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
- Process, then pass binary array to maze solving algorithm
RGB to HSV

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

\[
C_{\text{max}} = \max(R, G, B) \\
C_{\text{min}} = \min(R, G, B) \\
\Delta = C_{\text{max}} - C_{\text{min}}
\]

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

\[
S = \frac{\Delta}{C_{\text{max}}}
\]

\[
V = \frac{C_{\text{max}}}{255}
\]
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
2b. Maze Solving Algorithm: Lee’s Algorithm

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

<table>
<thead>
<tr>
<th>BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>4x76800</td>
</tr>
<tr>
<td>4'b0001</td>
</tr>
<tr>
<td>4'b0100</td>
</tr>
<tr>
<td>.</td>
</tr>
<tr>
<td>.</td>
</tr>
<tr>
<td>4'b1001</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Week</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>- Image processing pipeline (RGB -&gt; HSV, etc.)</td>
</tr>
<tr>
<td>Week 2:</td>
<td>- Project a predetermined path</td>
</tr>
<tr>
<td></td>
<td>- Maze solving algorithm</td>
</tr>
<tr>
<td>Week 3</td>
<td>- Refine maze solving algorithm</td>
</tr>
<tr>
<td></td>
<td>- Construct camera + projector mount</td>
</tr>
<tr>
<td></td>
<td>- Put together setup</td>
</tr>
<tr>
<td>Week 4</td>
<td>- Debugging + Testing + Final Touches</td>
</tr>
</tbody>
</table>