Distributed Pursuit-Evasion with Limited-Visibility Sensors via Frontier-based Exploration

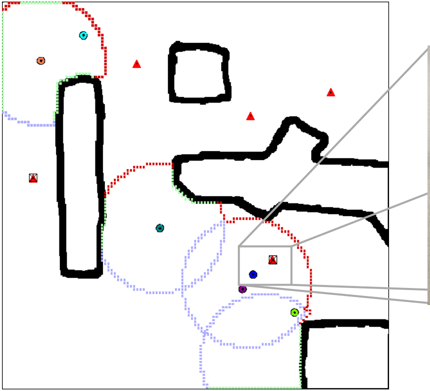
Joseph W. Durham* Antonio Franchi** Francesco Bullo*

*Center for Control, Dynamical Systems, and Computation
Department of Mechanical Engineering
University of California at Santa Barbara

**Department of Computer and System Sciences
Università di Roma "La Sapienza"

ICRA
Anchorage, Alaska, May 5, 2010

Pursuit-evasion



T34 security bot from tmsuk and Alacom in Japan

Our Pursuit-evasion Problem

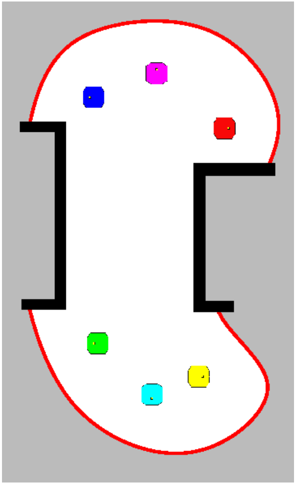
Assumptions

- A team of robots with **limited-range** sensors (pursuers)
- **Unknown** environment (non-polygonal, multiply connected)
- The pursuers start from the **same point**
- An evader is only **detected when seen** by a pursuer
- The evaders have **unlimited speed and knowledge**
- **Finite memory**-size per pursuer $\Rightarrow$ no global map

Problem (Distributed pursuit-evasion or environment clearing)

Design a **distributed** *method to* **guarantee detection** *of any evaders in the environment*

Pursuit-evasion vs. Exploration



Clearing an environment

$\Updownarrow$

A constrained form of exploration
(no recontamination)

For stationary evaders:

- cleared $\equiv$ explored

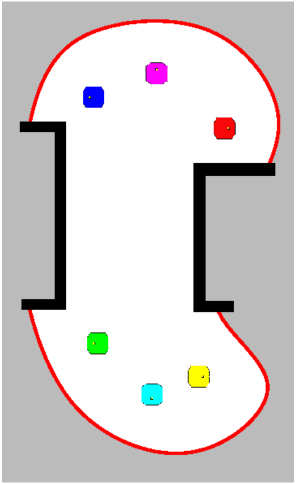
Otherwise:

- cleared can be **recontaminated**

Cooperative Exploration:

Franchi et Al. *A Randomized Method for Cooperative Robot Exploration* ICRA 2007

Frontier in Pursuit-evasion



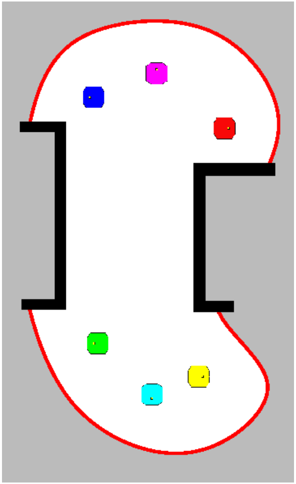
For exploration
Frontier = boundary between explored and unexplored areas

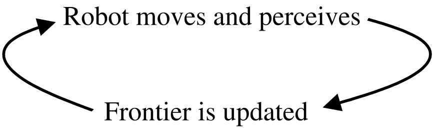
For pursuit-evasion
Frontier = boundary between cleared and contaminated areas

Our approach at each step

- Cover the frontier
- Push back the frontier as much as possible

Requirements for Frontier Updating





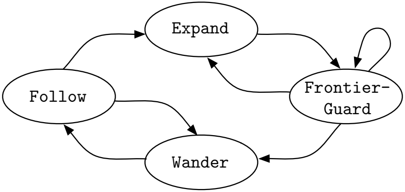
Exploration-based methods require:

- **Global map**
- **Global localization**
  (to build global map)

Our method requires:

- **Short-term mutual localization**
  between neighboring robots

Distributed Algorithm: Robot Behaviors



Leader behaviors:

Frontier-guard: Cover a local piece of the frontier and dispatch followers to new viewpoints.

Expand: Move to a viewpoint, sense, and update local frontier.

Non-Leader behaviors:

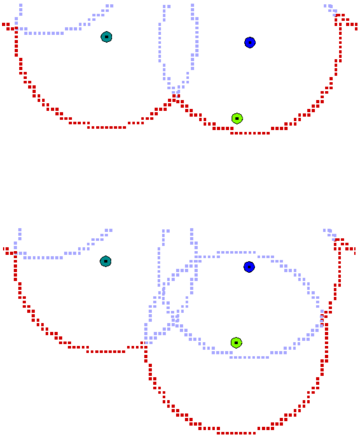
Follow: Shadow a frontier-guard and wait for orders.

Wander: Search for a guard to follow.

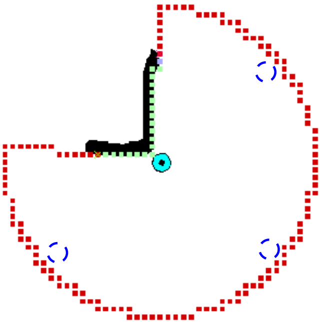
Distributed Algorithm: Frontier Updating

When an expander reaches its viewpoint, it must:

1. Ask for frontier arcs from neighboring guards (requires temporary mutual localization)
2. Inform neighbors when their frontier segments lie inside its sensor footprint
3. Determine local frontier based on intersections



Distributed Algorithm: Viewpoint Planner



A frontier-guard picks new viewpoints V:

- Minimizing $|V|$
- Maximizing area exposed

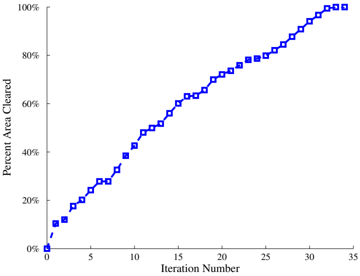
$\#$ of viewpoints required for frontier arc with angular width Ω:

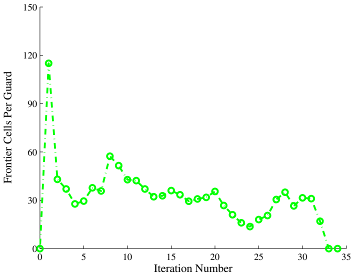
- $|V| = 1$, if $\Omega \leq \frac{2\pi}{3}$
- $|V| = 3$, if $\Omega = 2\pi$

If $\frac{2\pi}{3} < \Omega < 2\pi$, choice to optimize $|V|$ or the area exposed

Movie

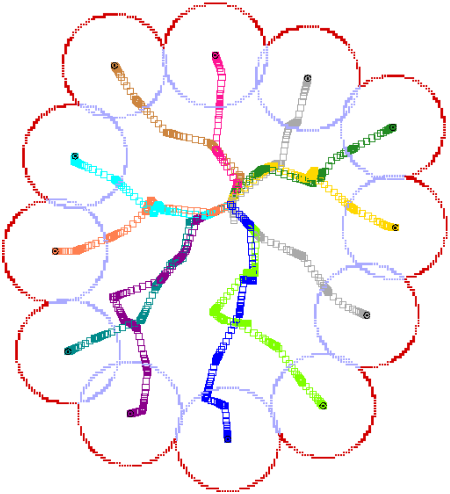
Frontier Storage Requirements





- $\#$ of frontier cells per guard independent of area cleared
- Distributed storage requires constant memory per agent

Expansion in Empty Space



Summary

Primary contributions

- Online clearing algorithm which works in **non-polygonal** environments with **holes**
- **Distributed** storage and updating of global frontier
- Requires only **temporary mutual localization** with neighbors
- Requires only **constant memory** per agent (w.r.t. environment size)

Current directions

- Distributed hardware implementation and experiments
- Viewpoint planner for more general sensor footprints
- Bounds on number of agents necessary to clear a map