On Distributed Coordination in Robotic Networks

Gossip Coverage and Frontier-based Pursuit

Joseph W. Durham

Center for Control, Dynamical Systems, and Computation
Department of Mechanical Engineering
University of California at Santa Barbara
motion.mee.ucsb.edu/~joey

February 5, 2010

Distributed Coordination Algorithms

Team of robotic agents tasked with performing a joint mission in an environment

Each individual

- *senses* its immediate surroundings
- *communicates* with nearby agents
- *processes* information gathered
- *performs* local action in response

Distributed Coordination Algorithms

Team of robotic agents tasked with performing a joint mission in an environment

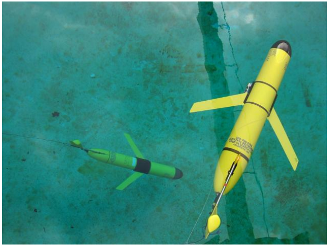
Each individual

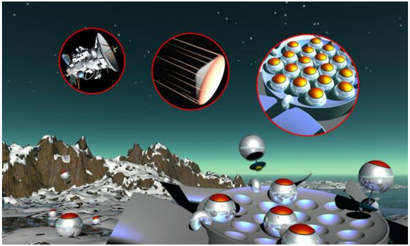
- *senses* its immediate surroundings
- *communicates* with nearby agents
- *processes* information gathered
- *performs* local action in response

Algorithm design goal

Design individual control and communication laws such that the group reaches a desired goal

Applications





Ocean monitoring gliders from `noc.soton.ac.uk`, warehouse robots from KIVA Systems,

hopping planetary explores from NASA

Papers in this Talk

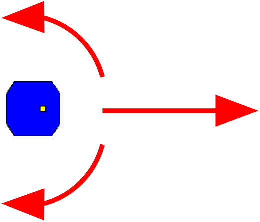
- J. W. Durham, A. Franchi, and F. Bullo. Distributed pursuit-evasion with limited-visibility sensors via frontier-based exploration. In *IEEE Int. Conf. on Robotics and Automation*, Anchorage, Alaska, May 2010. To appear
- J. W. Durham, R. Carli, P. Frasca, and F. Bullo. Discrete partitioning and coverage control with gossip communication. In *ASME Dynamic Systems and Control Conference*, Hollywood, CA, October 2009
- J. W. Durham and F. Bullo. Smooth nearness-diagram navigation. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 690–695, Nice, France, September 2008

Collaborators: Ruggero Carli, Antonio Franchi, Paolo Frasca, and my advisor Francesco Bullo.

Outline

Hardware

Robot Model



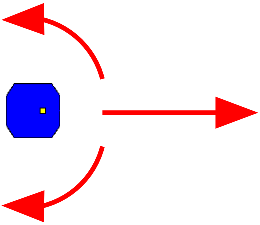
Differential drive

- Translational velocity v
- Rotational velocity $\dot{\theta}$

Physical state

$\mathcal{X} = (x, y, \theta)$

Robot Model



Differential drive

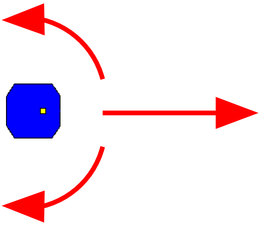
- Translational velocity v
- Rotational velocity $\dot{\theta}$

Physical state

$\mathcal{X} = (x, y, \theta)$

In theory state is an integral of velocities

Robot Model



Differential drive

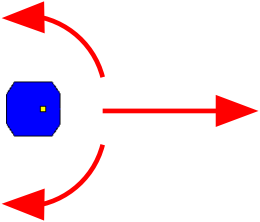
- Translational velocity v
- Rotational velocity $\dot{\theta}$

Physical state

$\mathcal{X} = (x, y, \theta)$

In practice measurement of actual velocities is imperfect, integrals diverge

Robot Model



Differential drive

- Translational velocity v
- Rotational velocity $\dot{\theta}$

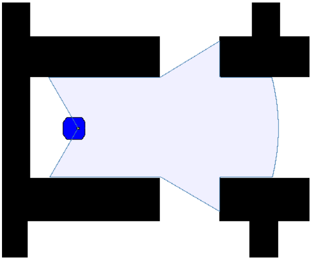
Physical state

$\mathcal{X} = (x, y, \theta)$

In practice measurement of actual velocities is imperfect, integrals diverge

Either must accept position errors or use sensors for localization

Sensor Model



Sensor footprint

$\mathcal{S}(x, y, \theta)$ is the intersection of visibility polygon from (x, y) and the area perceivable by the sensor oriented by θ

Sensor footprint can be used for:

- Obstacle detection
- Localization
- Intruder detection

Control and Communication Models

Processor State

$\mathcal{W}$: the state of the robot's processor – stored data, current behavior

Communication Alphabet

$\mathcal{L}$: set of messages a robot can send to other robots

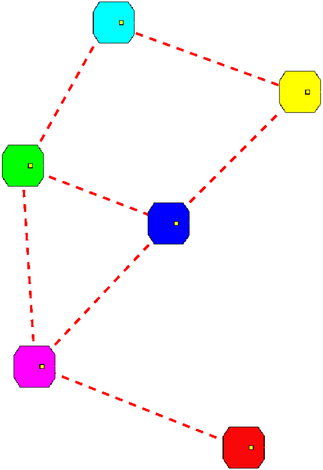
Network Connectivity Models

Communication Graph

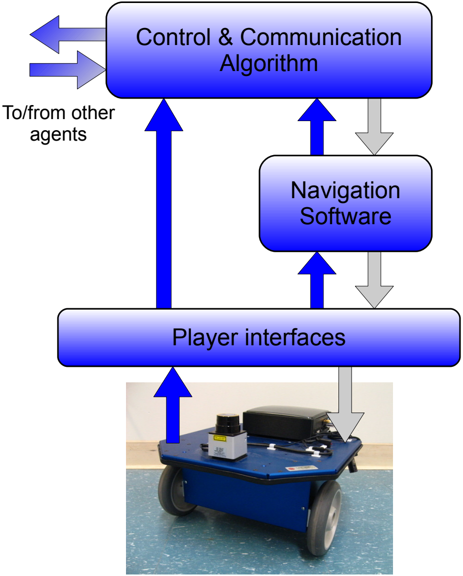
Many possible models for which agents can communicate

Combinations of:

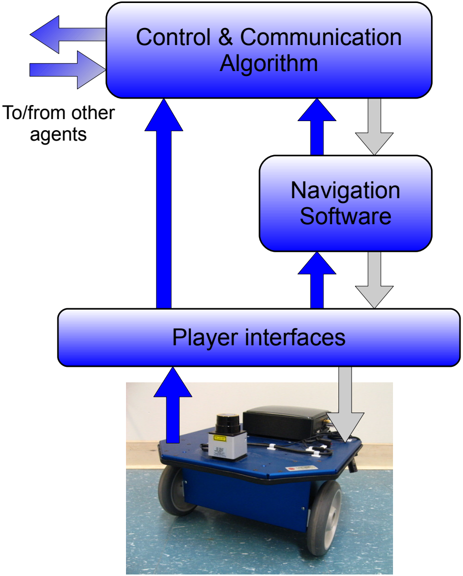
- Network geometry
- Physical proximity
- Current robot roles
- Randomness



Robotic Software Overview



Robotic Software Overview

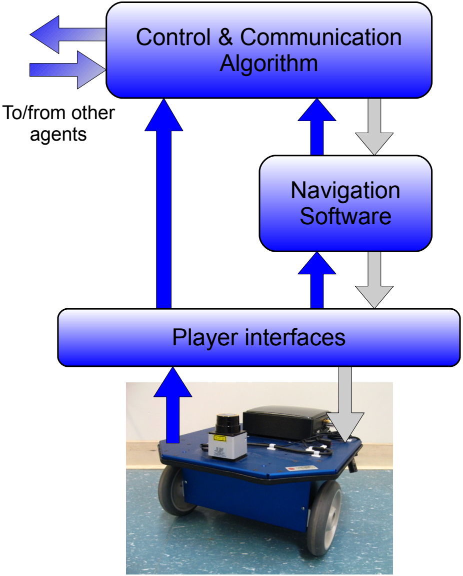


Player/Stage is an open-source robotics software library

Features

- Player provides interfaces for hardware
- Each robot is a server on a TCP/IP network
- Stage simulates hardware, interfaces to algorithms are the same

Robotic Software Overview



SND Navigation handles local path planning and execution

Algorithm design can focus on:

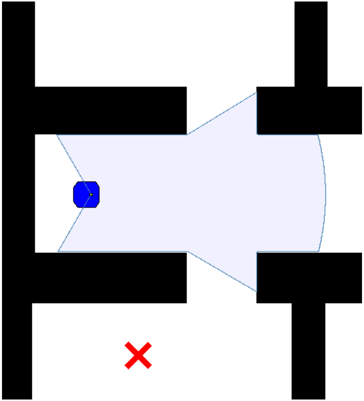
- Desired positioning of robot
- Communication for coordination

Smooth Nearness Diagram Navigation

Evolution of ND+ Nav by J. Mingues, J. Osuna, L. Montano

Input: $\mathcal{S}$, desired pose (x, y, θ)

Output: v and $\dot{\theta}$

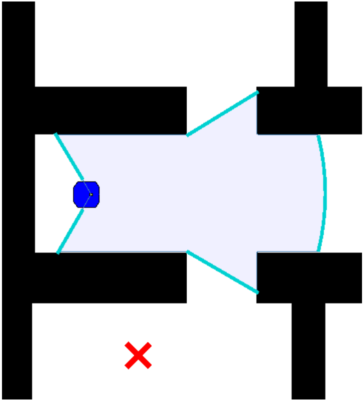


Smooth Nearness Diagram Navigation

Input: $\mathcal{S}$, desired pose (x, y, θ)

Output: v and $\dot{\theta}$

- Find gaps in sensor footprint

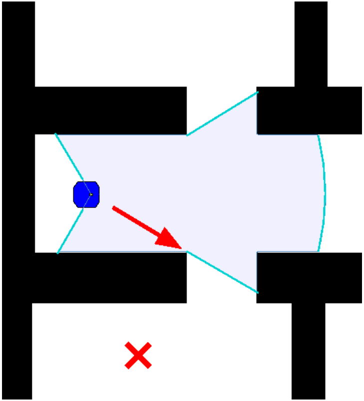


Smooth Nearness Diagram Navigation

Input: $\mathcal{S}$, desired pose (x, y, θ)

Output: v and $\dot{\theta}$

- Find gaps in sensor footprint
- Pick best gap to drive towards

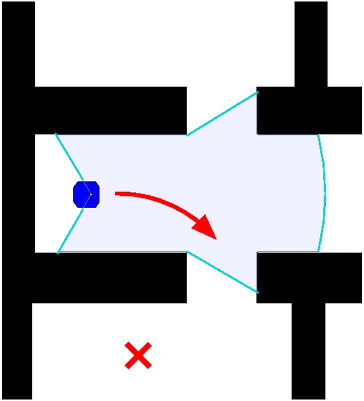


Smooth Nearness Diagram Navigation

Input: $\mathcal{S}$, desired pose (x, y, θ)

Output: v and $\dot{\theta}$

- Find gaps in sensor footprint
- Pick best gap to drive towards
- Adjust commands based on nearby obstacles



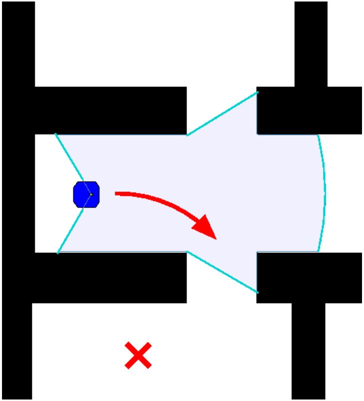
Smooth Nearness Diagram Navigation

Input: $\mathcal{S}$, desired pose (x, y, θ)

Output: v and $\dot{\theta}$

- Find gaps in sensor footprint
- Pick best gap to drive towards
- Adjust commands based on nearby obstacles

Available as the `snd` driver in Player/Stage



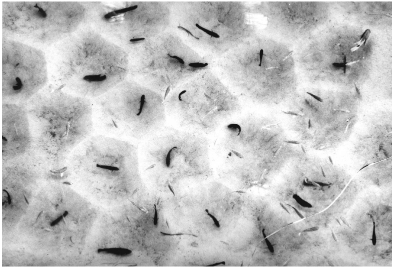
Summary of Robotic Network Model

Algorithm Design Requirements

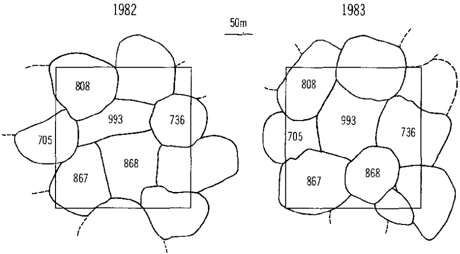
1. **Data structures**
   - Correct or account for errors in (x, y, θ)
   - Processor state $\mathcal{W}$ and communication alphabet $\mathcal{L}$
2. **Update functions**
   - Message-generation function
   - Processor state transition function
   - Motion control function to pick desired pose
3. **Communication graph model**

Motivation

Biological examples of coverage control



Tilapia mossambica, Barlow et al '74



Sage sparrows, Petersen et al '87

Related Prior Work I

Lloyd's Algorithm

- take convex environment Q with density function $\phi : Q \to \mathbb{R}_{\geq 0}$
- place N robots at $p = \{p_1, \dots, p_N\}$
- partition environment into $v = \{v_1, \dots, v_N\}$
- define expected quadratic deviation

$$H(v,p) = \int_{v_1} f(\|q - p_1\|)\phi(q)dq + \ldots + \int_{v_N} f(\|q - p_N\|)\phi(q)dq$$

Related Prior Work I

Lloyd's Algorithm

- take convex environment Q with density function $\phi : Q \to \mathbb{R}_{\geq 0}$
- place N robots at $p = \{p_1, \dots, p_N\}$
- partition environment into $v = \{v_1, \dots, v_N\}$
- define expected quadratic deviation

$$H(v, p) = \int_{v_1} f(\|q - p_1\|)\phi(q)dq + \ldots + \int_{v_N} f(\|q - p_N\|)\phi(q)dq$$

Theorem (Lloyd '57 "least-square quantization")

1. *at fixed partition, optimal positions are centroids*
2. *at fixed positions, optimal partition is Voronoi*
3. *Lloyd algorithm: alternate p-v optimization*

$\longrightarrow$ *convergence to the set of centroidal Voronoi partitions*

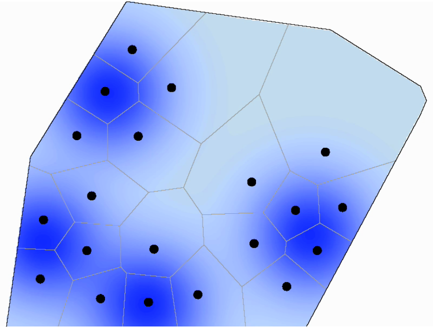
Related Prior Work II

Distributed Coverage Control

At each comm round:

1: acquire neighbors' positions
2: compute Voronoi region
3: move towards centroid of own Voronoi region

Result: convergence to the set of centroidal Voronoi partitions

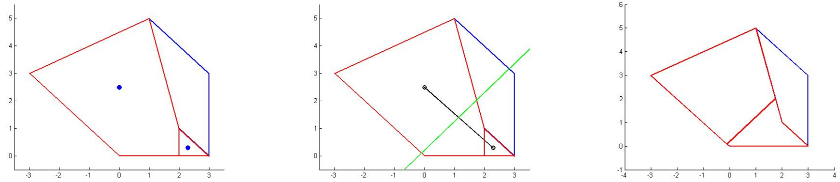


J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004

Related Prior Work III

Gossip coverage in continuous space

- Pairwise territory exchange between neighbors
- Regions may be non-convex during evolution
- Result: convergence to the set of centroidal Voronoi partitions



P. Frasca, R. Carli, and F. Bullo. Multiagent coverage algorithms with gossip communication: control systems on the space of partitions, March 2009. Available at http://arXiv.org/abs/0903.3642

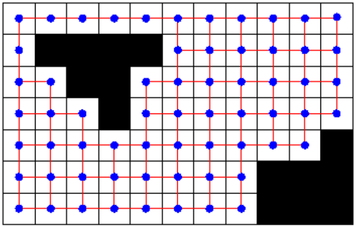
Discretized Environments

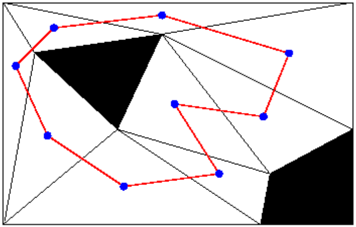
Domain is a weighted graph $G = (Q, E, w)$

Required properties

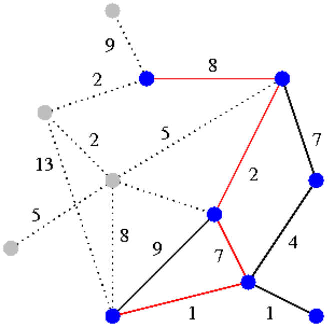
- G must be connected
- All edge-weights w must be positive

G can easily represent a non-convex environment with holes

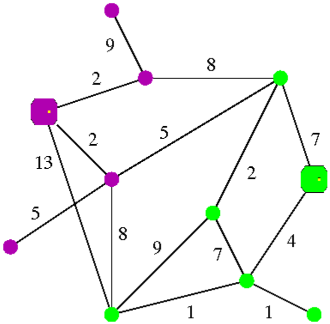




Voronoi Iteration on Graphs



Distances are shortest path lengths in connected sub-graphs of G



Vertices join partition of centroid they are closest to

Cost Function

Centroid p_i of sub-graph v_i is vertex which minimizes

$$H_i(h, v_i) = \sum_{k \in v_i} \text{dist}_{v_i}(h, k)$$

Total cost

$$\mathcal{H}_{\text{multi-center}}(p, v) = \sum_{i=1}^{N} H_i(p_i, v_i)$$

Cost Function

Centroid p_i of sub-graph v_i is vertex which minimizes

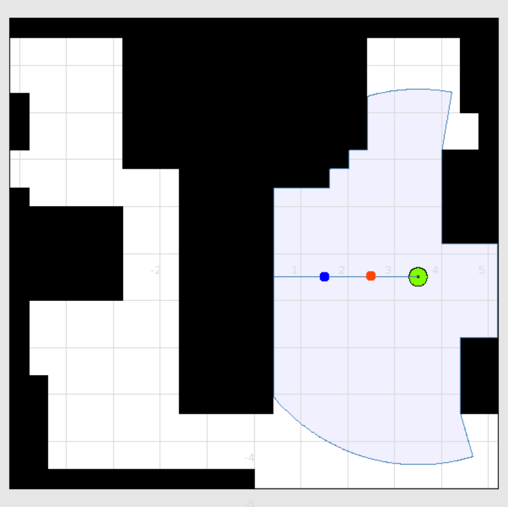
$$H_i(h, v_i) = \sum_{k \in v_i} \text{dist}_{v_i}(h, k)$$

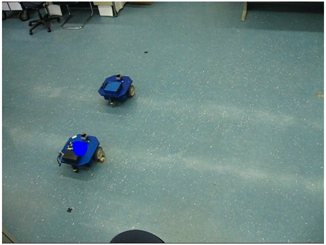
Total cost

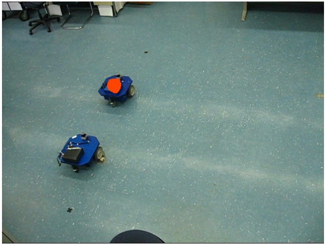
$$\mathcal{H}_{\text{multi-center}}(p, v) = \sum_{i=1}^{N} H_i(p_i, v_i)$$

Minimize expected distance between random vertex and closest robot

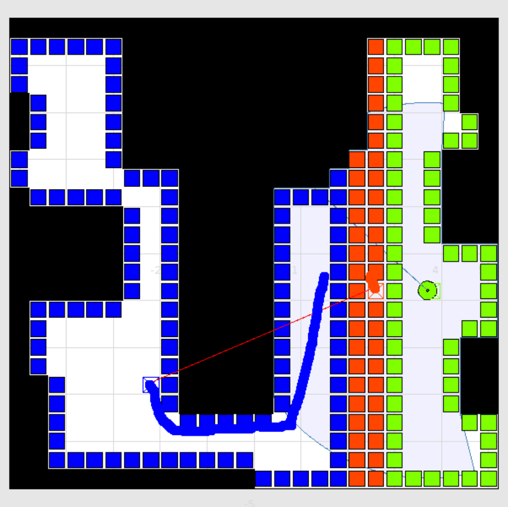
Hardware Experiment

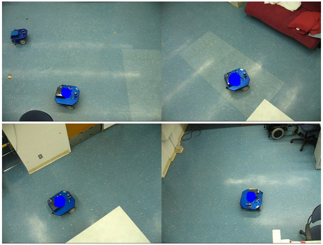


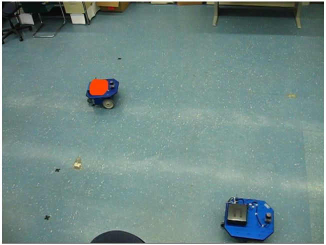




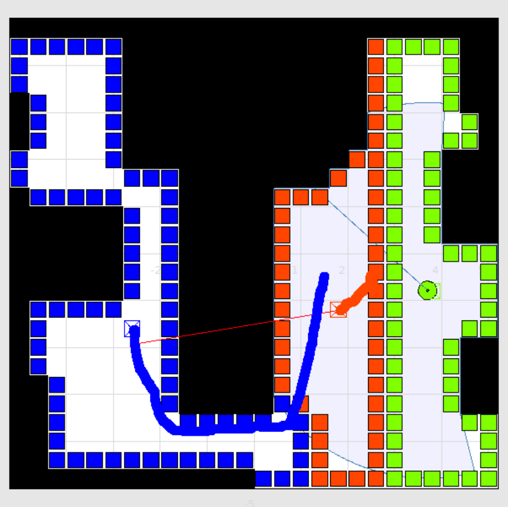
Hardware Experiment



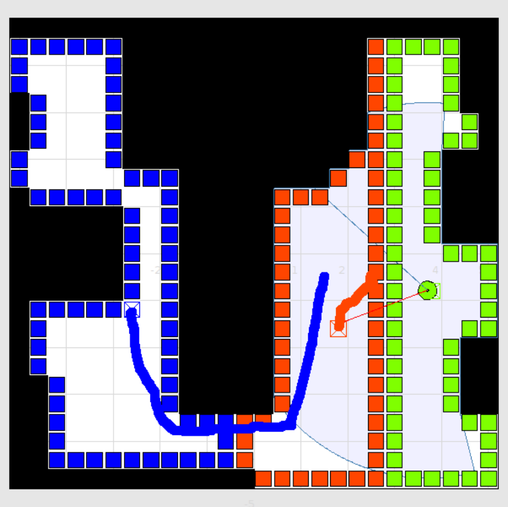




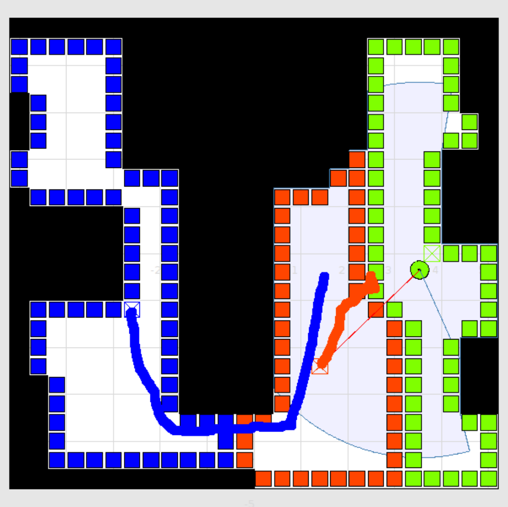
Hardware Experiment

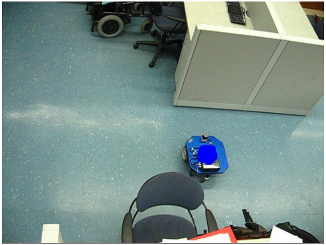


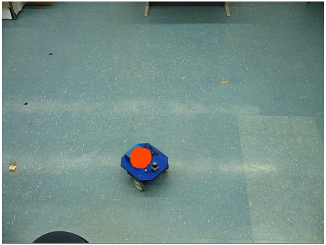
Hardware Experiment



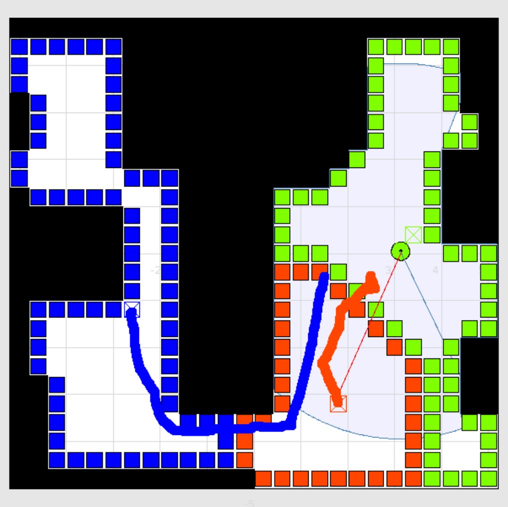
Hardware Experiment

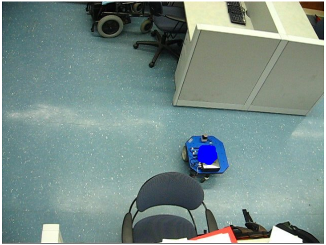


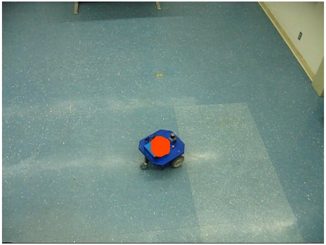




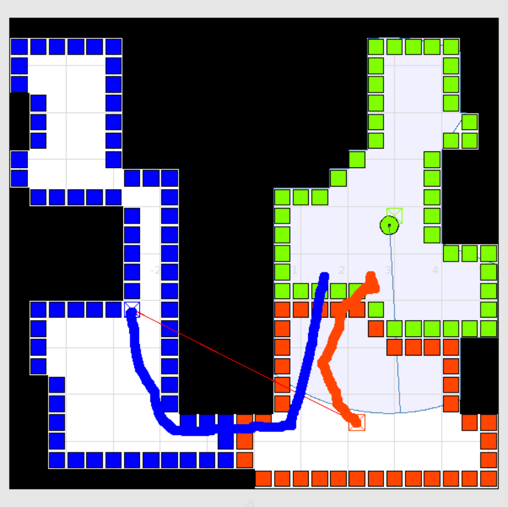
Hardware Experiment

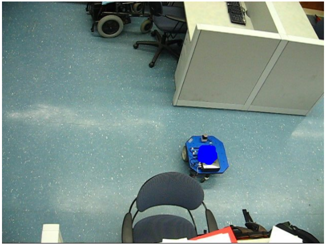


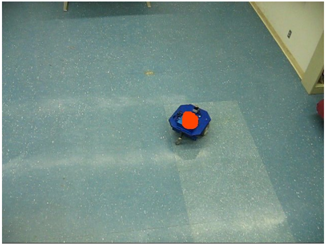




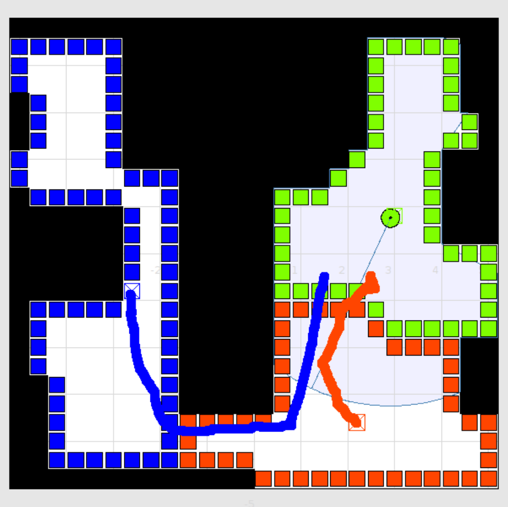
Hardware Experiment

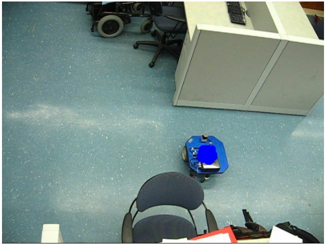


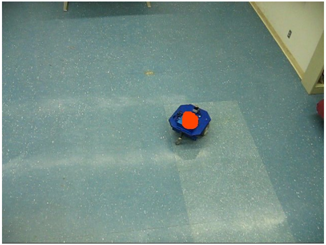




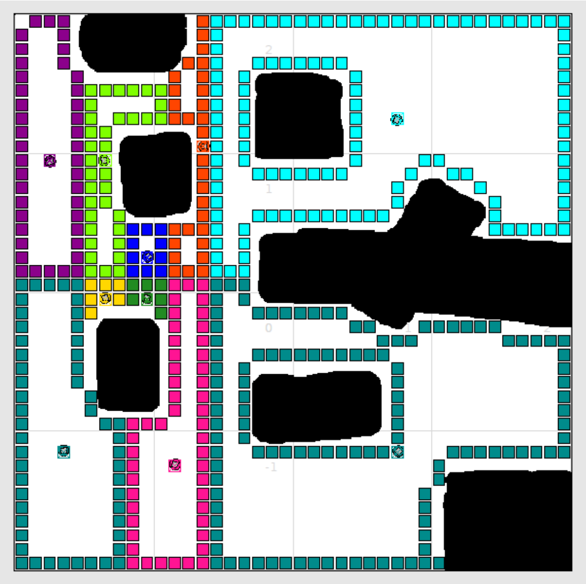
Hardware Experiment







Simulation Movie



Gossip Coverage Assumptions

Map assumptions:

- Team is provided an initial connected N-partition of environment
  - Initial agent partitions are connected
  - Cover space without overlap

Gossip Coverage Assumptions

Map assumptions:

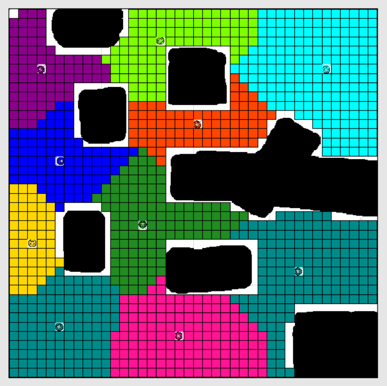
- Team is provided an initial connected N-partition of environment
  - Initial agent partitions are connected
  - Cover space without overlap

Communication assumptions:

- Given infinite time, each agent will talk to each of its neighbors an infinite number of times
- Two options:
  - There exists a finite upper bound on the time between conversations for each pair
  - There is a non-zero probability for each pairwise communication occurring at all times

Algorithm Claims

1. Maintain connected *N*-partition during evolution
   - Each region is connected
   - No overlap
2. Total cost decreases whenever agents exchange territory
3. Provable convergence to a single centroidal Voronoi partition in finite time



Convergence Theorem

- X finite set of connected N-partitions of graph G
- Algorithm defines set-valued map $T : X \rightarrow X$

Convergence Theorem

- X finite set of connected N-partitions of graph G
- Algorithm defines set-valued map $T : X \to X$

Version of the LaSalle Invariance Principle

Requirements for convergence

1. X is compact, positively invariant under T
2. $\mathcal{H}_{\text{multi-center}}$ non-increasing under T, decreasing under $T \setminus \{id\}$
3. $\mathcal{H}_{\text{multi-center}}$ and T are continuous on X
4. One of two communication assumptions
   - There exists a finite upper bound on the time between conversations for each pair (i, j)
   - There is a non-zero probability for each pair (i, j) to communicate at all times

Computational Complexity

$$H_i(h, v_i) = \sum_{k \in v_i} \text{dist}_{v_i}(h, k)$$

Key computation

Distances from h to all $k \in v_i$

- If edge-weights are uniform, can use BFS in linear time
- Otherwise, must use Dijkstra in log-linear time

Computational Complexity

$$H_i(h, v_i) = \sum_{k \in v_i} \text{dist}_{v_i}(h, k)$$

Key computation

Distances from h to all $k \in v_i$

- If edge-weights are uniform, can use BFS in linear time
- Otherwise, must use Dijkstra in log-linear time

Computing centroid

Most computationally complex piece, three options:

- Exhaustive search in $\mathcal{O}(|v_i|^2)$
- Gradient descent in $\mathcal{O}(|v_i| \log |v_i|)$
- Center of mass approximation in $\mathcal{O}(|v_i|)$

Summary

Chief contributions

- Converge to a single centroidal Voronoi partition in finite time
- Coverage control which works in non-convex environments with holes
- Computation can scale well to large areas with many robots

Ongoing Work in Coverage Control

Current directions

- Motion protocol
  - Agents will patrol boundary of territory to meet neighbors
  - Can model need to meet neighbors as tasks on boundary
- Local broadcast communication
  - More realistic model of wireless communication
  - Requires overlapping territories during evolution

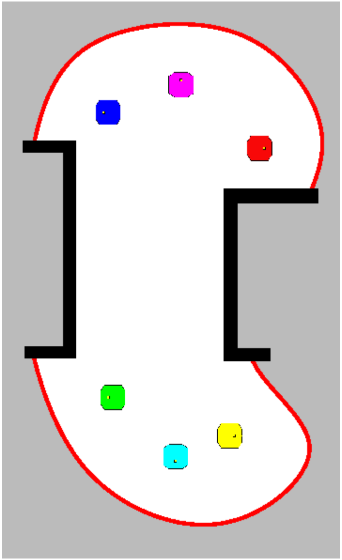
Our Clearing Problem

T34 security bot from tmsuk and Alacom in Japan

The Team: Robots with limited-range sensors

The Mission: Guarantee detection of any evaders in an unknown environment

Exploration Inspiration

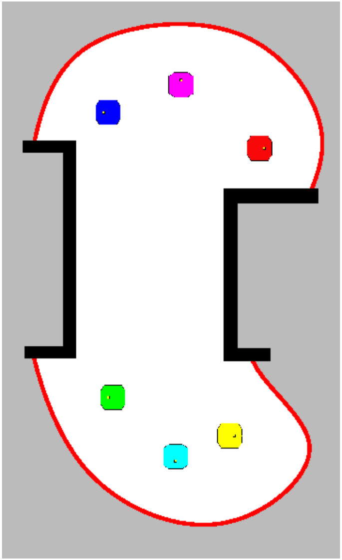


Observation

Clearing an environment is a constrained form of exploration

- For stationary evaders, cleared = explored
- Otherwise, cleared can be recontaminated

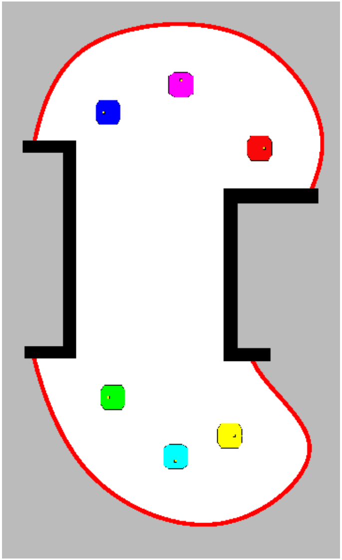
Exploration Inspiration



For exploration

Frontier: Boundary between explored and unexplored areas

Exploration Inspiration



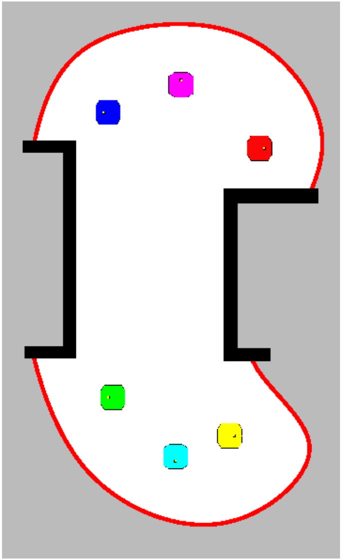
For exploration

Frontier: Boundary between explored and unexplored areas

For pursuit-evasion

Frontier: Boundary between cleared and contaminated areas

Exploration Inspiration



For exploration

Frontier: Boundary between explored and unexplored areas

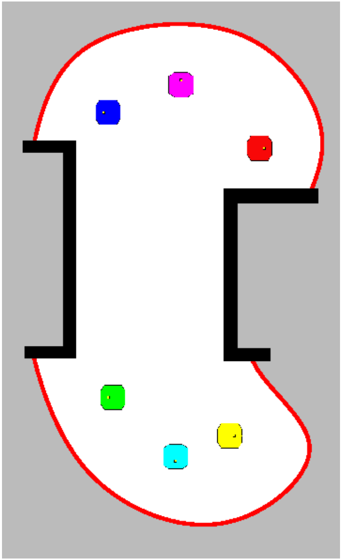
For pursuit-evasion

Frontier: Boundary between cleared and contaminated areas

Our Approach

- Completely cover frontier at all times
- Continuously push back frontier

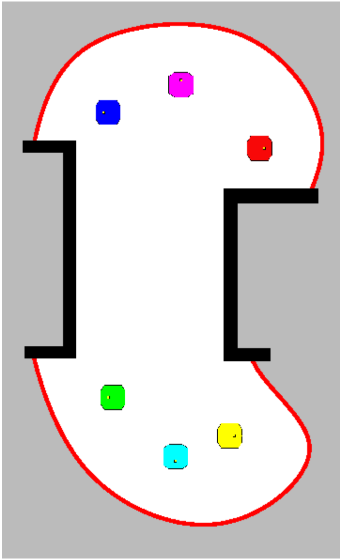
Key Issues



Existing methods for computing global frontier require:

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

Key Issues



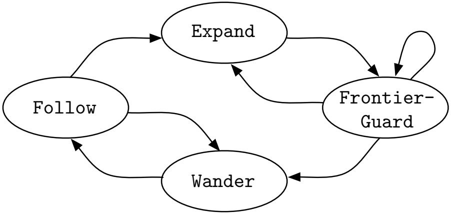
Existing methods for computing global frontier require:

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

Our new method requires:

- Complete coverage of frontier at all times
- Mutual localization between neighboring robots

Distributed Algorithm Roles



Leaders

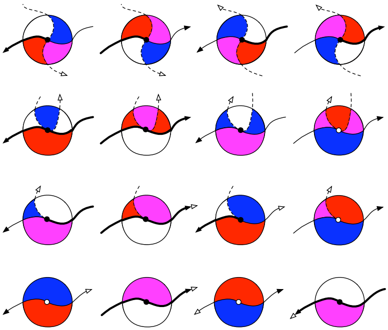
Frontier-Guard: Key role for algorithm. Cover local frontier and dispatch agents to expand it.
Expand: Agent moving to a viewpoint it was assigned.

Non-Leaders

Follow: Waiting for orders from a guard.
Wander: Cleared local area, now searching for a guard to follow.

Distributed Global Frontier

Each frontier-guard stores its local oriented frontier arcs

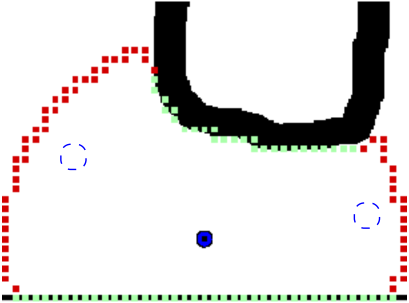


Frontier Updating

When a new guard reaches its viewpoint, it must:

1. Ask for frontier arcs from neighboring guards
2. Inform neighbors of frontier segments inside footprint
3. Classify local frontier based on intersections

Viewpoint Planner



Assumption: Sensor footprints are circular

Goal: Pick new viewpoints V

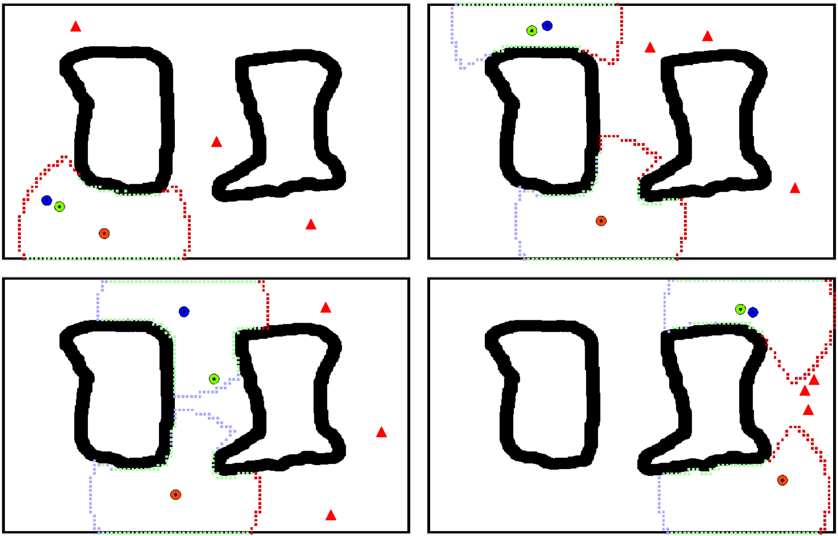
- Minimize $|V|$
- Maximize area exposed

Viewpoints required for angular width Ω of arc:

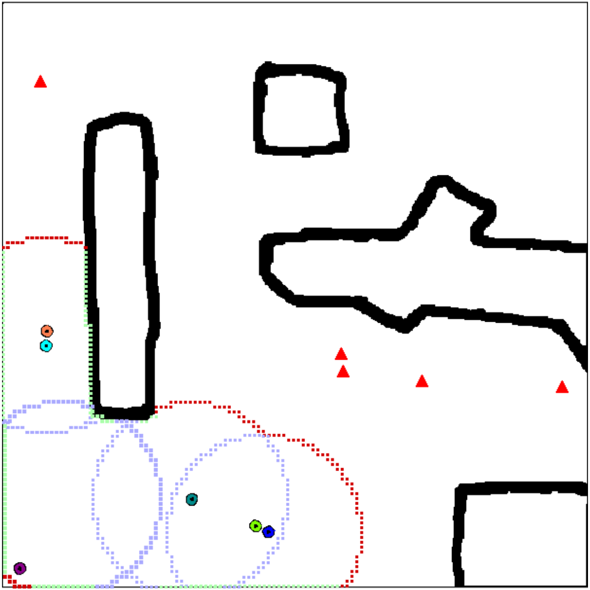
- $\Omega \leq \frac{2\pi}{3}$: $|V| = 1$
- $\Omega = 2\pi$: $|V| = 3$

For intermediate, choice of what to optimize

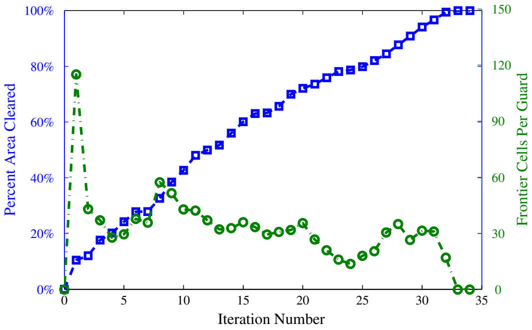
Example Simulation



Movie

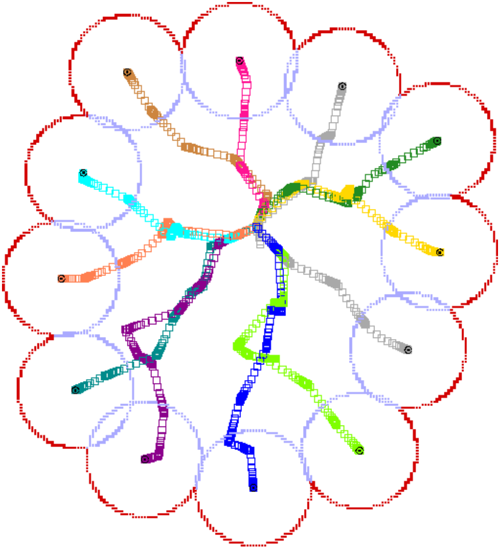


Frontier Coverage



- Frontier cell count per guard does not grow with area cleared
- Distributed storage requires only constant memory per agent

Empty Space



Summary

Chief contributions

- Online clearing algorithm which works in non-convex environments with holes
- Distributed storage and updating of global frontier
- Requires only mutual localization

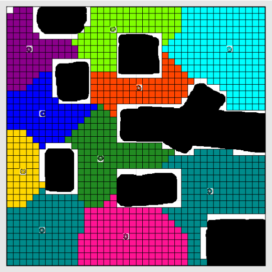
Ongoing Work in Pursuit-Evasion

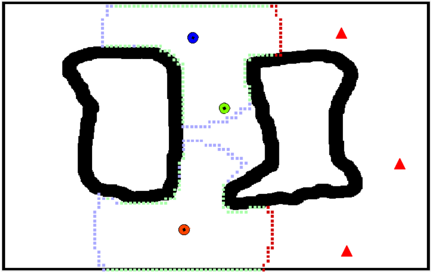
Current directions

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

Conclusion

- Distributed coordination algorithm framework for hardware
- Two parallel algorithm implementations:
  1. Coverage of discretized environments
  2. Frontier-based pursuit-evasion





The End

Questions?