
2

3

4

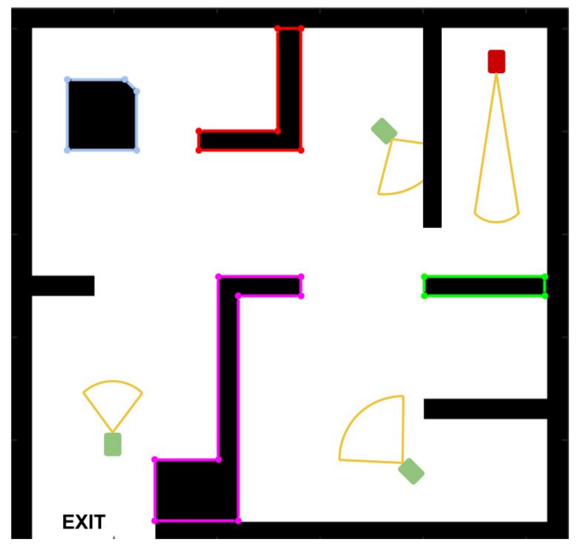
Topological Navigation for Multi-Robot Pursuit-Evasion

Aren Petrossian[1,2], Arsh Singhal[1,3], & Joseph Hardin[1,4]

[1]Georgia Institute of Technology

Introduction

Objective: Adapt topological navigation + effective RL agent for a multi-robot system to collaboratively locate and corner smart, mobile evader



Environment Constraints:
- Limited Sensing → No metric (SLAM) map
- Limited Bandwidth → Can't communicate large data
- Unknown environment → Have to explore
- Limited FOV and speed

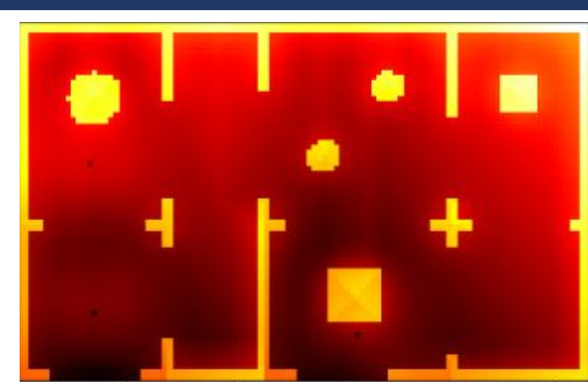
Motivation: Efficient, fast localization of targets using a system of robots would be very useful in real-life deployments of indoor security robots → minimize risks for humans. Also, an explosion in real-life robotics, so it's important now. Our environment constraints were designed to closely mimic real-life difficulties.


Prior Work:
- Pursuit-evasion has been studied in many fields
- Other robotics work use known maps, single pursuer, centralized communication, etc.
- No use of topologies + RL (TopoNav) with multiple robots

Methods

Core Contributions & Insights:
- Realistic scenario needed for useful applications → **lack of intense computation or communication**
- Framework for **multi-robot topological mapping** + exploration through **shared landmarks**
- MDP logic + deep RL agent for **autonomous subgoal selection**
- Designed **extrinsic + intrinsic rewards** to balance different goals

POTENTIAL FIELD EVADER



Static: A* attraction + wall repulsive + wall raising
Dynamic: Repulsion radius near pursuers
Fallback: Pushed in opposite direction → tries using A* to get around pursuers and to goal

RL-AGENT PURSUER (Single Policy)

Markov Decision Process (MDP)

State: G(V, E) of topological map

Edges: traversable path from nodes
Nodes: Pre-defined landmark
- # times visited
- t since last visit
- who visited last
- uncertainty
- dist. to curr loc
- # of neighbors

Action: best subgoal *g* in V

Rewards: Extrinsic + Intrinsic

Extrinsic: Seeing Evader
Intrinsic:
- increase area explored
- re-visit stale (by time) nodes
- help expand frontier nodes

Loop till Evader found

DQN: $Q(s_g, g; \theta)$
- GCN to get node embeddings
- Feed through Q network and pick subgoal with max Q

Local Planner + Movement:
- Find path to *g* through graph
- Assume optimal local planning to get to neighbor node (A*)

While Moving (src → tgt):
- If landmark seen, add node to graph with edge to *src* and *tgt*

Reached node *g*:
- Update state and reward
- Expand neighbor nodes of *g* if seen by another pursuer
- If no unvisited neighbors, add 'artificial' node towards any unexplored region

Experimental Design

BASELINE METHODS
- **Random Agent:** Picks random subgoal to traverse to as topological graph of landmarks grow normally
- **Greedy Agent:** Greedily selects a seen, but unvisited, node as subgoal from topology
- **Naive Patrol (extra knowledge):** Doesn't need to explore since it has hardcoded patrol paths for each pursuer. Once any pursuer finds evader, all pursuers towards evader with help of A* to avoid immediate obstacles

*If seen evader → chase evader, note close landmarks

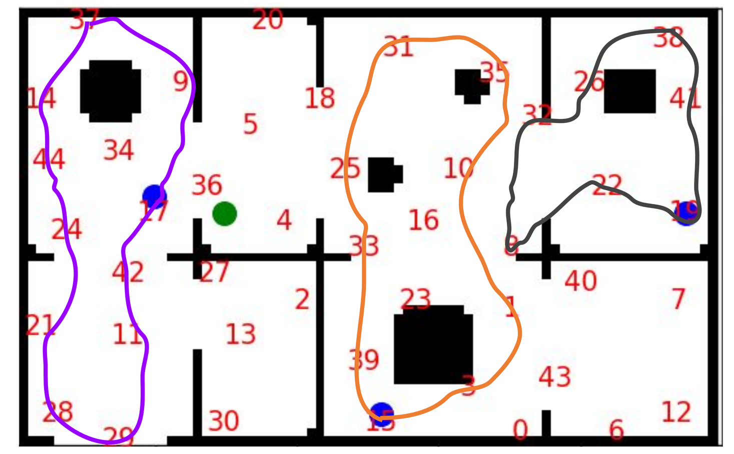


Figure 1: Hard-coded patrol paths for each pursuer

Training Reward: $R_{\text{ext}} = 50 \cdot \mathbf{1}_{\{\text{caught}\}}$

$$R_{\text{int}} = 40 \times \frac{\Delta A}{4000} - 0.5\,\frac{(\bar{n}-1)}{\sqrt{\bar{n}}} + 0.1\,\tanh\left(\frac{\bar{t}}{200}\right)$$

ΔA: new area seen, **n:** num visits, **t:** time since last visit

METRICS

Run each pursuer strategy on varying evader speeds
- **Full Capture %:** All 3 pursuers see evader
- **# of Successful Pursuers:** 0-3
- **% ≥ 1, ≥ 2, ≥ 3 Successful Pursuers**
- **Average Episode Time:** Iterations to Chase/Evade

Hypotheses:
- **H1:** RL-agent achieves higher capture rate than all baselines, esp. with faster evader
- **H2:** Intrinsic reward will push RL-agent to explore a lot
- **H3:** Naive patrol will be strong because of extra knowledge given to it

Results

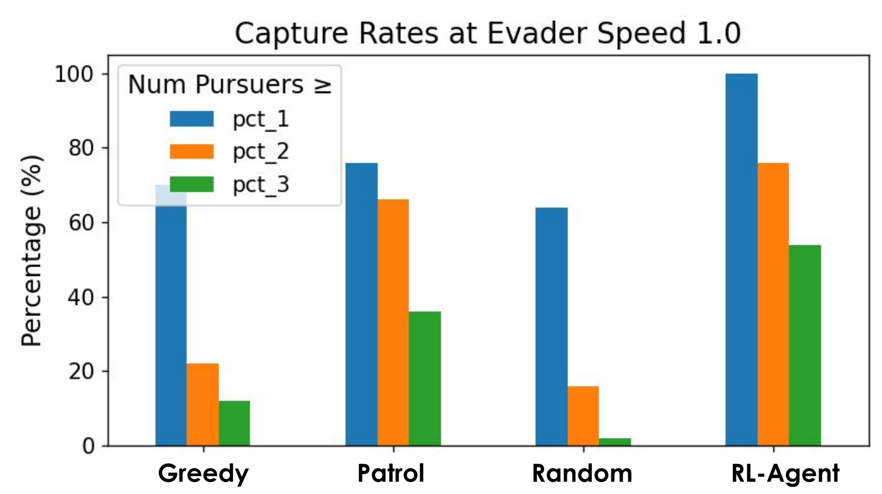


Table 1: Avg. Chase Time for Full Captures

Strategy	0.33	0.50	0.75	0.90	1.0
Greedy	577.3	510.0	377.0	848.6	350.5
Naive Patrol	**100.4**	**94.5**	**109.6**	**94.0**	**73.2**
Random	609.8	672.6	434.6	983.8	1488.0
RL-Agent	543.2	533.2	763.0	648.2	667.1

Table 2: Full Capture Rate (%)

Strategy	0.33	0.50	0.75	0.90	1.0
Greedy	46	16	22	16	12
Naive Patrol	**100**	**100**	**74**	60	36
Random	30	22	10	10	2
RL-Agent	52	58	60	**66**	**54**

Table 3: Avg. Number of Successful Pursuers

Strategy	0.33	0.50	0.75	0.90	1.0
Greedy	2.28	1.52	1.45	1.20	1.04
Naive Patrol	**3**	**3**	**2.38**	1.80	1.78
Random	1.86	1.62	0.88	1.02	0.82
RL-Agent	2.52	2.36	2.34	**2.32**	**2.30**

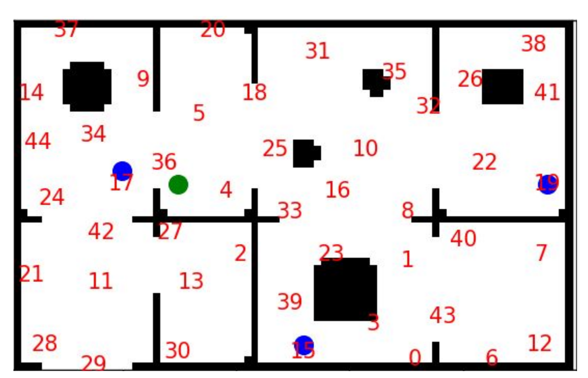


Figure 2: Map with Pre-Set Landmarks

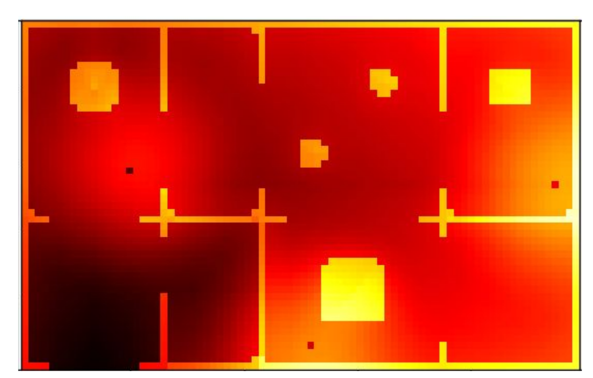


Figure 3: Evader Pot. Field from Fig. 1

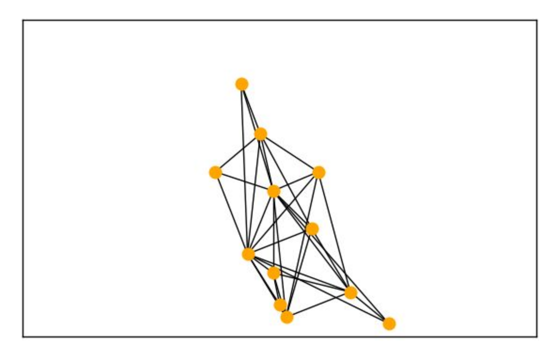


Figure 4: Example Pursuer Topology

Takeaways:
- RL-Agent outperforms all baselines in key metrics besides against Naive Patrol for slower evaders (0.33 - 0.75 speed). Note that Naive Patrol is not a fair comparison since it has more hard-coded knowledge (i.e. patrol routes + uses A*)
- RL-Agent has 100% capture rate for evaders of all speeds.

Discussion of Results

Notable Phenomena:
- **Robust Learned Subgoal Selection:** RL-Agent maintains ≥ 50% full-capture rate while other baselines drop to 10%-30%
- **Strategic vs. Reactive Coverage:** Naive patrol collapses at top speed → simple compass-heading lacks flexibility

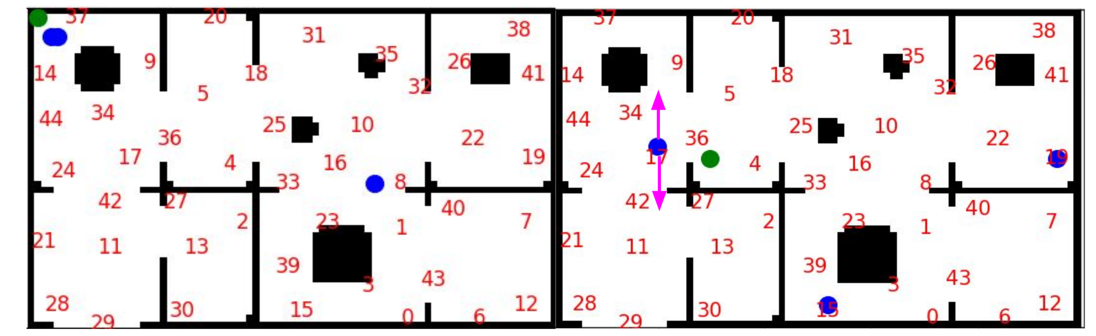


Figure 5: Learned Strategies → Cornering (Left) and Cutoff (Right)

- **Efficiency of Topological Fusion:** RL approach matches/outperforms baseline that implicitly "sees" full grid. Lightweight topology framework is successful
- **Faster Chase Time for Naive Patrol:** Metric maps has faster chase down since exact navigation details known
- **Consistent Capture Rate (through speeds):** RL-Agent builds performance as training speed increases → could use more episodes at each speed

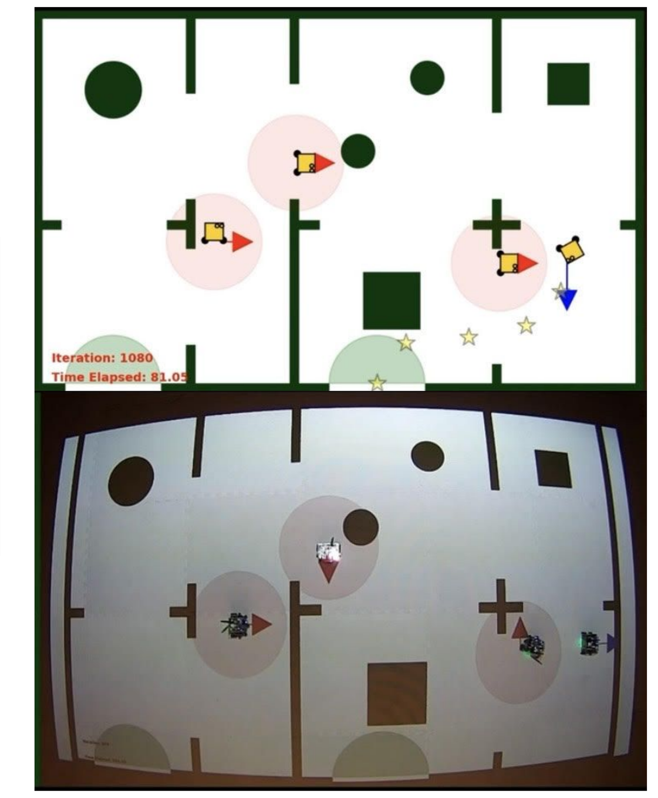


Figure 6: Potential Field Evader + Naive Pursuers in Simulator (top) and Robotarium Deployment (bottom)

Limitations
- **Limited Training + Experimentation:** Many more combinations of rewards, training schedules, features, etc. could be tested to reach optimality
- **Not Robust to Randomness:** Evader has semi-random start but pursuers are fixed starts. Haven't fully tested or trained on random starts
- **Not Robust to Evader Variety:** We fixed evader potential field and A* parameters for "realistic" evader, but could be smarter in real-life
- **Pre-Set Landmarks:** Landmarks are static and pre-defined. No real-time detection or formation
- **Suitability for Real Deployment:** Only worked with 2D environment, ignores struggles of 3D world. Also, loose "capturing" / apprehension mechanism

Conclusion

RL-Agent achieves **robust coordination, lightweight mapping**
- Reliably **outperforms other baselines** in capture rate, chase time

Builds on prior work in Pursuit-Evasion and navigation such as TopoNav
- First time **combining topologies, deep RL, multi-robots** for Pursuit-Evasion

Real-world application for 'apprehend-the-intruder' with robot security guards

Future Direction: Investigate more reward structures and features to enable faster chase-down. Generate landmarks real-time based on 3D features