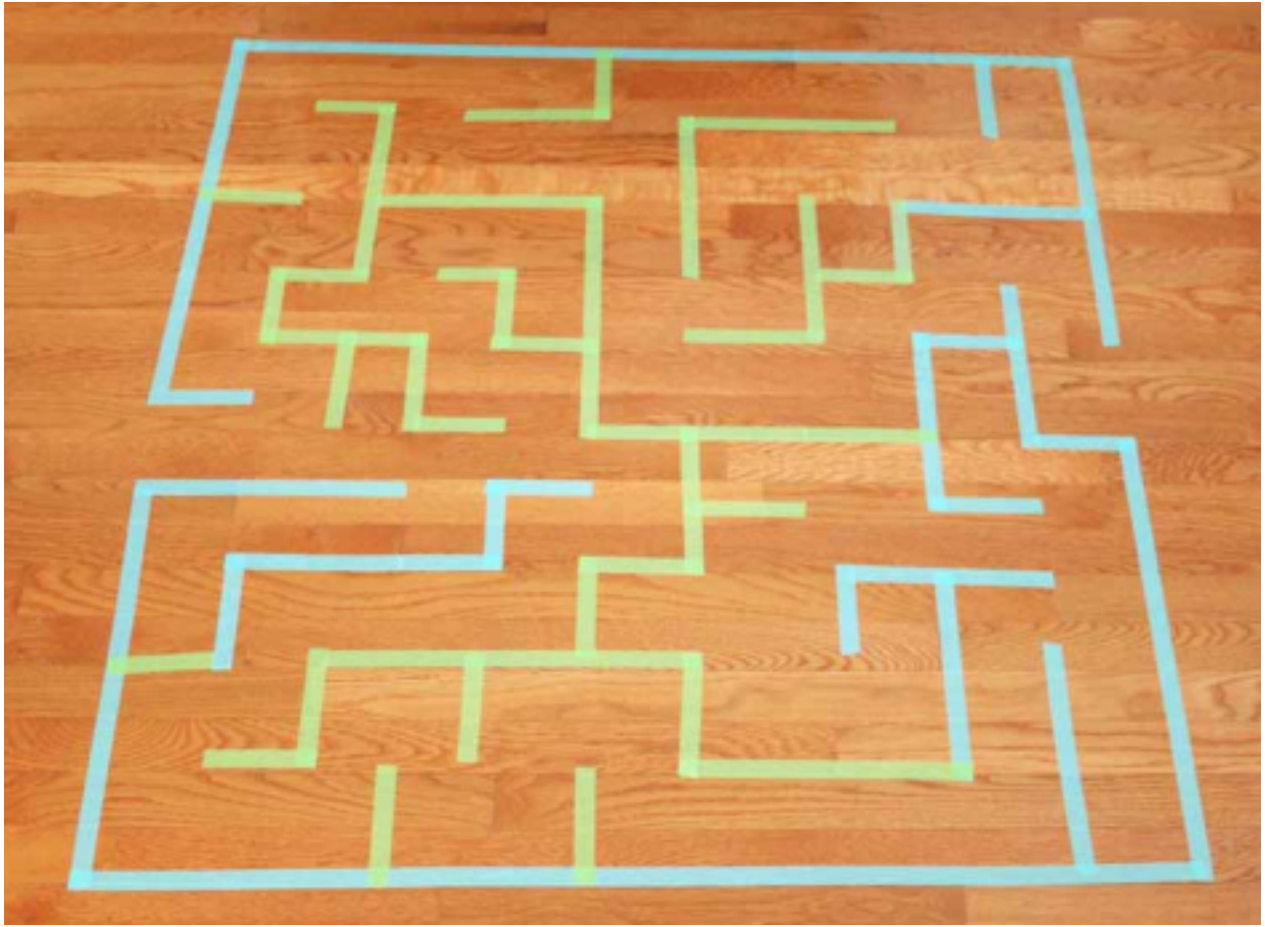
Gim’s Labyrinth

Gian Delfin, Vivian Huang, Luis Terrones-Verastegui

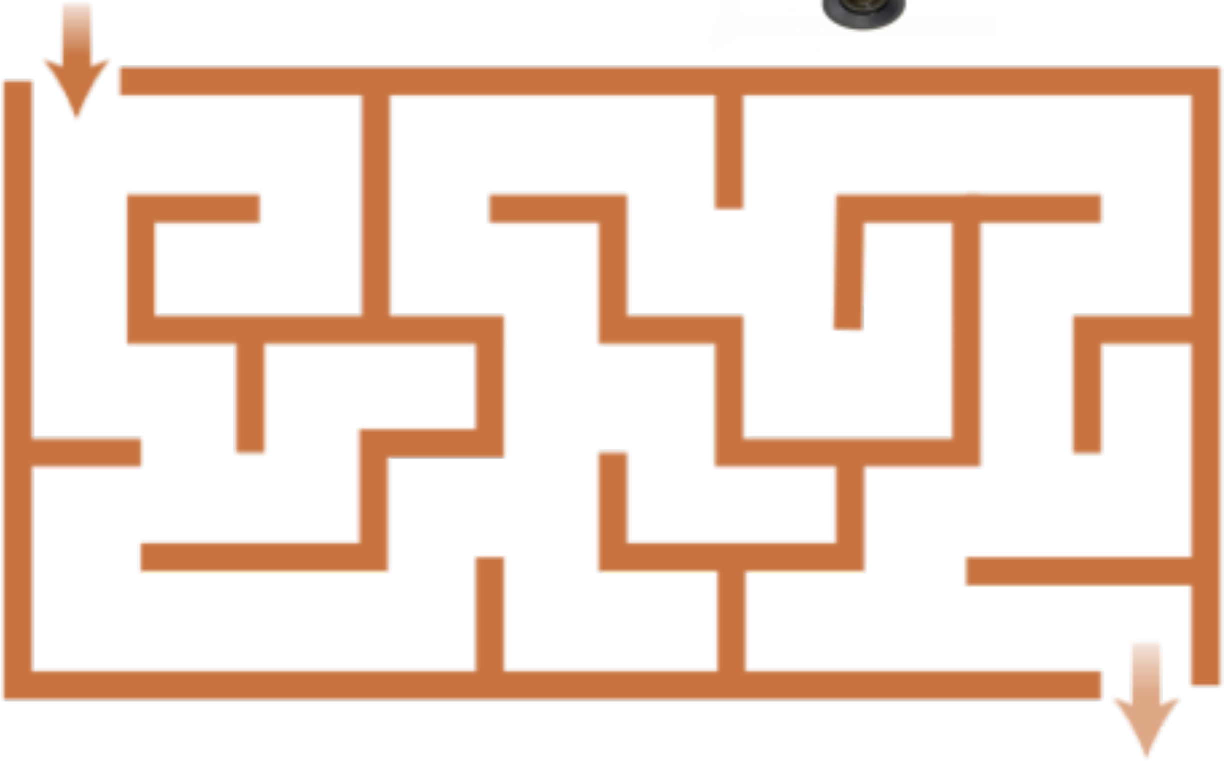
Overview

- Maze Setup



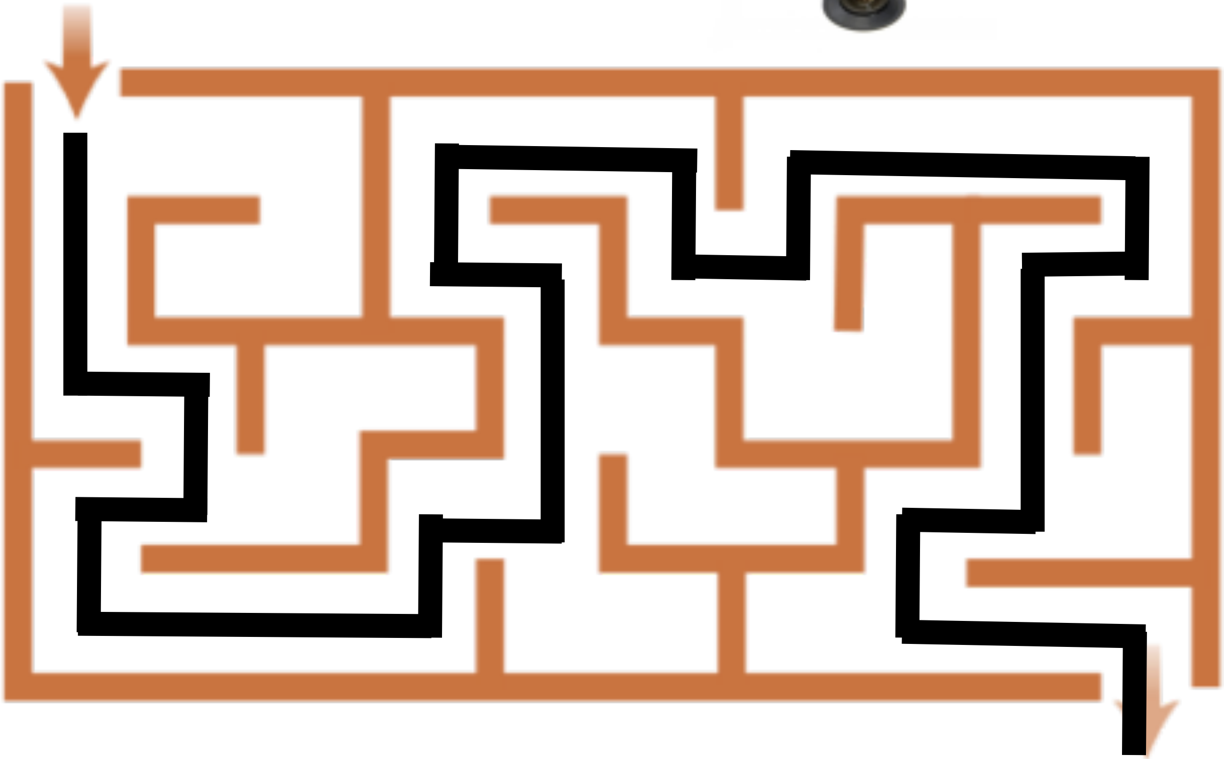
Overview

- Maze Setup
- Image Processing



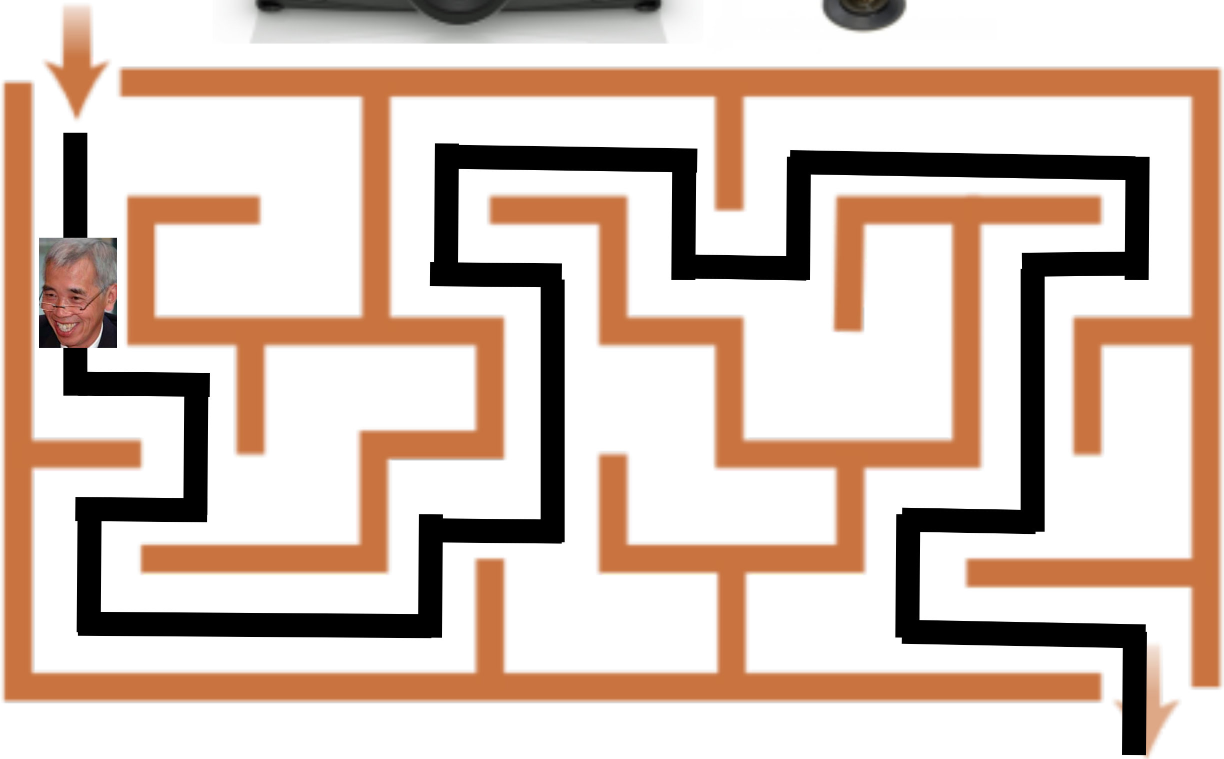
Overview

- Maze Setup
- Image Processing
- Path Solving



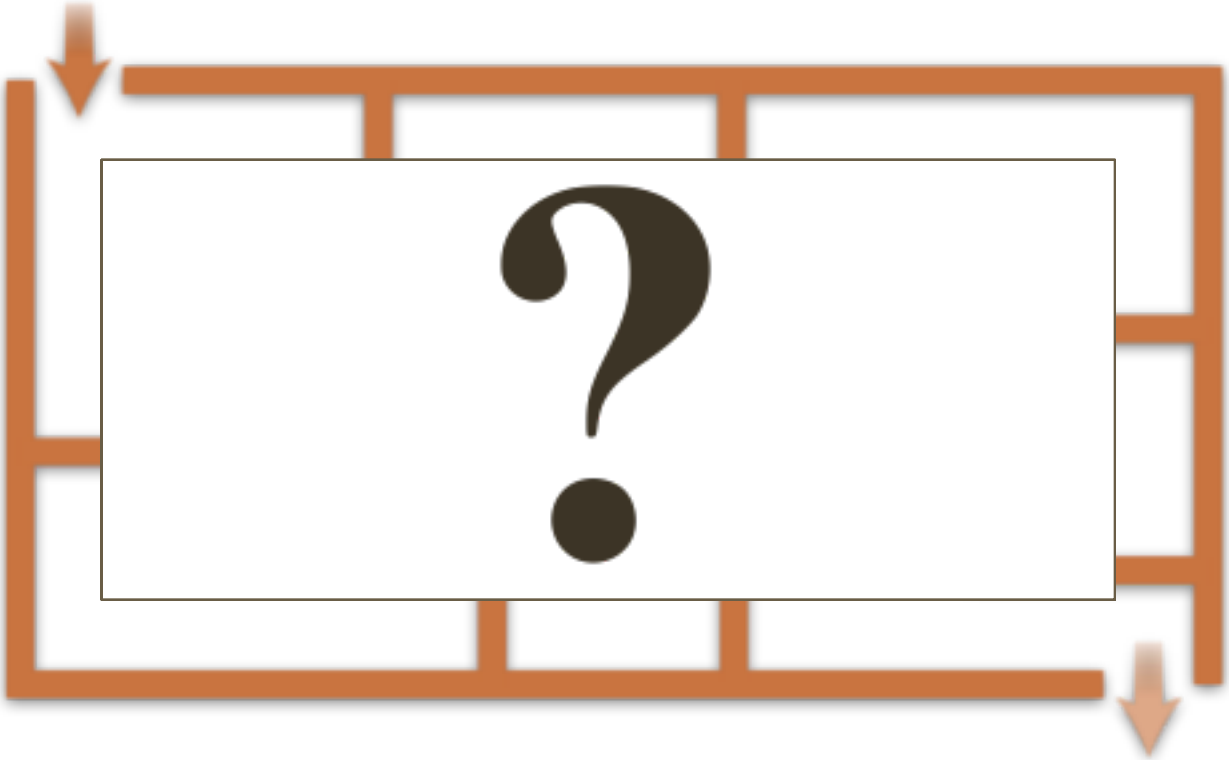
Overview

- Maze Setup
- Image Processing
- Path Solving
- Projection

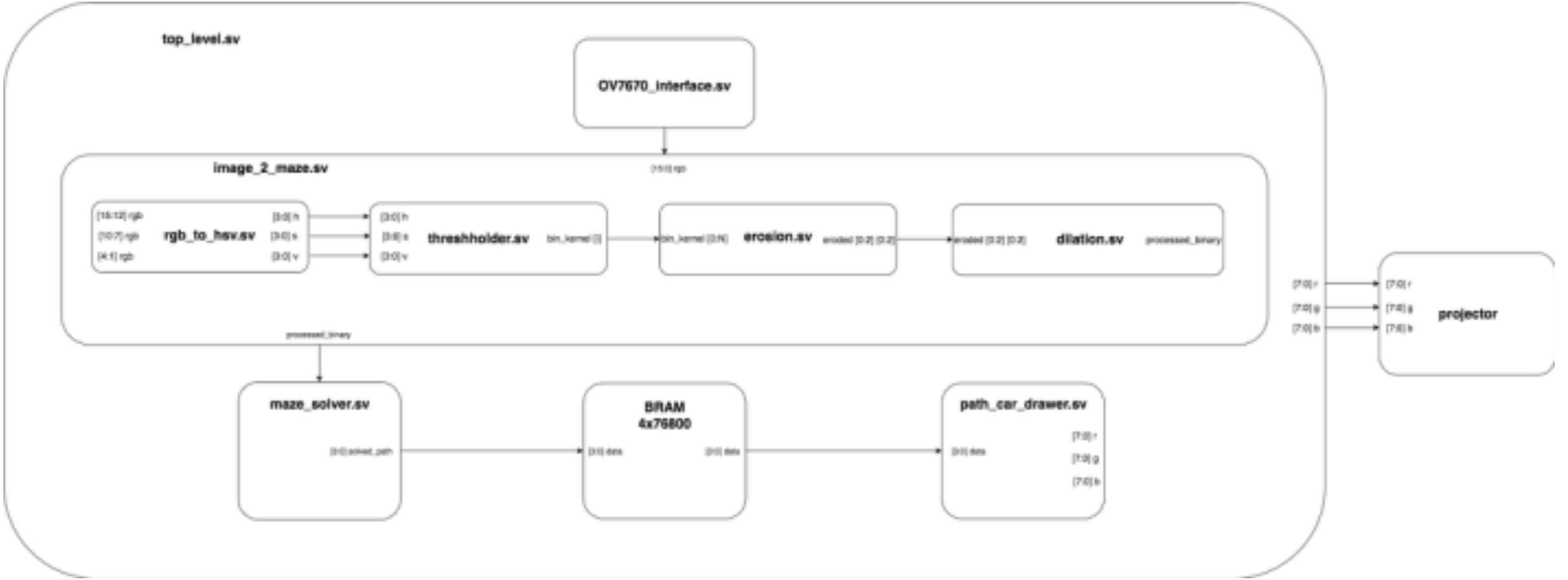


Overview

- Maze Setup
- Image Processing
- Path Solving
- Projection
- Stretch Goal: Real-Time Maze Manipulation



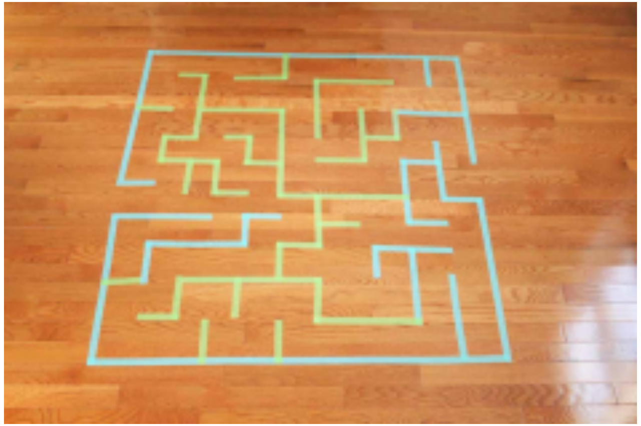
Block Diagram

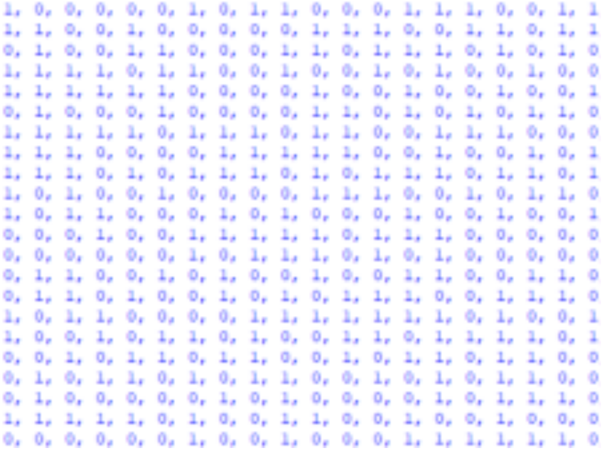


Modules

1. Image Processing

- Interface with OV7670
- Convert 16 bit RGB image to 2D binary array



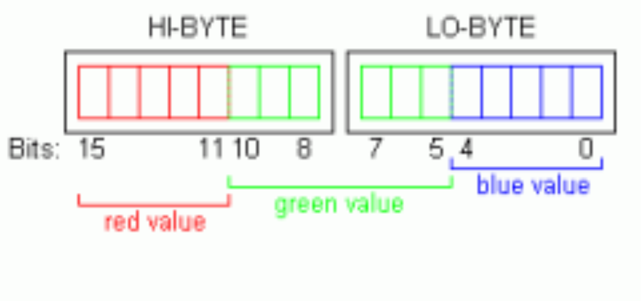


```
1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1
1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1
0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0
1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1
0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0
1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0
1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1
1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1
1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0
1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1
0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0
0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0
1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1
1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1
0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0
0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0
0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0
1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0
```

- Process, then pass binary array to maze solving algorithm

RGB to HSV

- 16 bit RGB pixels
- Sample twice from OV7670 to obtain one pixel



$$C_{\max} = \max(R, G, B)$$
$$C_{\min} = \min(R, G, B)$$
$$\Delta = C_{\max} - C_{\min}$$

$$H = \begin{cases} 0 & \text{if } C_{\max} = 0 \\ \left(60 \times \frac{G-B}{\Delta} + 360\right) \bmod 360 & \text{if } R = C_{\max} \\ 60 \times \frac{B-R}{\Delta} + 120 & \text{if } G = C_{\max} \\ 60 \times \frac{R-G}{\Delta} + 240 & \text{if } B = C_{\max} \end{cases}$$

$$S = \frac{\Delta}{C_{\max}}$$

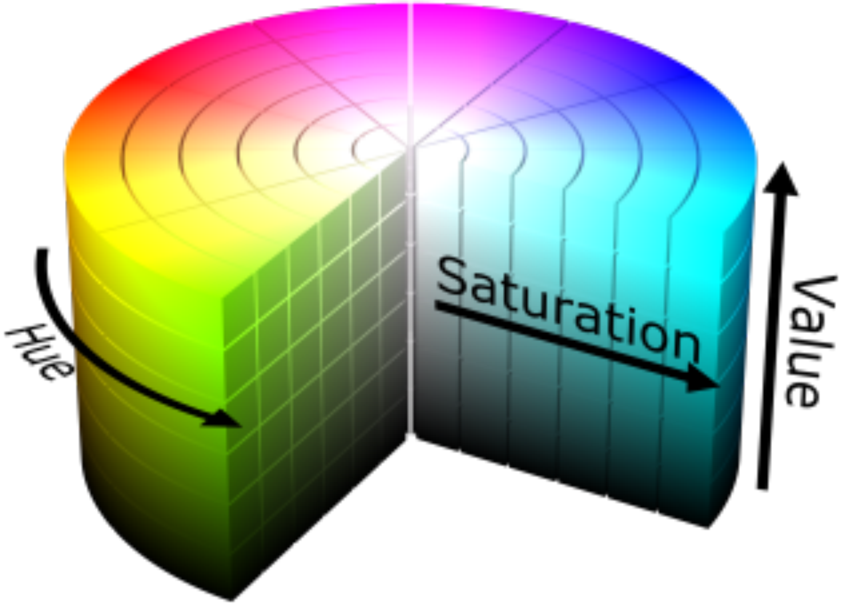
$$V = \frac{C_{\max}}{255}$$

Threshold

- HSV easier to threshold
- Slice cylinder to get wall colors

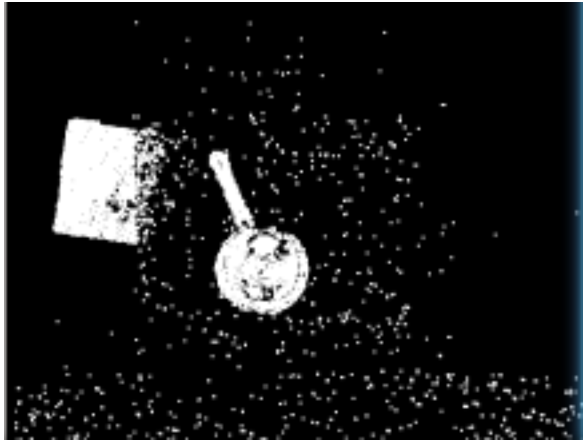
Wall = 0

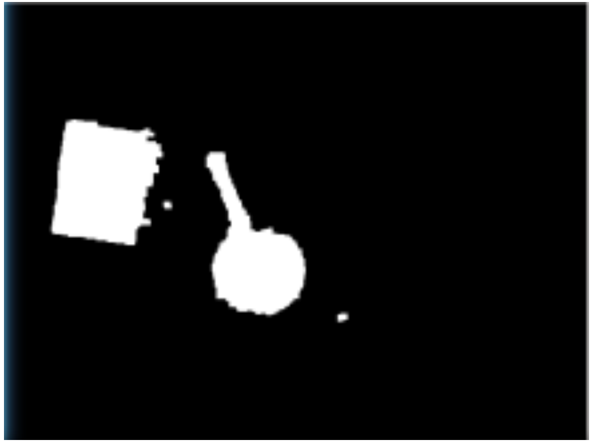
No Wall = 1



Binary Image Smoothing

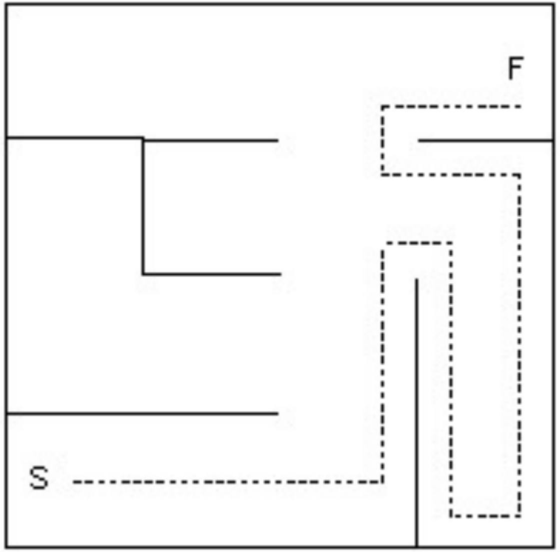
- Necessary to smooth/denoise binary image
- Erosion/Dilation
- Median Filter
- Graph Cuts



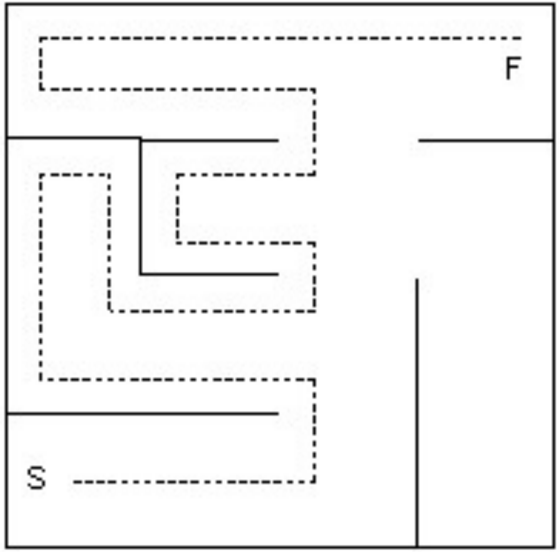


2a. Maze Solving: Wall Following Algorithm

- Guaranteed not to get lost
- No solution? Returns to entrance
- Stuck if start at isolated segment



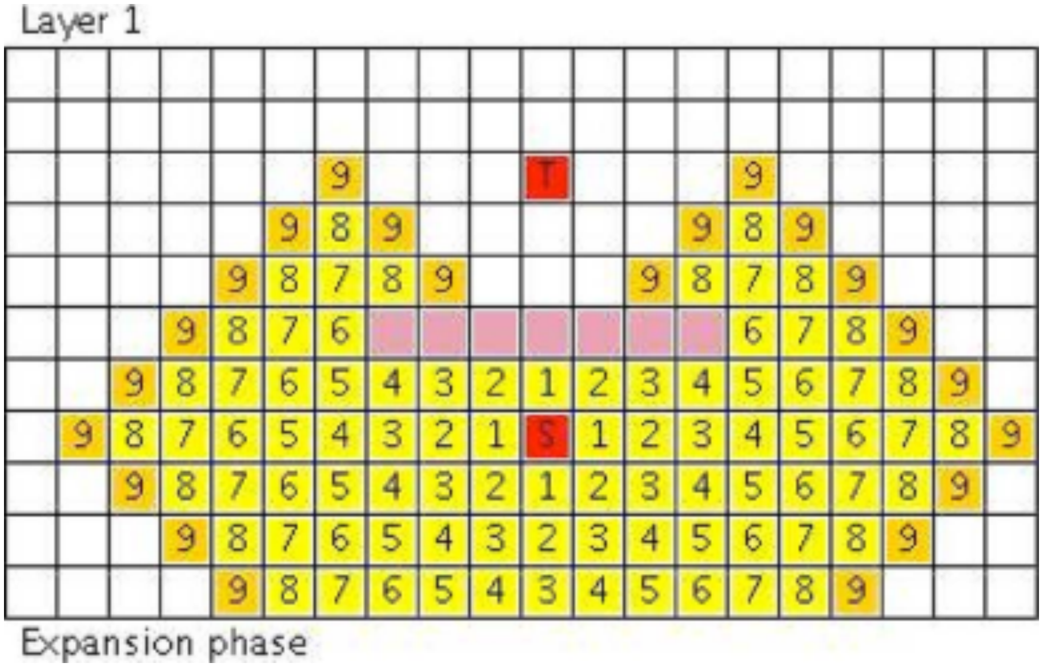
Right Wall Follow



Left Wall Follow

2b. Maze Solving Algorithm: Lee's Algorithm

- BFS exploration of maze
- Expand one move at a time
- Guaranteed shortest path



Layer 1

						9				T				9					
					9	8	9						9	8	9				
				9	8	7	8	9				9	8	7	8	9			
			9	8	7	6	▨	▨	▨	▨	▨	▨	▨	6	7	8	9		
		9	8	7	6	5	4	3	2	1	2	3	4	5	6	7	8	9	
	9	8	7	6	5	4	3	2	1	S	1	2	3	4	5	6	7	8	9
		9	8	7	6	5	4	3	2	1	2	3	4	5	6	7	8	9	
			9	8	7	6	5	4	3	2	3	4	5	6	7	8	9		
				9	8	7	6	5	4	3	4	5	6	7	8	9			

Expansion phase

3. Path Projection

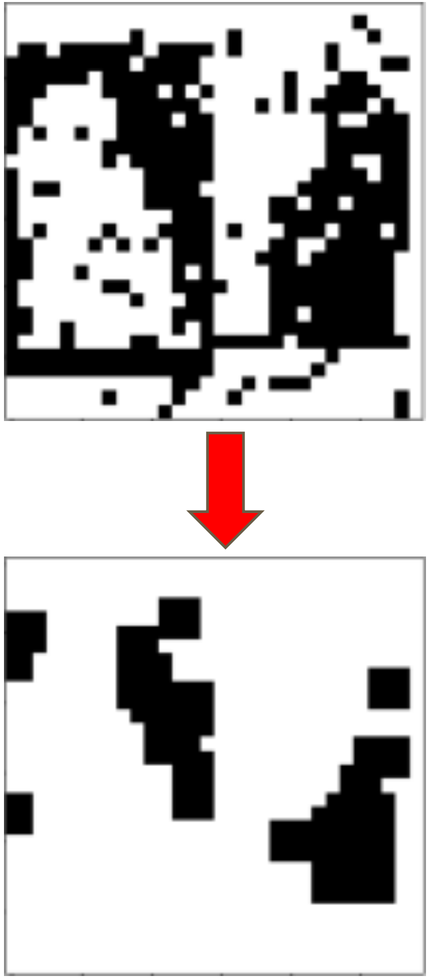
- Represent path as deltax & deltay values
- Write path found to BRAM
- Draw path by following these deltas from start to finish in a cycle

BRAM 4x76800
4’b0001
4’b0100
. . .
4’b1001

Possible Issues

- Memory
  - Image resolution : 320 x 240 pixels
  - Binary image requires 76800 bits of RAM
  - Maze solver path requires 4 bits for each displacement. Could get large for complicated paths

- Image noise
  - Misclassified walls
  - Erosion/Dilation may eliminate thin walls



Timeline

Week 1	- Image processing pipeline (RGB -> HSV, etc.)
Week 2:	- Project a predetermined path - Maze solving algorithm
Week 3	- Refine maze solving algorithm - Construct camera + projector mount - Put together setup
Week 4	- Debugging + Testing + Final Touches