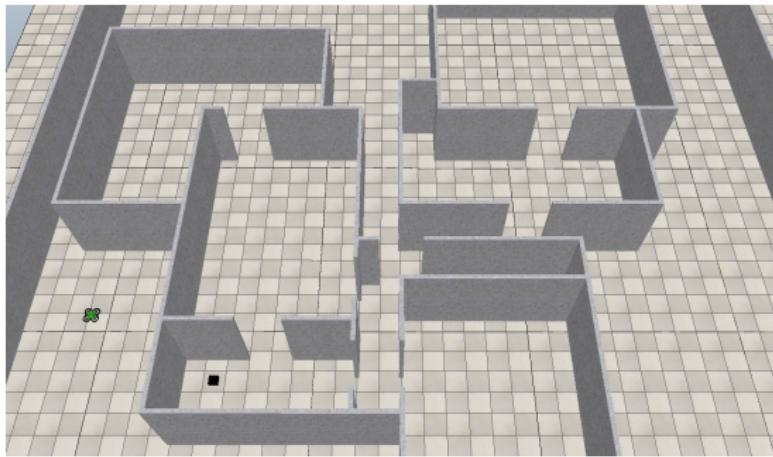


Maze Solver Quadrotor

Diogo Alexandre Martins



29 de outubro de 2019

Summary

- 1 Introduction
- 2 Maze Solving Algorithm
- 3 Mapping the Maze
- 4 Control Tuning and Movement Restriction
- 5 Image Recognition
- 6 Final Strategy
- 7 Results

Summary

1 Introduction

2 Maze Solving Algorithm

3 Mapping the Maze

4 Control Tuning and Movement Restriction

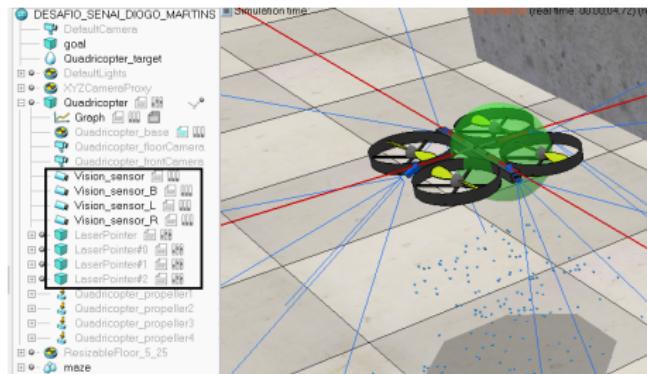
5 Image Recognition

6 Final Strategy

7 Results

Using Columns

- Find a black box
- Avoid walls
- Laser sensor
- Vision sensor

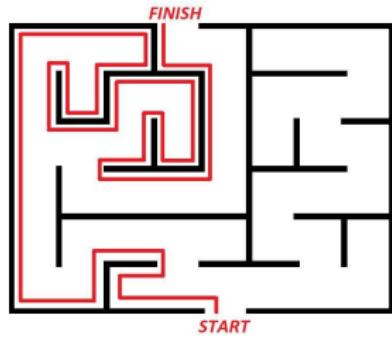


Summary

- 1 Introduction
- 2 Maze Solving Algorithm
- 3 Mapping the Maze
- 4 Control Tuning and Movement Restriction
- 5 Image Recognition
- 6 Final Strategy
- 7 Results

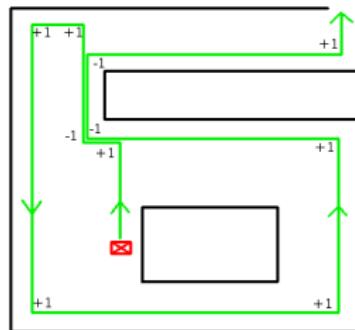
Maze Solving Algorithm: Wall Follower

- Left-hand rule or the right-hand rule
- Guaranteed solution for simply connected mazes



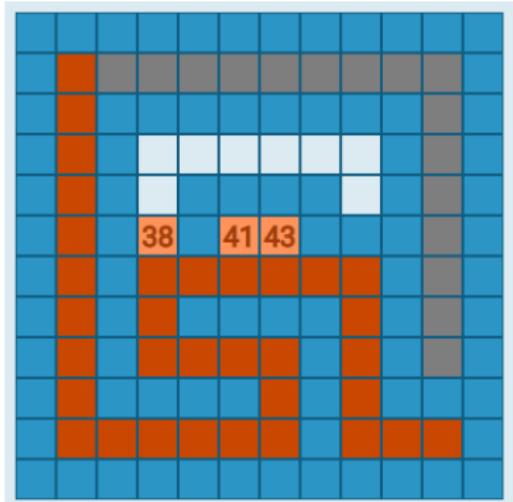
Maze Solving Algorithm: Pledge Algorithm

- Similar to Wall Follower
- Angles turned are counted
- Guaranteed solution for disjoint mazes
- Does not work if the maze is surrounded by walls



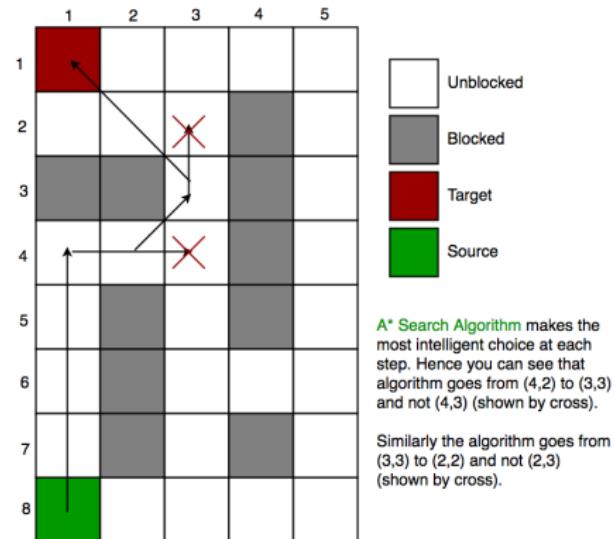
Maze Solving Algorithm: Depth-first Search

- Explores as far as possible along each direction
- Stores the explored path for backtracking
- Not optimal or guaranteed to find the best solution



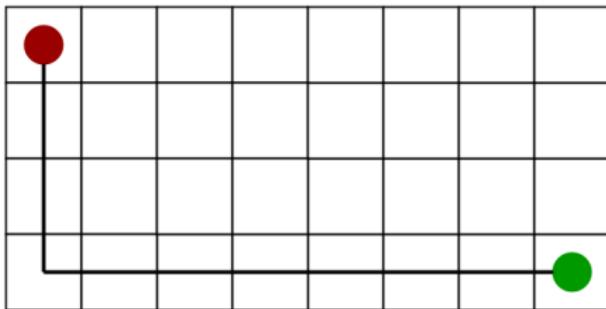
Maze Solving Algorithm: A* Search Algorithm

- Popular technique used in path-finding
- Requires a destination
- Score = $g + h$
- g = movement cost to move from source to a given square
- h = estimated movement cost to move from that given square to the final destination
(Manhattan Distance)



Maze Solving Algorithm: A* Search Algorithm

- Popular technique used in path-finding
- Requires a destination
- Score = $g + h$
- g = movement cost to move from source to a given square
- h = estimated movement cost to move from that given square to the final destination
(Manhattan Distance)

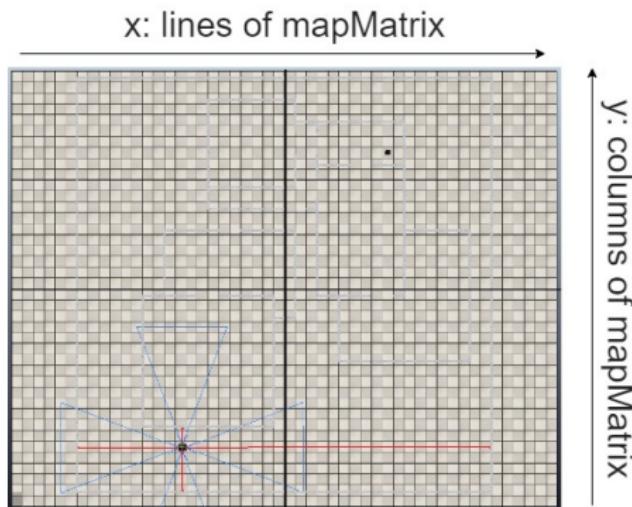


Summary

- 1 Introduction
- 2 Maze Solving Algorithm
- 3 Mapping the Maze
- 4 Control Tuning and Movement Restriction
- 5 Image Recognition
- 6 Final Strategy
- 7 Results

Mapping the Maze

- Discretize the maze
- Every region represents a matrix position
- Requirement for applying maze solving algorithms



Mapping the Maze

Region status

- Unknown: 9
- Visited: 2
- Visiting: 1
- Obstacle: 3

mapMatrix

9	9	9	0	0	9
3	9	9	0	0	9
3	3	0	1	0	0
3	3	0	2	0	0
3	3	0	2	2	0
3	9	9	3	2	0

Mapping the Maze

A diagram showing a quadcopter at an angle to a vertical wall. A red line segment of length d extends from the bottom left towards the wall. A horizontal dotted line represents the wall. The projection of the red line onto the wall is labeled $d \cdot \cos(\beta)$. The angle between the red line and the horizontal is labeled β .

$$d \cdot \cos(\beta)$$

PITCH/ROLL CORRECTION

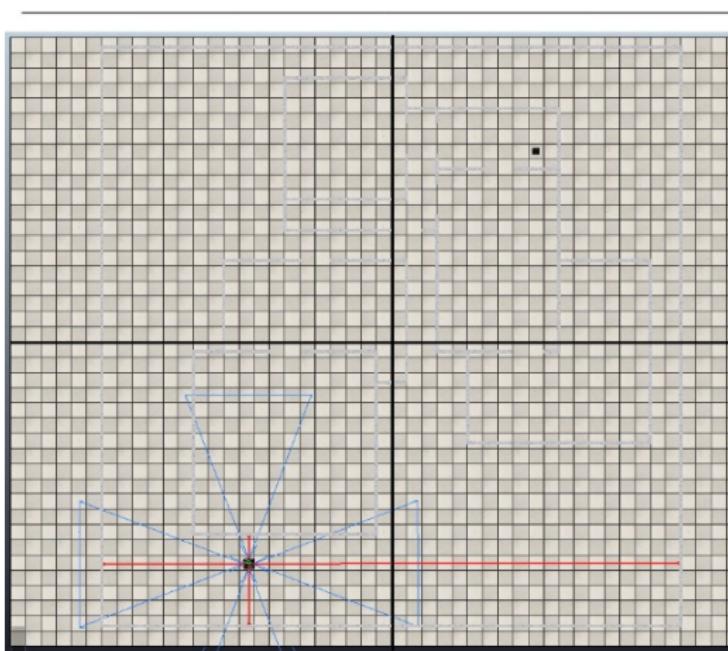
A diagram showing a quadcopter at an angle to a vertical wall. A red line segment of length D extends from the bottom right towards the wall. A horizontal dotted line represents the wall. The projection of the red line onto the wall is labeled $(D \cdot \cos(\Theta) ; D \cdot \sin(\Theta))$. The angle between the red line and the horizontal is labeled Θ . Below the diagram, the formula $D = d \cdot \cos(\beta)$ is shown.

$$(D \cdot \cos(\Theta) ; D \cdot \sin(\Theta))$$
$$D = d \cdot \cos(\beta)$$

YAW CORRECTION

Mapping the Maze

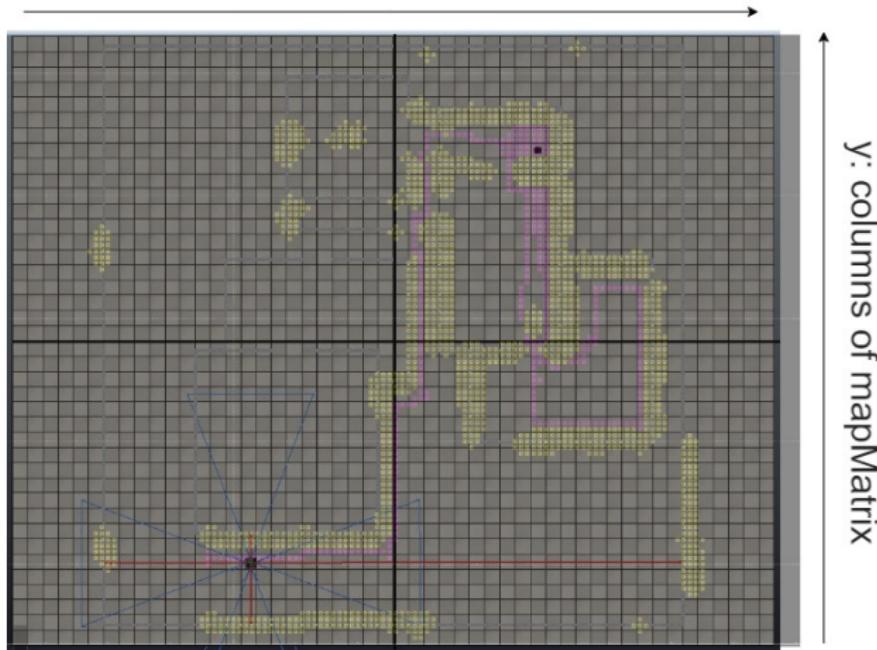
x: lines of mapMatrix



y: columns of mapMatrix

Mapping the Maze

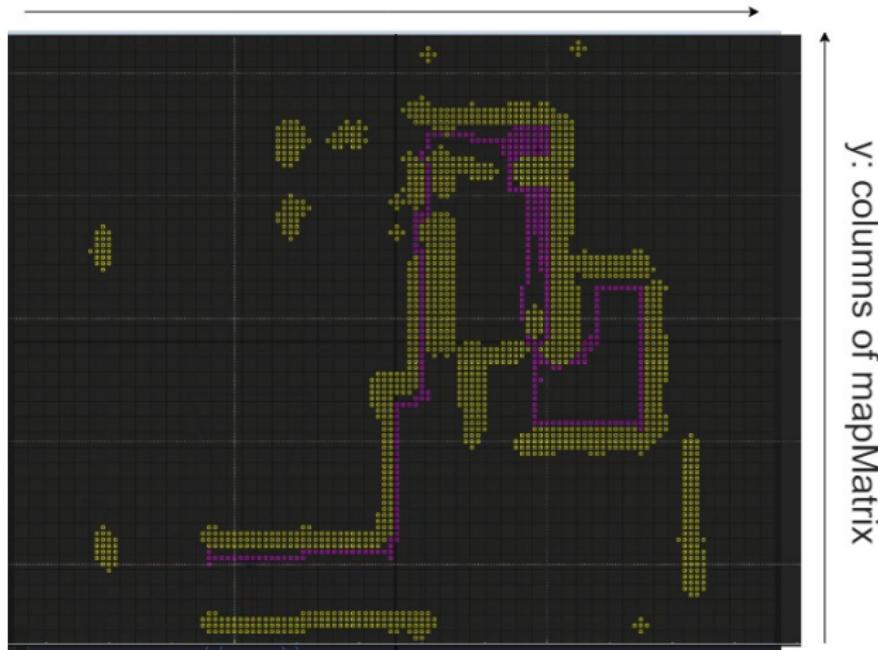
x: lines of mapMatrix



y: columns of mapMatrix

Mapping the Maze

x: lines of mapMatrix



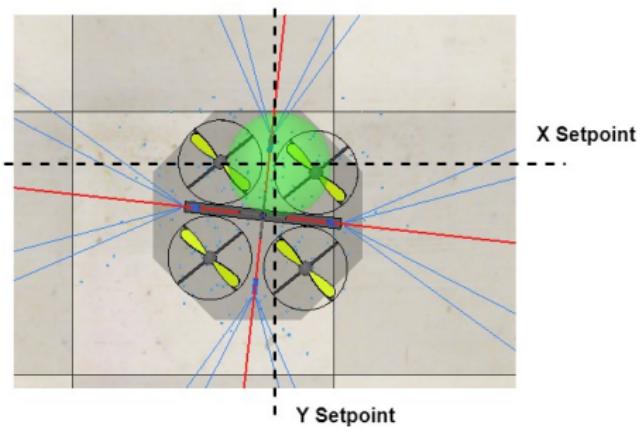
Summary

- 1 Introduction
- 2 Maze Solving Algorithm
- 3 Mapping the Maze
- 4 Control Tuning and Movement Restriction
- 5 Image Recognition
- 6 Final Strategy
- 7 Results

Control Tuning and Movement Restriction

Horizontal Control and Yaw Control

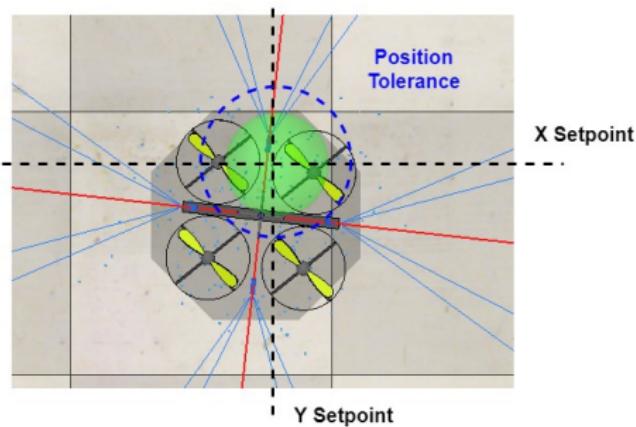
- Added integral action to eliminate offset
- Increased derivative action to reduce overshoot



Control Tuning and Movement Restriction

Movement Restriction

- Position tolerance
- Velocity tolerance
- Reset integral action within the tolerance region



Summary

- 1 Introduction
- 2 Maze Solving Algorithm
- 3 Mapping the Maze
- 4 Control Tuning and Movement Restriction
- 5 Image Recognition
- 6 Final Strategy
- 7 Results

Image Recognition

Used filters

- "Binary work image and trigger"
- "Edge detection on work image"
- "Blob detection on work image"



Image Recognition

Used filters

- "**Binary work image and trigger**"
- "Edge detection on work image"
- "Blob detection on work image"

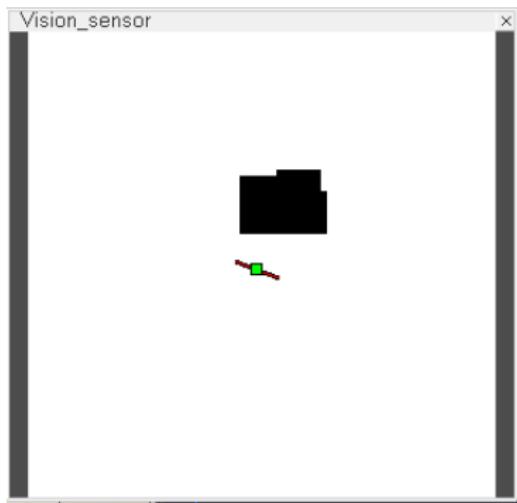


Image Recognition

Used filters

- "Binary work image and trigger"
- "**Edge detection on work image**"
- "Blob detection on work image"

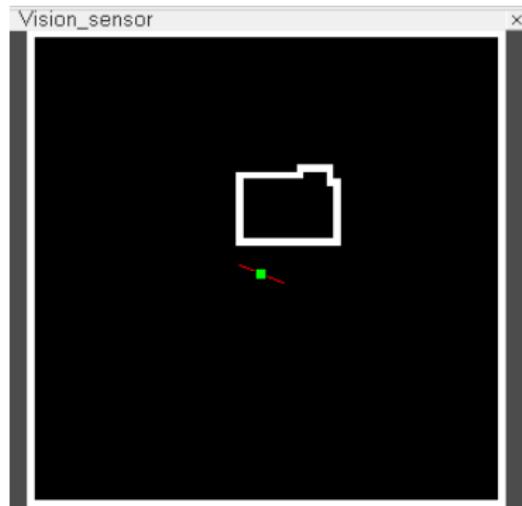
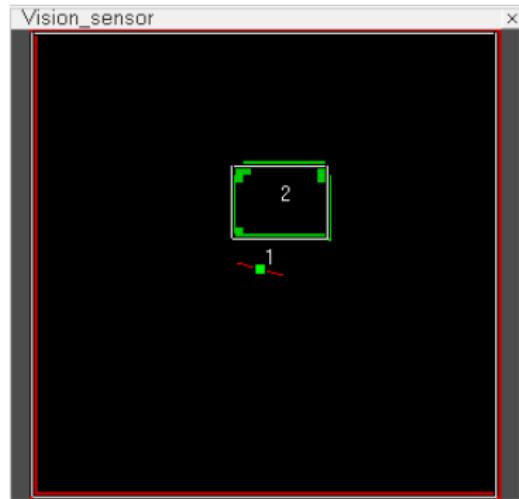


Image Recognition

Used filters

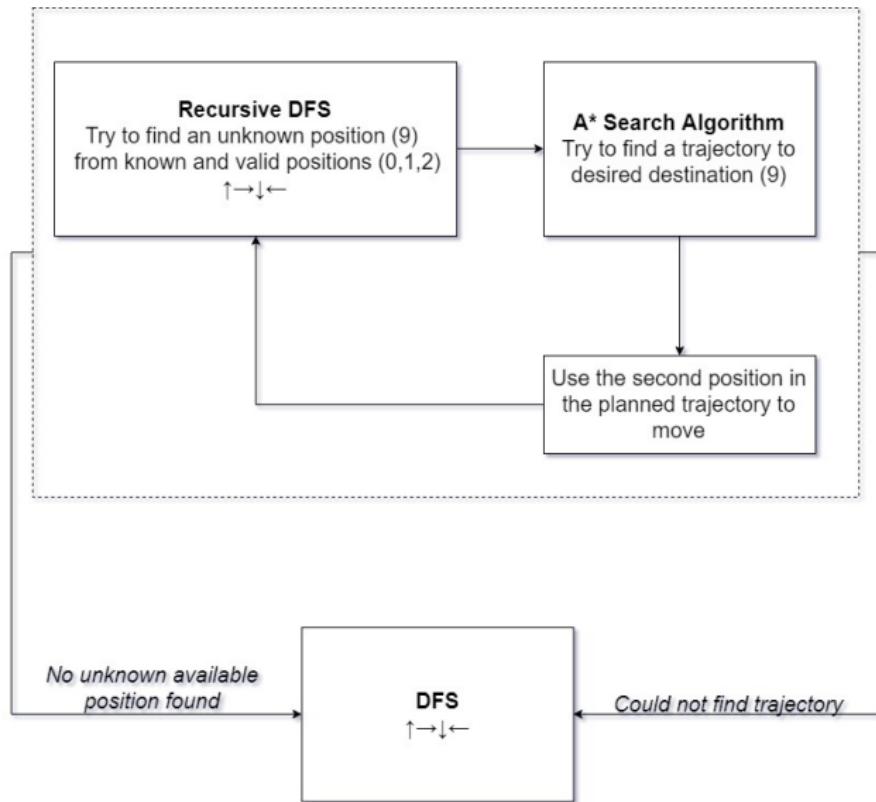
- "Binary work image and trigger"
- "Edge detection on work image"
- "**Blob detection on work image**"



Summary

- 1 Introduction
- 2 Maze Solving Algorithm
- 3 Mapping the Maze
- 4 Control Tuning and Movement Restriction
- 5 Image Recognition
- 6 Final Strategy
- 7 Results

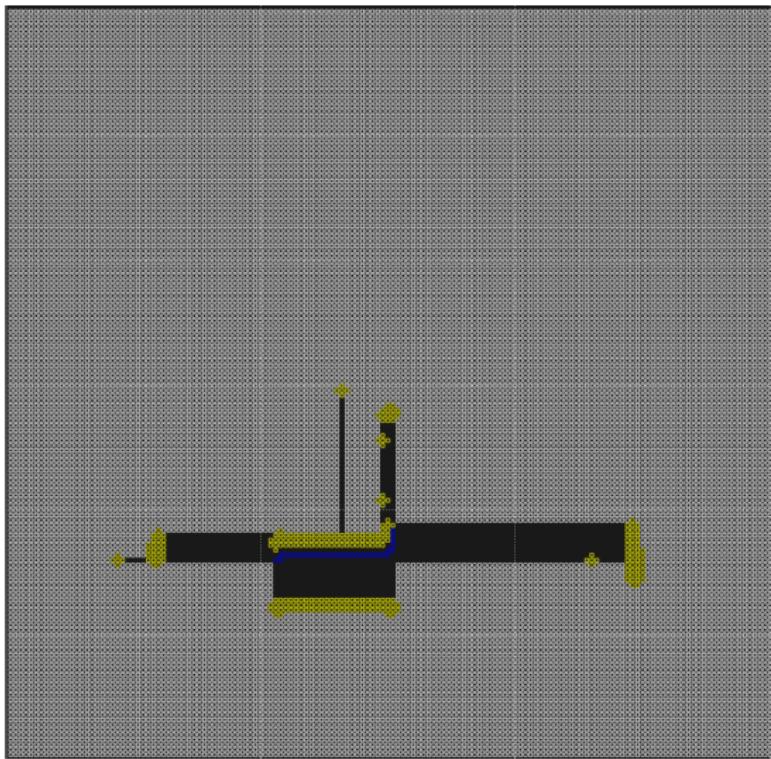
Final Strategy



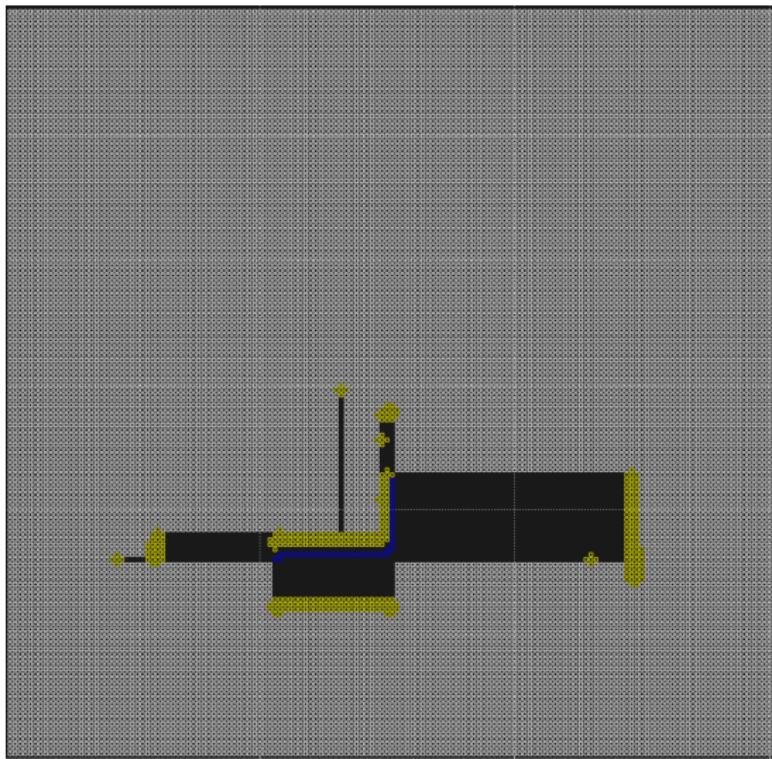
Summary

- 1 Introduction
- 2 Maze Solving Algorithm
- 3 Mapping the Maze
- 4 Control Tuning and Movement Restriction
- 5 Image Recognition
- 6 Final Strategy
- 7 Results

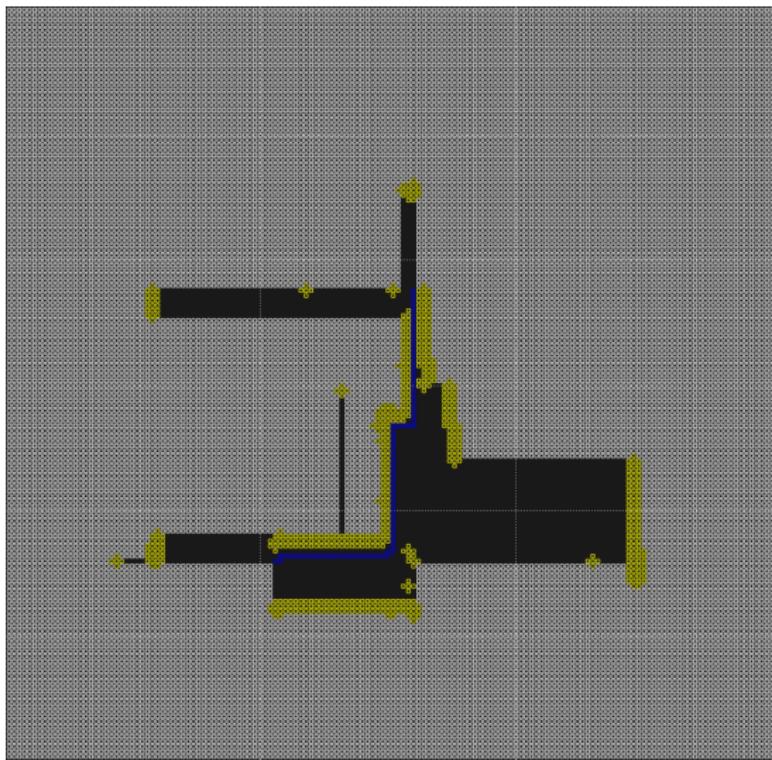
Results



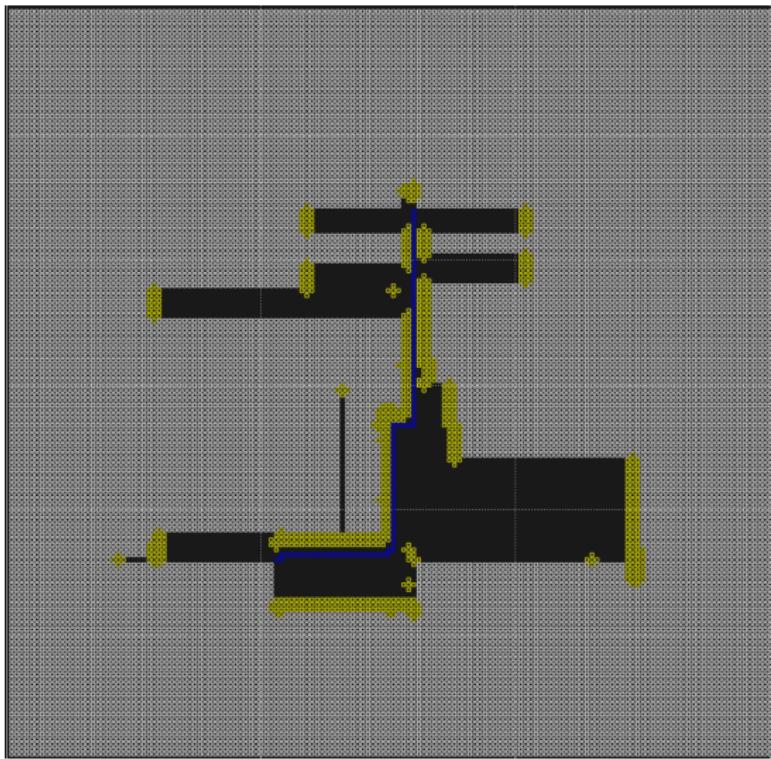
Results



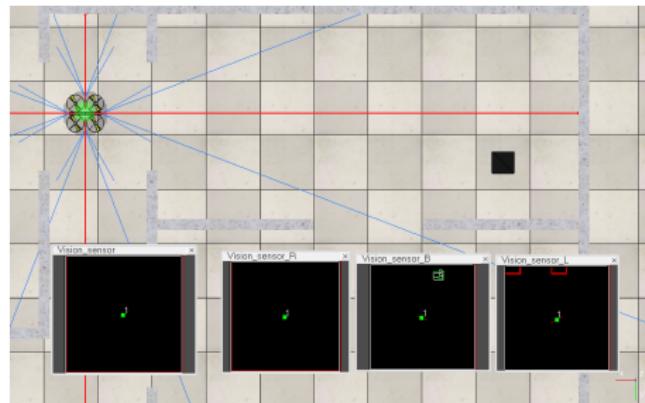
Results



Results



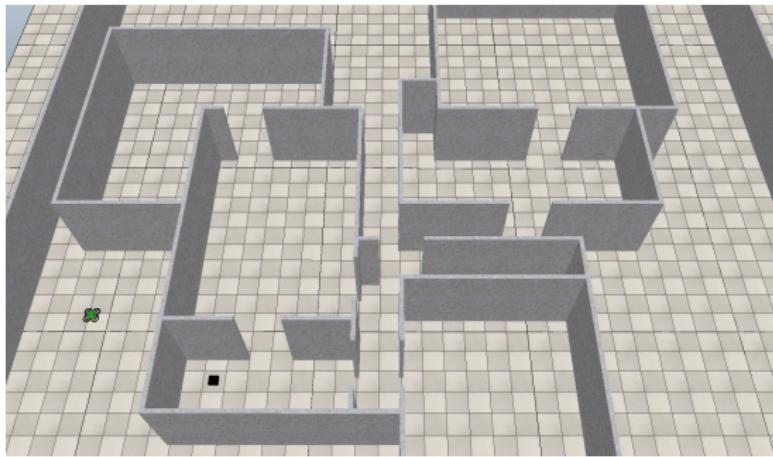
Results



```
Back sensor: Goal on sight  
Back sensor: Goal on sight  
Step 1 >> 80, 109  
Step 2 >> 81, 109  
Step 3 >> 81, 110  
  
Back sensor: Goal on sight  
  
Back sensor: Goal on sight
```

Maze Solver Quadrotor

Diogo Alexandre Martins



29 de outubro de 2019