Obstacle Avoidance and Goal Detection Robot using RPi and LRF.

Atabak Hafeez Maria Ficiu Rubin Deliallisi Siddharth Shukla

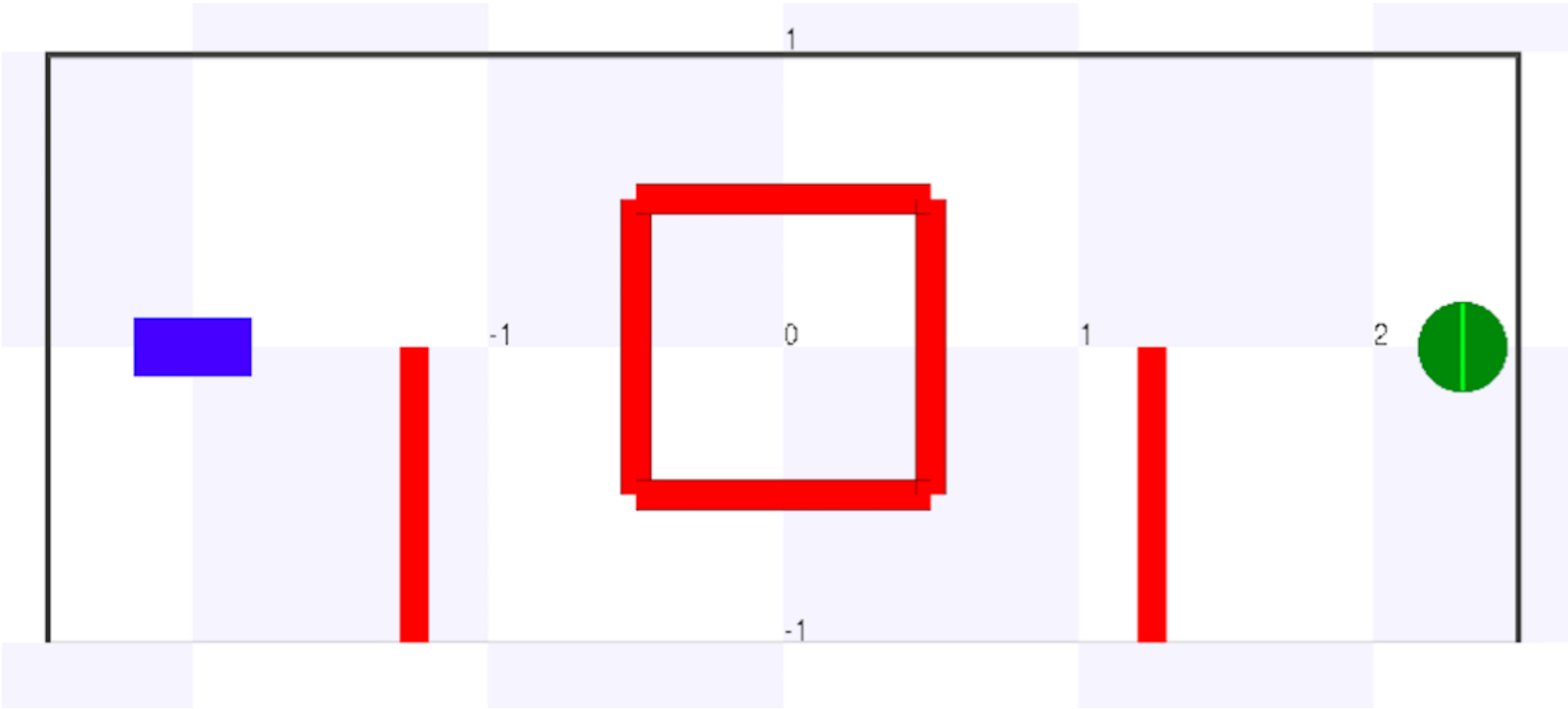
Jacobs University Bremen

May 10, 2016

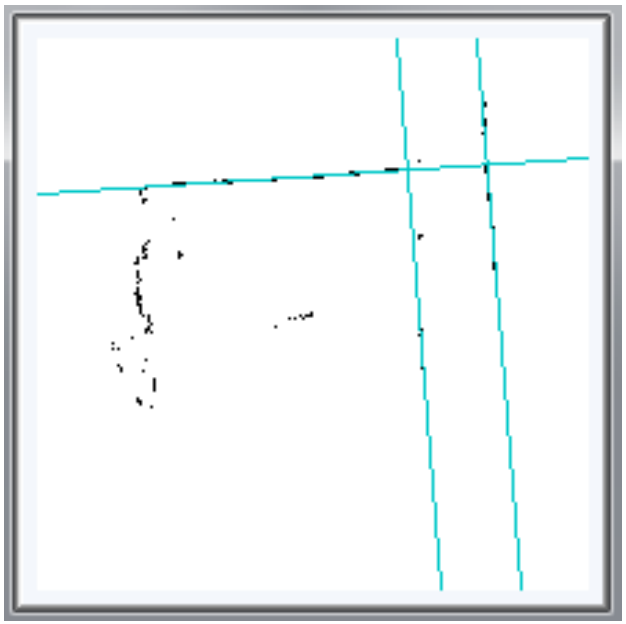
Overview

Goal

Develop a robot using ROS that can navigate a maze based on the input from a laser range finder.



Hough Transform



Detect imperfect instances of

- outer walls and obstacles, and
- (semi) circles (i.e. goals)
- OpenCV
  HoughLines and HoughCircle
- Careful
  tuning of params for detecting real cases and avoiding false positives

Wall Following

- Primary and Secondary security distances to avoid walls
- Wall Follow distance used to stay close to wall with in place turns
- Wall to be followed determined automatically
- Upon Circle Detection, hit if spatial limits allow else follow wall

Wall Following (Special cases)

- Robot gets reset if it can't see a wall (too far) after consecutive in place turns.
- Move towards the wall with a certain rotation in case the robot gets stuck in consecutive turn loop.
- Set a threshold for the difference of wall vs obstacle turns in case robot gets stuck in a maze loop.
- Reset the robot state if beyond threshold count.

Design Patterns

- Singleton (i.e. Only one instance of a given class)
- Publish-Subscribe (i.e. ROS nodes and topics)
- Mediator (Encapsulating communication between objects. Abandoned eventually in favour of Pub/Sub)

DevOps

- Git (using Github for remote)
- Divide tasks into issues
- Branching Model
  - Separate branch for each feature
  - Merge to develop after completion of issue
  - Rebase to develop at sprint end
- Pre commit hooks for code style guideline compliance (only linting)

Stay Agile! Stay Alive!

- Biweekly code sprints
- Weekly review and retrospective
- Coordinated Pair programming sessions

Testing

- Test Friendly Design Decisions
  - Requires ROS parameters to initialize
  - Parameter loading through config files
- Test Driven Development (TDD)
  - Wrote tests before code
  - Helped clearly plan out program functionality
  - Reduced debug time drastically
- Focused on different domains:
  - Unit Testing (Google Tests and ROS tests)
  - System Testing (ROS tests)
  - Stress Testing (ROS tests)

Simulation

- Great for rapid prototyping
- Helpful for TDD (automated test suites)
- Easy to benchmark

Simulation (goal)

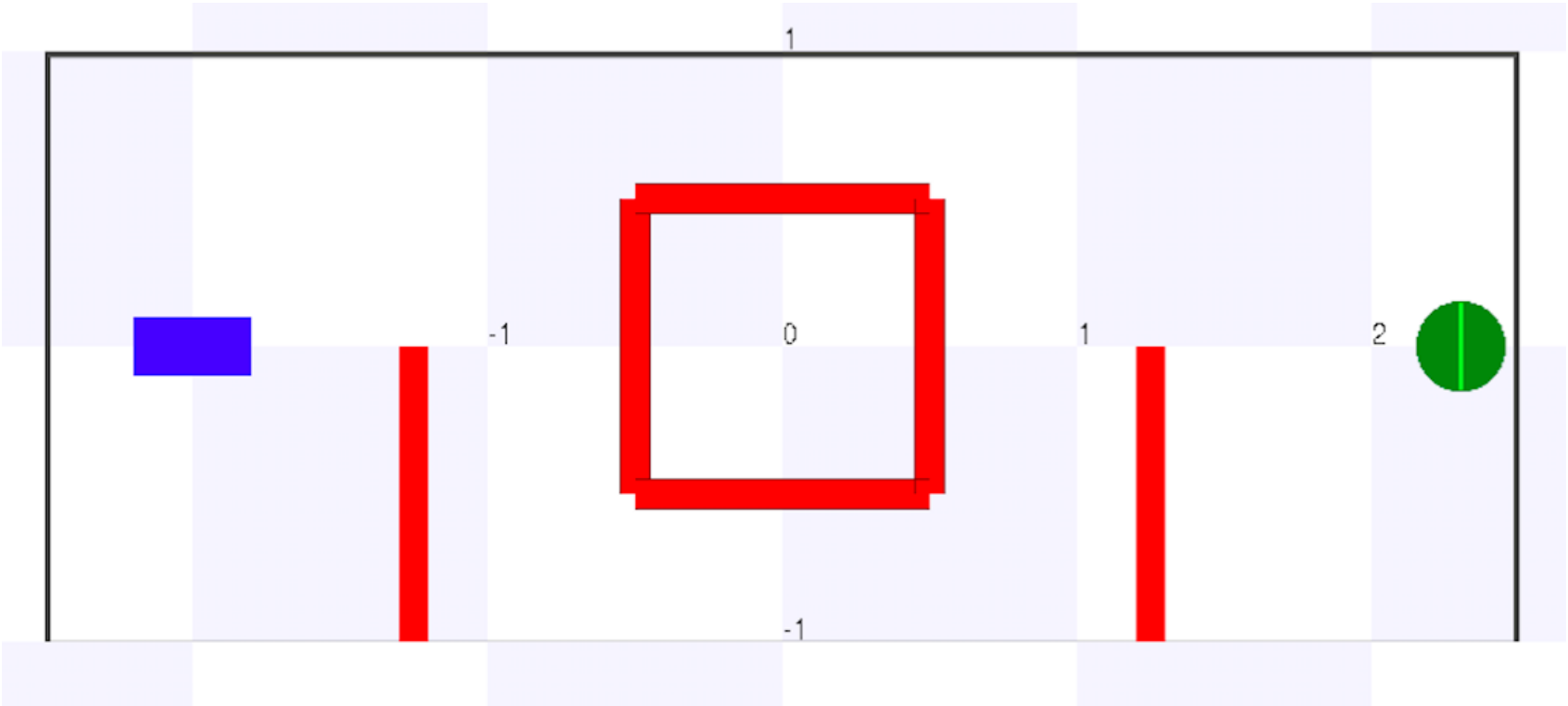


Figure: Simulation on different levels

Simulation (loop detection)

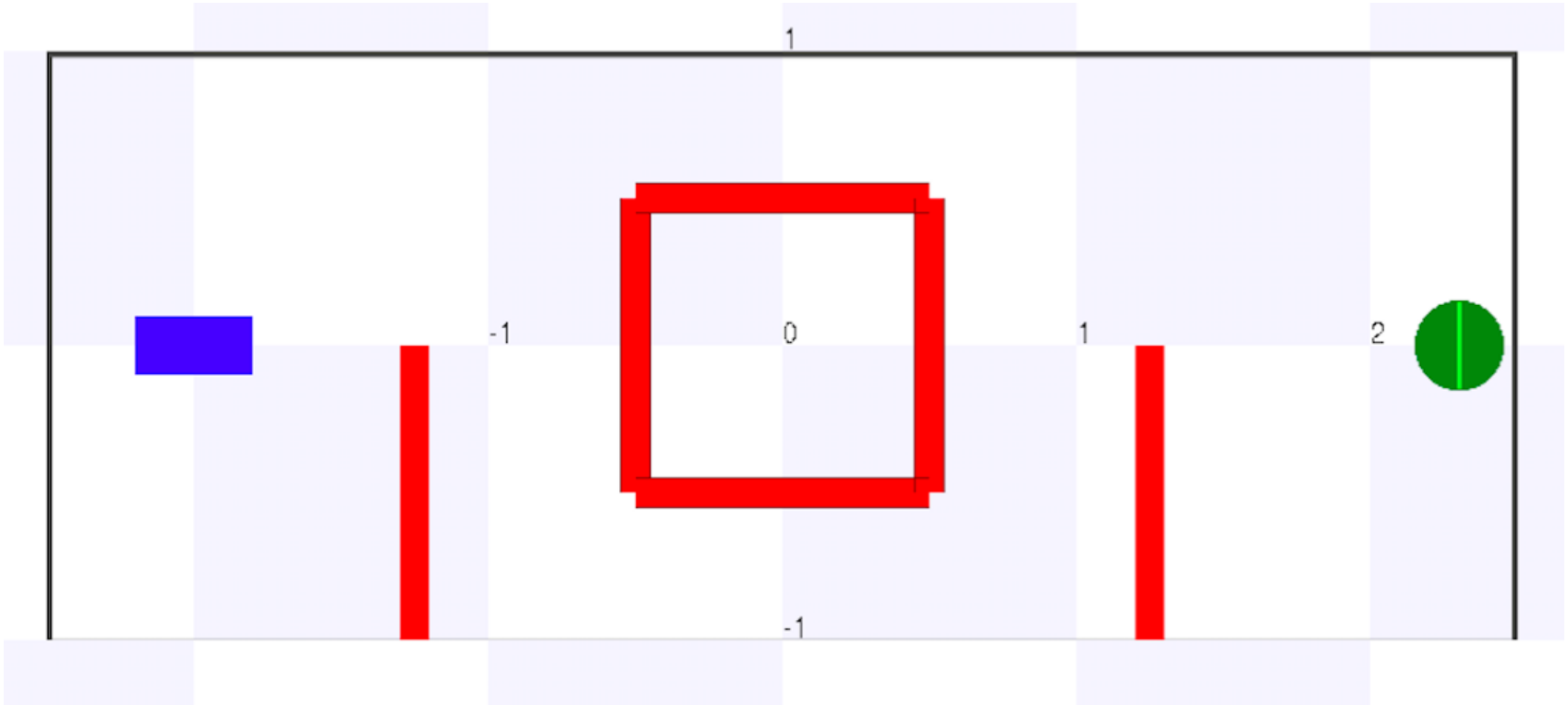


Figure: Simulation on different levels

Real World

- Unsuspecting differences between Simulation and Real World
- Everything more prone to errors and Data more noisy
- Increased Linear and Angular velocities for Real world
- Decreased wall follow distance because of data delay
- Adjust circle hit mode to do adaptive refinement of movement direction

References

Atabak Hafeez, Maria Ficiu, Rubin Deliallisi, Siddharth Shukla (2016)

Obstacle Avoidance and Goal Detection Robot

Jacobs University Bremen.

Michael Misha Kazhdan(2004)

Seminar on Shape Analysis and Retrieval

Department of Computer Science, Johns Hopkins University, Baltimore, MD.

Howie Choset

Robotic Motion Planning: Bug Algorithms

The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.