

# Grididdy: Robot Navigation Using AI Models

## 1 Perception

The robot uses two sensors:

- **Camera:** Detects the exact positions of adjacent wall tiles deterministically.
- **Magic Sensor:** Returns 1 if at least one adjacent tile is a danger tile. It does not indicate which one, and it cannot detect the goal.

## 2 State Estimation Using Hidden Markov Model (HMM)

We model the world with an HMM:

- Hidden state  $X_t$ : configuration of danger tiles
- Observation  $E_t$ : 1 if danger nearby, else 0
- Transition model:  $P(X_t | X_{t-1})$  (static environment, so  $X_t = X_{t-1}$ )
- Sensor model:  $P(E_t | X_t)$

We estimate:

$$P(X_t | E_{1:t})$$

to update beliefs about where danger tiles are.

## 3 State Creation

Each tile  $(x, y)$  has a label:

$$m_{x,y} \in \{\text{UNKNOWN, SAFE, WALL, SUSPECTED\_DANGER, CONFIRMED\_DANGER, GOAL}\}$$

Full robot state:

$$S_t = (x_t, y_t, \{m_{x,y}\}, \{b_{x,y}\})$$

where  $b_{x,y}$  is the belief (probability) that tile  $(x, y)$  is dangerous.

## 4 Knowledge Update using Bayesian Inference and Propositional Logic

When the sensor triggers ( $E_t = 1$ ), update belief of adjacent unknowns using Bayes' Rule:

$$P(D_{x,y} | E_t = 1) = \frac{P(E_t = 1 | D_{x,y})P(D_{x,y})}{\sum_j P(E_t = 1 | D_{x_j,y_j})P(D_{x_j,y_j})}$$

If only one adjacent tile is unknown, and sensor is active:

$$m_{x,y} \leftarrow \text{CONFIRMED\_DANGER}, \quad b_{x,y} = 1.0$$

If multiple unknowns:

$$b_{x,y} \leftarrow \min(b_{x,y} + 0.3, 1.0) \quad \text{and} \quad m_{x,y} \leftarrow \text{SUSPECTED\_DANGER}$$

If no danger is detected, downgrade nearby  $D$  tiles to SAFE and reset  $b_{x,y} = 0$ .

## 5 State Space Search

The grid is treated as a graph. Each tile is a node. We prune:

- WALL tiles
- CONFIRMED\_DANGER tiles

The robot considers paths toward unknown areas (?) first. If none are reachable, it tries safe paths, then slightly risky ones.

## 6 Path Cost Function

The robot evaluates each path using:

$$\text{cost\_total} = \sum_{\text{path}} (b_{x,y} + 1.0 \cdot \text{visit\_count}_{x,y})$$

## 7 Action

Actions:

$$A = \{\text{UP, DOWN, LEFT, RIGHT}\}$$

The robot moves to the next tile in the lowest-cost path.

## 8 Statistics

At the end of the simulation, the robot reports:

- Steps taken
- Number of unique tiles explored
- Number of slightly dangerous moves (where  $b_{x,y} \geq 0.3$ )

## Full Execution Flow (Simplified)

### 1. Initialize the environment:

- Place 6 wall tiles, 6 danger tiles, 1 goal, and the robot.

### 2. Perceive surroundings:

- Use camera to mark adjacent WALLs.
- Use magic sensor to detect nearby danger (if any).

### 3. Update beliefs:

- If sensor triggers:
  - If only 1 adjacent UNKNOWN tile: mark it as `CONFIRMED_DANGER`,  $b = 1.0$ .
  - If multiple: mark as `SUSPECTED_DANGER`, increase  $b$  by  $+0.3$ .
- If no danger: downgrade adjacent  $D$  to `SAFE` and set  $b = 0.0$ .

### 4. Track visits:

- Increment visit count on every tile visited.

### 5. Plan path using BFS:

- Avoid WALL and `CONFIRMED_DANGER` tiles.
- Prioritize paths adjacent to unknown tiles.
- Choose the path with the lowest total cost:

$$\sum_{\text{path}} (b_{x,y} + 1.0 \cdot \text{visit\_count}_{x,y})$$

### 6. Move the robot:

- Step to the next tile in the path.
- Mark as `SAFE` and record stats.

### 7. Repeat until goal reached or trapped:

- Simulation stops when goal is reached or no safe path remains.