Implementation and Test of Two High-level Controllers for Controlling a Mobile Robot

Mohammad Rami Koujan & Yogesh Langhe

Vibot10

May 29, 2016

Outline

- Introduction
- Stage simulator
- Deliberative Architecture based Controller
  - Results and Demo
- Behavioral Architecture based Controller
  - Results and Demo
- Comparison
- Conclusion

Introduction

- Main task:
  - Send a TurtleBot to a set of goals inside its environment
- Two adopted methods:
  1. DBA with RRT path planner
  2. BBA with Tangent Bug path planner
- Implementation platform
  - TurtleBot in ROS with Stage simulator

Stage Simulator

- Robot simulation tool with Hardware Abstraction Layer
- Simple and Computationally cheap
- Several physics-based models for robot sensors and actuators
- Supports research into multi-agent autonomous systems

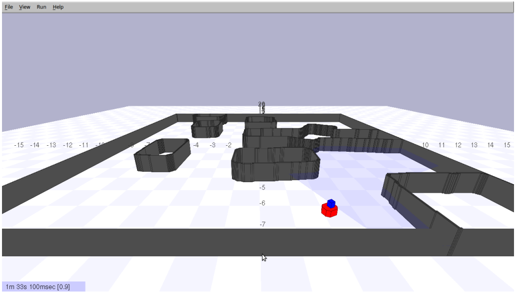


Figure: Stage simulator

Deliberative Architecture based Controller

- A top-down fashion
- Sequential processing
- Hierarchical
  - division of the mission

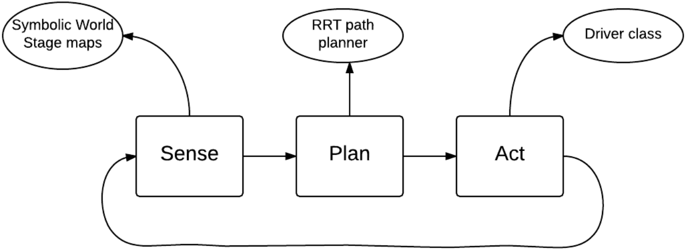


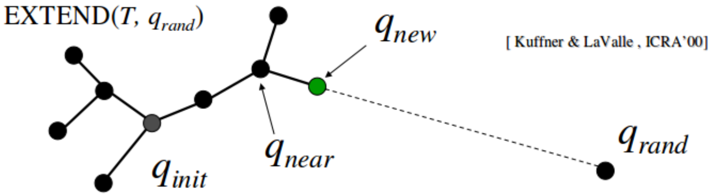
Figure: Hierarchical Paradigm schema

Deliberative Architecture based Controller

- Sense phase
  - Represented by 3 files
    1. A .world file
       - Defines the map
    2. A .inc (include) file
       - Defines global models
    3. A .cfg (configuration) file
       - Translate the map to the simulator
- Plan phase
  - RRT path planner
    1. Rapidly Exploring Random Trees
    2. Path Smoothing based on greedy approach

- Rapidly Exploring Random Trees

```
BUILD_RRT (q_init) {
  T.init(q_init);
  for k = 1 to K do
    q_rand = RANDOM_CONFIG();
    EXTEND(T, q_rand)
}
```



RI 16-735, Howie Choset with slides from James Kuffner

- Path Smoothing based on greedy approach

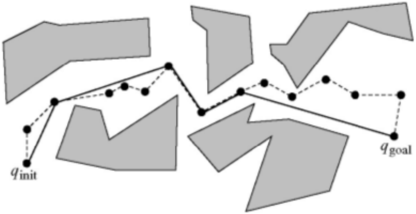


Figure: Path smoothing using greedy approach

Deliberative Architecture based Controller

- Act phase
  -Driver class with the following tasks
  1. Loading a set of goals from the planner
  2. Obtaining TurtleBot's current position
  3. Sending velocity commands to arrive at a goal from the provided set
  4. Going back to step 2 unless no more goals are in the set

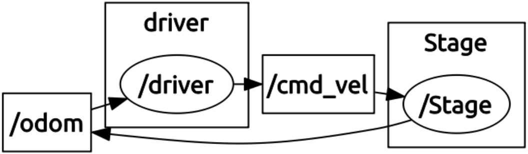


Figure: rqt graph for communicated messages between driver and stage simulator

Results and Demo

- Two selected cases from map 1

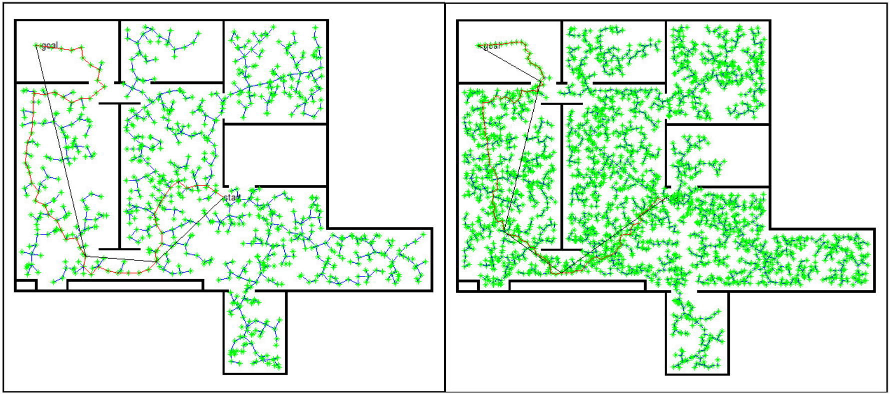


Figure: Left: k=10000,p=0.3,delta=10, *delta_q*=20 - Right: k=10000,p=0.3,delta=1, *delta_q*=10

- Two selected cases from map 2

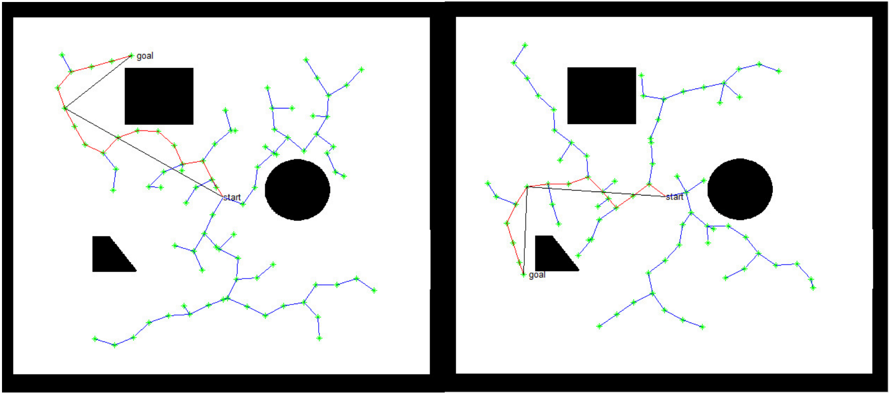
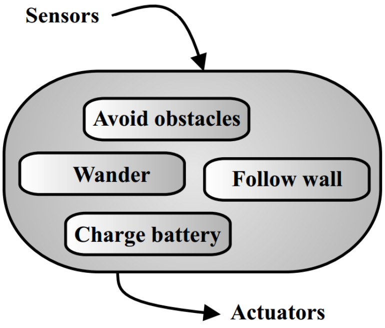


Figure: Left: k=5000, p=0.3, delta=1, *delta_q*=40 - Right: k=2000, p=0.3, delta=1, *delta_q*=40.

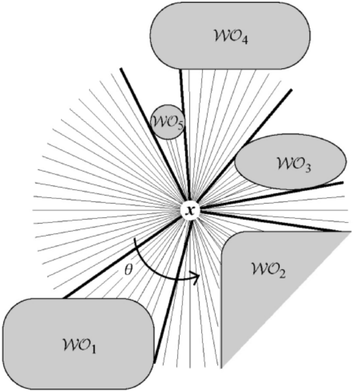
Behavioural Architecture based Controller

- Reactive
- No model plan
- Parallel Processing
- Basic feature: Obstacle Avoidance
- Real time



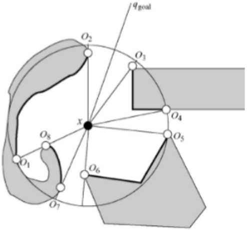
Tangent Bug

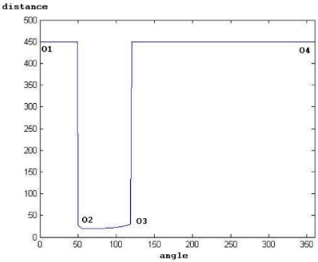
- Bug Algorithms
  1. Early and easy approaches to path planning
  2. Bug 1: safe and reliable
  3. Bug 2: better in some cases; worse in others
  4. Bug1 and Bug2: Tactile Sensing
- Tangent Bug
  1. Range sensors
  2. Proximity Sensing

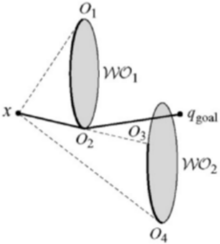


Tangent Bug

- Range Sensor : Array of distance
- The robot then moves toward the Oi that decreases a heuristic distance to the goal.
    1. choose O_i that minimizes: $d(x, O_i) + d(O_i, q_{goal})$

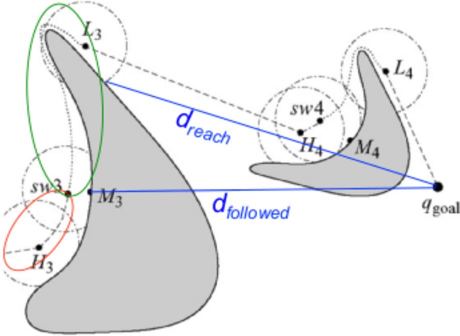






Tangent Bug Behaviours

- Motion to goal
- Boundary Following
  1. $d_{followed}$
  2. d_{reach}
  3. $d_{followed} > d_{reach}$

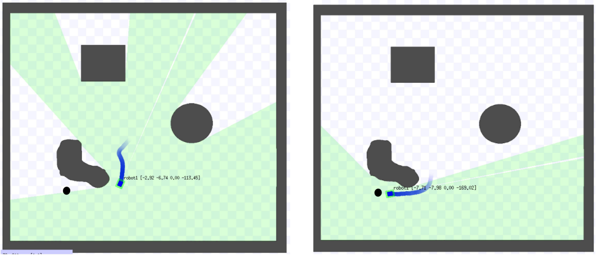


Tangent Bug Algorithm

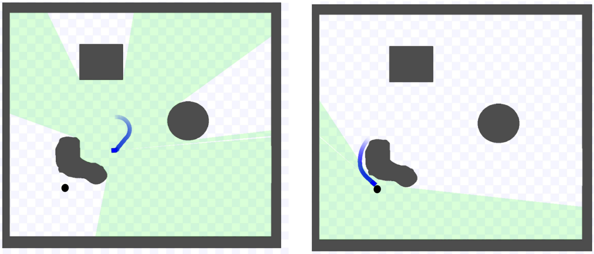
1. Move towards goal
2. Until you reach goal, otherwise switch to boundary following
3. Update d_{reach}, $d_{followed}$, O_i
4. Continuously move towards O_i that is in chosen boundary direction
5. Execute boundary following until goal is reached or terminate if robot completes one cycle around obstacle or
6. Terminate boundary following if $d_{reach} < d_{followed}$ and switch back to move towards goal until you reach the goal.

Results and Demo

Finite Sensor Range

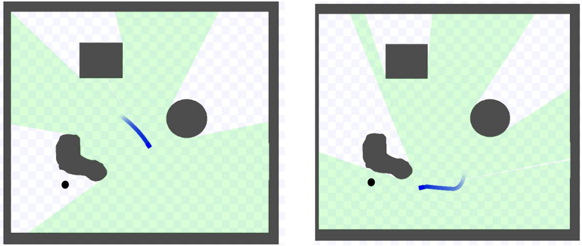


Zero Sensor Range



Results and Demo

High Sensor Range



Comparison

- Deliberative Control Architectures
  1. Sequential processing
  2. Suitable for structured or known or static environments
  3. RRT planner has predictable behaviour and is slow.
  4. Not real time
- Behavioural Control Architectures
  1. Real Time
  2. Suitable for changing and unstructured environments
  3. It is fast compared to RRT
  4. Tangentbug has a random behavior

Conclusion

- Implemented both deliberative and behivoural architectures
- Two different path planners, RRT and Tangent BUG
- Tested on different maps with different set of parameters
- A comparison has been drawn between both showing the pros and cons and emphasizing the importance of using a hybrid architecture

References

- N. Sariff, N. Buniyamin. An Overview of Autonomous Mobile Robot Path Planning Algorithms. 4^{th} Student Conference on Research and Development (Scored 2006), June 2006.
- M. LaValle. Rapidly-Exploring Random Trees: A new Tool for Path Planning.
- V.Alcazar, M.Veloso, D.Borrajo. Adapting a Rapidly-Exploring Random Tree for Automated Planning. The Fourth International Symposium on Combinatorial Search (SoCS-2011)
- Lecture Notes by Prof. Marc Carreras.
- Robot Operating System. `http://wiki.ros.org/ /~uno/abcde.html`
- Stage Simulator. `http://playerstage.sourceforge.net/index.php?src=stage /~uno/abcde.html`

Thank you