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
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
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
Ocean monitoring gliders from noc.soton.ac.uk, warehouse robots from KIVA Systems,
hopping planetary explores from NASA
Papers in this Talk


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

2 Robotic Network Model

3 Gossip Coverage
   - Problem sketch
   - Current results
   - Future directions

4 Frontier-based Pursuit-Evasion
   - Problem sketch
   - Current results
   - Future directions

5 Conclusion
Hardware
Robotic Network Model

**Differential drive**
- Translational velocity $v$
- Rotational velocity $\dot{\theta}$

**Physical state**
\[ \mathbf{x} = (x, y, \theta) \]
**Differential drive**
- Translational velocity $v$
- Rotational velocity $\dot{\theta}$

**Physical state**
\[ x = (x, y, \theta) \]

In theory state is an integral of velocities
Robotic Network Model

**Robot Model**

**Differential drive**
- Translational velocity $v$
- Rotational velocity $\dot{\theta}$

**Physical state**
$$\mathbf{x} = (x, y, \theta)$$

*In practice* measurement of actual velocities is imperfect, integrals diverge
**Differential drive**

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

**Physical state**

$$\mathbf{x} = (x, y, \theta)$$

In practice measurement of actual velocities is imperfect, integrals diverge

Either must accept position errors or use sensors for localization
Sensor footprint

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

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
Communication Graph

Many possible models for which agents can communicate

Combinations of:
- Network geometry
- Physical proximity
- Current robot roles
- Randomness
Robotic Software Overview

- Control & Communication Algorithm
- Navigation Software
- Player interfaces

To/from other agents
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

Control & Communication Algorithm

Navigation Software

Player interfaces

SND Navigation handles local path planning and execution

Algorithm design can focus on:
- Desired positioning of robot
- Communication for coordination
Evolution of ND+ Nav by J. Mingues, J. Osuna, L. Montano

**Input:** $S$, desired pose $(x, y, \theta)$

**Output:** $v$ and $\dot{\theta}$
Input: $S$, desired pose $(x, y, \theta)$

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

- Find gaps in sensor footprint
**Input:** $S$, desired pose $(x, y, \theta)$

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

- Find gaps in sensor footprint
- Pick best gap to drive towards
**Input:** $S$, desired pose $(x, y, \theta)$

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

- Find gaps in sensor footprint
- Pick best gap to drive towards
- Adjust commands based on nearby obstacles
**Input:** \( S \), desired pose \((x, y, \theta)\)

**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 \texttt{snd} driver in Player/Stage
# Summary of Robotic Network Model

## Algorithm Design Requirements

<table>
<thead>
<tr>
<th>1. Data structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Correct or account for errors in ((x, y, \theta))</td>
</tr>
<tr>
<td>- Processor state (W) and communication alphabet (L)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Update functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Message-generation function</td>
</tr>
<tr>
<td>- Processor state transition function</td>
</tr>
<tr>
<td>- Motion control function to pick desired pose</td>
</tr>
</tbody>
</table>

| 3. Communication graph model |
1. Introduction

2. Robotic Network Model

3. Gossip Coverage
   - Problem sketch
   - Current results
   - Future directions

4. Frontier-based Pursuit-Evasion
   - Problem sketch
   - Current results
   - Future directions

5. Conclusion
**Motivation**

*Biological examples of coverage control*

- Tilapia mossambica, Barlow et al '74
- Sage sparrows, Petersen et al '87
Lloyd’s Algorithm

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

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

Problem sketch

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 $p = \{p_1, \ldots, p_N\}$
- partition environment into $v = \{v_1, \ldots, v_N\}$
- 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
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

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

**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
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

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

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

Minimize expected distance between random vertex and closest robot
Hardware Experiment
Hardware Experiment
Hardware Experiment

Gossip Coverage
Current results

Joseph Durham (UCSB)
Distributed Coordination
Feb 5 23 / 45
Hardware Experiment

Joseph Durham (UCSB)
Hardware Experiment
Hardware Experiment
Hardware Experiment

Gossip Coverage

Current results
**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 $T : X \rightarrow X$

**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
1 Introduction

2 Robotic Network Model

3 Gossip Coverage
   • Problem sketch
   • Current results
   • Future directions

4 Frontier-based Pursuit-Evasion
   • Problem sketch
   • Current results
   • Future directions

5 Conclusion
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
**For exploration**
Frontier: Boundary between explored and unexplored areas

**For pursuit-evasion**
Frontier: Boundary between cleared and contaminated areas
**Problem sketch**

**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
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.
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
**Assumption:** Sensor footprints are circular

**Goal:** Pick new viewpoints $V$

- Minimize $|V|$
- Maximize area exposed

Viewpoints required for angular width $\Omega$ of arc:

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

For intermediate, choice of what to optimize
Example Simulation
- 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
Current directions

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

Two parallel algorithm implementations:
1. Coverage of discretized environments
2. Frontier-based pursuit-evasion
Questions?