# 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: $P(X_t \mid X_{t-1})$ (static environment, so $X_t = X_{t-1}$)
- Sensor model: $P(E_t \mid X_t)$

We estimate:

$$P(X_t \mid E_{1:t})$$

to update beliefs about where danger tiles are.

## 3 State Creation

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

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

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:

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

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

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

# 5 Reasoning with Bayesian Dynamic Network

The joint probability model over time is:

$$P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=1}^{t} P(X_i \mid X_{i-1}) P(E_i \mid X_i)$$

We use filtering to maintain:

$$P(X_t \mid E_{1:t})$$

# 6 State Space Search

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

- WALL tiles
- CONFIRMED_DANGER tiles

Cost function:

$$\text{cost}(x, y) = 1 + \lambda \cdot b_{x,y}$$

where $\lambda$ is a penalty multiplier (e.g., 100).

# 7 Action

Actions:

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

The policy is derived using the Bellman equation:

$$U(S) = R(S) + \gamma \max_a \sum_{S'} P(S' \mid S, a) U(S')$$

Optimal action:

$$\pi^*(S_t) = \arg\max_a \sum_{S'} P(S' \mid S_t, a) U(S')$$

Rewards:

- Reaching goal: $+1000$
- Stepping on danger: $-1000$
- Any movement: $-1$

# 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` and set belief = 1.
     - If multiple: mark them as `SUSPECTED_DANGER` and increment their belief by +0.3.
   - If no danger detected: downgrade nearby $D$ tiles to `SAFE`.
4. **Track visits**:
   - Count how many times each tile has been visited.
5. **Plan path using BFS**:
   - Avoid WALL and CONFIRMED_DANGER tiles.
   - Cost of each tile is:
     $$\text{cost}(x, y) = b_{x,y} + 0.2 \cdot \text{visit\_count}_{x,y}$$
   - Choose the path with the lowest total cost.
6. **Move the robot**:
   - Step to the next tile in the best path.
   - Mark it SAFE and increment visit count.
7. **Repeat until goal reached or stuck**:
   - Stop if the goal is found or no valid moves remain.