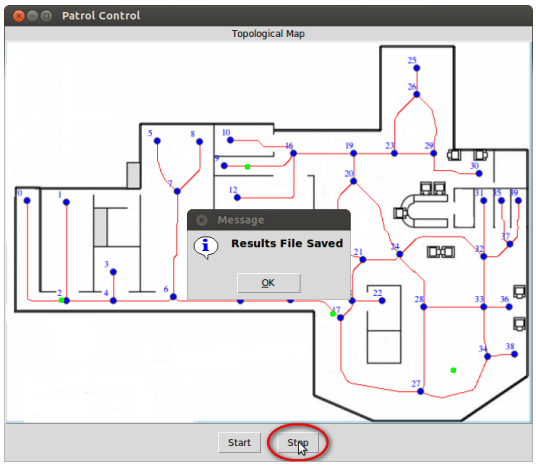
   (Start the patrol decision program)
6. Open a new terminal and run:

   $ cd ~/catkin_ws/src/my_patrol_sim/scripts/my_monitor

   $ python my_monitor.py cumberland 4

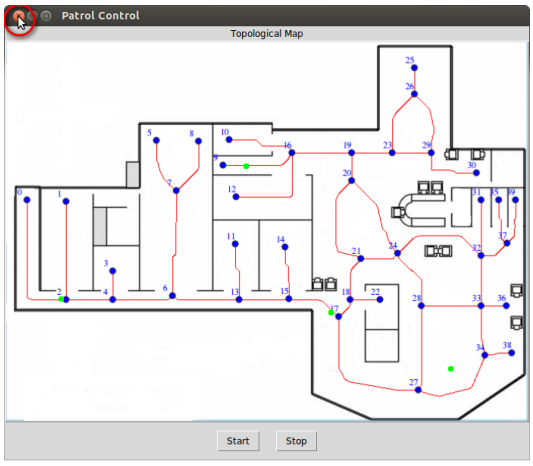


Click the button “Start” to start the simulation.



Click the button “Stop” to stop the simulation.

The statistical result will be saved in the folder “scripts/my_monitor/results”, named as “cumberland_4_results.txt”



Click the Close button in the upper left corner to exit the monitor program.

(Use “Ctrl-C” in the terminal will not make the program exit completely.)

7. To restart the simulation, you need to stop all the programs above and redo step 4 to 6.

8. The statistical result

In the file “cumberland_4_results.txt”, the content is similar to the picture below. The term “VertexID” makes a list of vertex ID in graph cumberland (40 vertices). The term “VisitCount” means the number of visits to the vertex; “MaxIdleness” means the maximum idleness of the vertex during patrolling; “MeanIdleness” means the average idleness and “StdDevIdleness” means the standard deviation. The term “IdlenessRecord” gives the record of each vertex, and the number of items in a row equals to the “VisitCount” of the corresponding vertex.

More information can be found in the source file “scripts/my_monitor/my_monitor.py”.

```
2015-05-10 19:24:24
VertexID 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
VisitCount 5 5 16 5 15 6 13 20 7 3 3 5 4 14 6 13 16 10 12 11 7 8 5 9 11 4 9 12 8 8 3 5 17 14 15 5 6 17 7 5
MaxIdleness 557.700 557.800 405.000 557.100 348.100 559.700 253.900 482.200 559.400 525.600 525.100 485.600 591.300 354.200 394.300 292.900
444.500 264.700 307.000 288.600 332.300 329.800 430.300 410.200 371.400 526.000 505.900 285.800 411.600 488.400 525.600 420.100 295.400
298.900 328.500 420.500 544.600 355.900 349.000 420.300
MeanIdleness 347.860 374.780 120.512 347.700 126.320 297.083 143.823 94.970 265.757 399.267 399.300 327.600 447.350 133.614 305.217 140.900
111.906 168.640 139.025 176.018 276.686 205.675 329.180 179.789 176.173 384.950 173.333 153.383 203.150 192.450 399.533 338.640 106.953
121.414 124.093 338.500 283.300 103.406 262.986 338.480
StdDevIdleness 131.328 112.678 93.329 131.238 95.787 151.331 53.420 115.989 159.420 89.332 88.955 136.307 113.669 110.721 46.714 108.334
122.349 68.383 120.345 44.059 55.520 93.748 54.483 97.150 71.418 81.457 179.662 64.920 96.277 165.204 89.143 54.500 77.161 104.368 118.387
55.137 129.330 113.339 62.248 54.769
IdlenessRecord
(VertexID0) 208.100 207.600 399.300 366.600 557.700
(VertexID1) 206.800 400.100 367.100 557.800 342.100
(VertexID2) 54.400 66.400 86.400 54.000 66.500 87.400 246.000 66.500 86.900 213.700 66.200 86.700 405.000 66.100 86.100 189.900
(VertexID3) 207.000 208.200 399.000 367.200 557.100
(VertexID4) 21.800 185.300 21.500 186.700 21.600 190.600 186.900 21.300 159.000 186.600 23.200 348.100 185.900 23.000 133.300
(VertexID5) 232.300 222.500 399.600 77.700 290.700 559.700
(VertexID6) 147.300 84.600 147.900 74.100 148.500 250.700 116.500 110.100 142.400 148.200 157.000 253.900 88.500
(VertexID7) 38.700 37.900 155.400 38.900 38.200 145.100 39.400 37.900 321.900 39.400 38.400 39.500 38.100 213.500 39.100 38.200 482.200
39.300 38.900 39.400
(VertexID8) 231.900 222.100 399.300 78.300 290.700 559.400 78.600
```

You can import the data into excel like the picture below, or use other methods to process it.

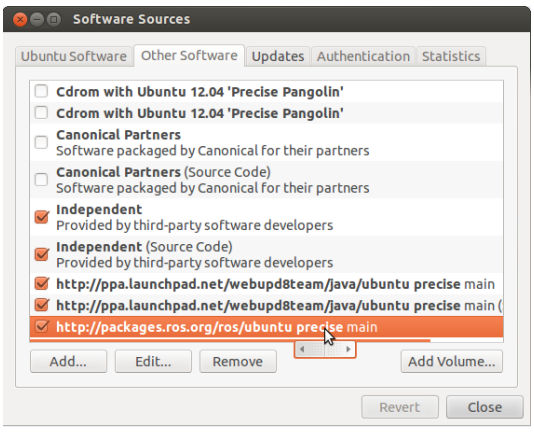
VertexID	0	1	2	3	4	5	6	7	8	9	10	11
VisitCour	5	5	16	5	15	6	13	20	7	3	3	5
MaxIdlene	557.7	557.8	405	557.1	348.1	559.7	253.9	482.2	559.4	525.6	525.1	485.6
MeanIdler	347.86	374.78	120.512	347.7	126.32	297.083	143.823	94.97	265.757	399.267	399.3	327.6
StdDevIdl	131.328	112.678	93.329	131.238	95.787	151.331	53.42	115.989	159.42	89.332	88.955	136.307

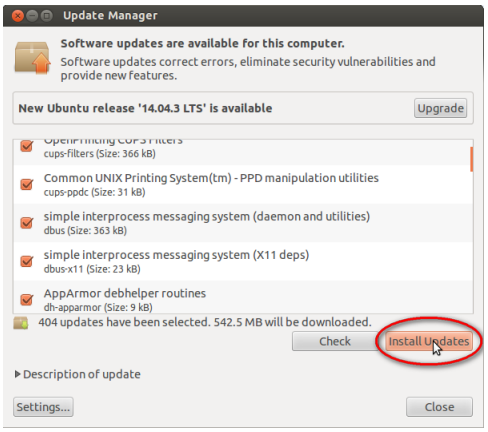
...

Possible problems

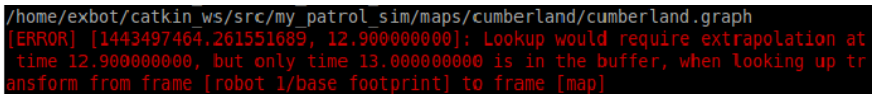
1. If you meet an error like this, it may be the problem related to tf package in ROS. You can install updates to fix it (from http://packages.ros.org/ros/ubuntu precise main).

```
[ERROR] [1443340483.404163114, 37.600000000]: Global Frame: robot_2/odom Plan Frame
size 134: map

[ WARN] [1443340483.404234388, 37.600000000]: Could not transform the global plan to
 the frame of the controller
[ERROR] [1443340483.704490702, 37.900000000]: Extrapolation Error: Lookup would requ
ire extrapolation into the future.  Requested time 37.900000000 but the latest data
is at time 37.800000000, when looking up transform from frame [robot_3/odom] to fram
e [map]

[ERROR] [1443340483.704999738, 37.900000000]: Global Frame: robot_3/odom Plan Frame
size 135: map
```

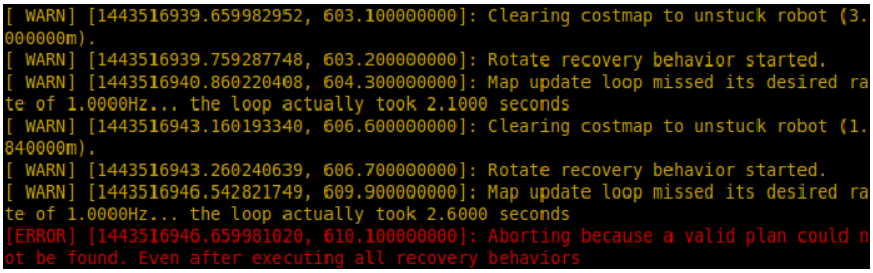




2. An error like this may occur when you run “roslaunch my_patrol_sim patrol_multi_robot.launch”. It is caused by ROS tf transform, and does not affect the program.



3. An error like this may occur in the terminal which runs “roslaunch my_patrol_sim navigation_multi_robot.launch”. It is caused by ROS navigation module, and means that robots are likely to be stuck with each other.



Usage

1. Simulation with different maps and different numbers of robots

The files named “*.world” in folder “maps/cumberland” and “maps/grid” contain the configuration for the map and number of robots for the Stage Simulator.

The file “launch/navigation_multi_robot.launch” contains the navigation configuration according to the number of robots.

The file “launch/patrol_multi_robot.launch” contains the patrolling configuration according to the number of robots.

You can configure these three files to run simulation in any map. You can also use the script “scripts/initialize_sim.py” to configure some specific parameters quickly as follow:

```
$ cd ~/catkin_ws/src/my_patrol_sim/scripts
$ python initialize_sim.py <map_name> <robot_number>
```

The <map_name> can be “cumberland” or ”grid”. The <robot_number> can be “1”, ”2”, ”4”, ”6”, “8” or “12”. For example:

```
$ python initialize_sim.py cumberland 8
```

The script can generate the files “*.world” and “*.launch” according to the input information. More information can be found in the source file.

2. Simulation with different patrol algorithms

The algorithms are stored in source files “src/my_patrol_robot/my_algorithms.cpp” and “src/my_patrol_robot/algorithms.cpp”. The function “TaskExecution::decide_next_vertex(...)” in the file “src/my_patrol_robot/task_execution.cpp” calls one of these algorithms. You can edit this function to choose different patrol algorithms.

3. Structure

my_patrol_sim	data_type_lcm	exlcm	header files for C++ to use LCM messages	
		*.lcm	LCM messages' definition	
	launch	includes	the included files for *.launch	
		*.launch	roslaunch files used to start simulations	
	maps	cumberland	files for the map cumberland	
		grid	files for the map grid	
		initial_poses.txt	file that stores initial positions of robots	
	param_sim	*.yaml	configurations for navigation	
	scripts	my_monitor	exlcm	files for Python to use LCM messages
			maps	the same as "my_patrol_sim/maps" folder used by "my_monitor.py"
			plan_files	files for the partition plan used by "plan_center.py"
			results	results of simulations *_results.txt : average idleness etc. *_robotpath.txt : paths of robots *_timeidleness.txt : instantaneous idleness along time
			my_monitor.py	the monitor program
			plan_center.py	program used for the partition plan
		initialize_sim.py	python script to configure simulations	
	src	my_patrol_robot	getgraph.h/cpp	functions to read the *.graph file
			algorithms.h/cpp	functions of different patrol algorithms
			my_algorithms.h/cpp	
			global_variables.h/cpp	global variables for multi thread
			message_listener.h/cpp	thread that listens to LCM messages
			task_execution.h/cpp	thread that executes the patrol task
			my_patrol_robot.cpp	the main thread
	CMakeLists.txt	configurations for ROS catkin make		
	package.xml			

4. Simulation with special configurations

If you want to change the velocity of the robot, you can refer to files "launch/hete_*.launch" to override some parameters, or directly edit your launch files.

If you want to test the impact of unreliable communications, you can edit the file "src/my_patrol_robot/message_listener.h".

The program would ignore the received message with a certain probability, so as to simulate the error rate of communication. The variable "int Handler::com_rate_" which represents the percentage of reliability can be set from 0 to 100, and 100 means to ignore no message.

The program would only receive messages from robots whose distance to the robot is less than "int Handler::com_dist_" meter, which represents the restriction on communication range. You can also set it to a large number, so as to relax this restriction.

Summary

The programming style of this package may be not good. The selection of different algorithms need to modify the source code and rebuild, rather than pass parameters easily. It is grateful for the effort to improve this package. Any questions, please email ycb11@mails.tsinghua.edu.cn .