IMPROVED COVERAGE CONTROL USING ONLY LOCAL INFORMATION

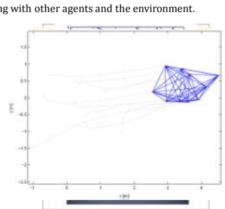
Jay Wagenpfeil, Adrian Trachte

Overview

Motivation and Tasks

3

- Multi-Agent System (MAS):
  Number of autonomous agents sensing and interacting with other agents and the environment.
- Cooperative tasks:
  - Rendezvous [1] [2] [3]:
    all agents rendezvous at an arbitrary point
  - Deployment [1] [4]:
    achieve maximum deployment of agents in environment
  - Coverage [1] [4] [5]:
    achieve maximum coverage of regions of interest
  - Flocking [6] [7] [8]
    motion coordination in a synchronized manner and obstacle avoidance

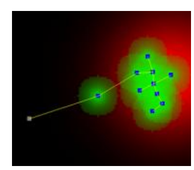


[1] Martinez et al - Motion Coordination with Distributed Information - 2007 / ... agents - 2003
[2] Cortés et al - Spatially-distributed coverage optimization and control with limited-range interactions - 2004 / ... partitions - 2006
[8] ... Robots - 2006

Basic Setup [9]

4

- Basic Setup:
  - Coverage of Regions of Interest:
    Agents can sense information given by the environment, e.g. temperature gradients.
  - Base:
    A Base is introduced as a central unit which processes the information obtained by the agents.
  - Communication:
    Communication cost is introduced. Agents act as a relay to transfer the information obtained by other agents to the base.



Goal:
Gain coverage of the areas with high information density while keeping the power consumption low.

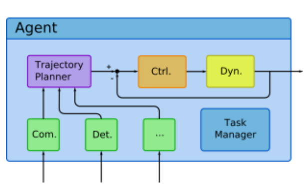
[9] Li et al – **Distributed Cooperative Coverage Control of Sensor Networks - 2005**

Modularization

5

- Divide agent in certain functional parts ... and control them: / For different control tasks it is necessary to have different trajectory planning, different controllers or even a different sensor behaviour.
  - Agent Dynamics
  - Environment detector
  - Communicator
  - Trajectory Computation
  - Controller

Task Manager



Multi-Agent Simulator MASIM

Multi-Agent Simulator MASIM

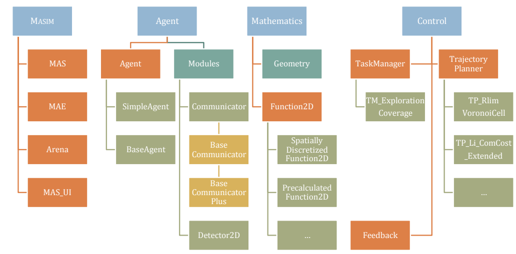
7

- Need for a tool to simulate multi-agent behavior without restrictions
- Self-programmed Framework
  - Implemented in JAVA
  - Modular design using OO-programming techniques
- Based on MASON [10]
  - Multi-agent simulation core library
  - Provides basic visualization
  - Free availability

[10] http://www.cs.gmu.edu/~eclab/projects/mason/

Class Overview

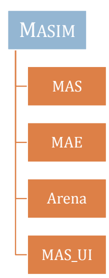
8



Package Overview: MASIM

9

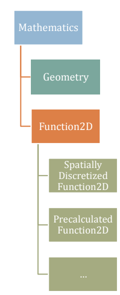
- Class MAS
  - Initialization, Scheduling
- Class MAE
  - Encapsules the environment, density function, mission space, etc …
- Class Arena
  - Geometry of the mission space
- Class MAS_UI
  - Visualization, GUI



Package Overview: Mathematics

10

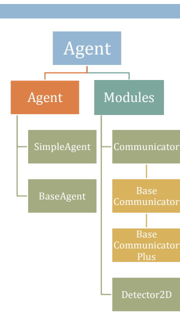
- Package for mathematic computations
- Sub-Package Geometry:
  - Classes for geometric computations, e.g.
    - r-limited voronoi cells
- Class Function2D
  - Encapsules functions f : R^2->R
  - Subclasses implement functionality for
    - spatially discretized grids,
    - precalculated function values and
    - other special types of functions



Package Overview: Agent

11

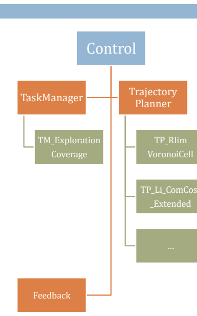
- Class Agent
  - Implements basic functionality
  - Sub-classes:
    - SimpleAgent: Simple Dynamics
    - BaseAgent: Specialized agent
- Sub-package Modules:
  - Non control-related modules
  - Class Communicator
  - Class Detector2D



Package Overview: Control

12

- Control-related modules of the agent:
  - Class TaskManager
  - Class TrajectoryPlanner
  - Class Feedback
- Subclasses implement specialized functionality



Multi-agent simulator MASIM

Class Overview
General Structure
Modularization example: Communicator

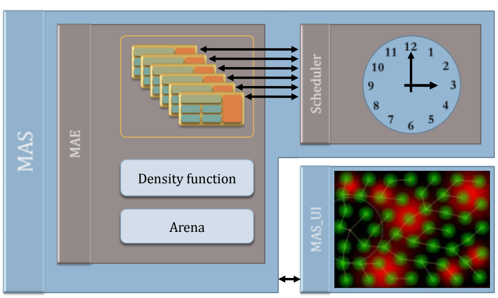
Functionality of the Simulation

14

- So far hierarchical relation of classes.
- But, how does the simulation work?
  - How is the simulation built from these classes?
  - Functional relation between classes.
- Address the issue in two steps:
  - General structure of the simulator:
    - Which classes are required to build a simulation environment for the agents?
  - Modular structure of the agents:
    - Which classes model the functionality of the agents?

General Structure of the Simulator

15



Modular Structure of the Agents

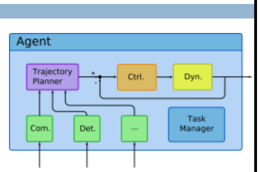
16

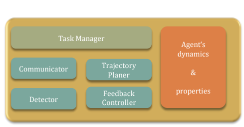
- From before:
  - Agents' functionality is divided into modules.
  - This can as well be seen in the software implementation
- Class Agent:
  - Very basic functionality
  - Container for modules
- Subclasses extend functionality beyond modularization

Modular Structure of the Agents

17

- Communication
  - Interagent communication
- Detector
- Trajectory planning
  - Different motion behaviors.
- Feedback control
- Task managing
  - Select proper modules to achieve desired tasks





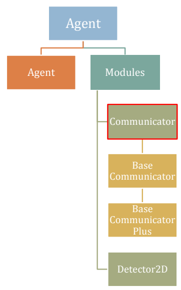
Multi-agent simulator MASIM

Class Overview
General Structure
Modularization example: Communicator

Example Module: Communicator

19

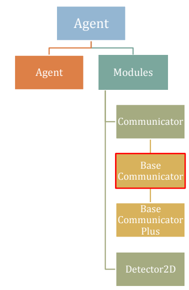
- Communicator
  - Basic Properties
    - Owned by agent
    - Communication range
  - Basic Methods
    - Constructors



Example Module: Communicator

20

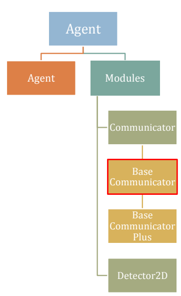
- Base Communicator
  - Important Properties
    - Neighbors in com-range
    - Communication Costs
      - To Base
      - Amount of data to be transferred
        - Data of agent's detector
        - Data to be relayed
      - Cost Function



Example Module: Communicator

21

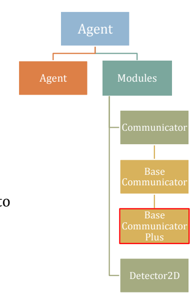
- Base Communicator
  - Important methods
    - Compute communication cost to neighbors and base
    - Update communication neighbors
    - Find shortest path to base
      - Depends on neighbors' communication cost to base
      - Loop avoiding



Example Module: Communicator

22

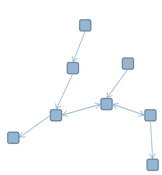
- Base Communicator Plus
  - Important Properties
    - Validation flags
  - Important Methods
    - Search path to base
      - Improved robustness against link failures
      - Enables Agent to find a com-path to base after loss of com-neighbor



Example Module: Communicator

23

- Base Communicator Plus
  - Algorithm:
    1. Connection to base via relaying neighbor is checked.
    2. If connection is lost, agent is invalid and sets all com. neighbors to invalid as well.
    3. Invalid agents search for a new connection to the base by searching for valid agents.
    4. If a new connection to the base is found, all connected neighbors are set valid again.



Theoretical Work and Simulation Results

Problem Formulation
Keep-together function
Exploration
Combining Tasks

Mission Space and Sensor Model

25

- Mission Space:
  - 2-D plane: $\Omega \subset \square^{2}$
  - Event density Function: $R(x), x \in \Omega$
  - Agent position: s_i and $s = (s_i, \ldots, s_N)$
- Detector Model:
  the probability to detect an event depends on:
  - Distance to position, where event takes place, as signal strength declines.

$$p_i(x) = p_{0i} e^{\lambda_i \| x - s_i \|}$$

Coverage Control

26

- Goal is to maximize the expected event detection probability over the mission space

$$P(x,s) = 1 - \prod_{i=1}^{N} [1 - p_i(x)]$$ Probability that an event is detected.

$$F(s) = \int_{\Omega} R(x) P(x,s) dx$$ Value function

⇨ $\max_{(s_1,\ldots,s_N)} F(s)$

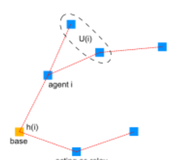
Communication

27

- Communication between agents:
  A function describing the communication cost to receive and forward data, increases monotonically with increasing distance to target agent.

$$e(d) = \alpha_1 + \alpha_2 d^n$$ Energy for transmitting one bit data over a distance d

- Communication to base:
  Computing the shortest path to the base via a routing protocol.
  - Downstream neighbor: h_i
    is the next agent in the shortest path to the base
  - Upstream neighbors: U_i
    is a set of neighbors, which use agent *i* to communicate with the base



Communication Cost

28

- Goal is to minimize the communication cost

c_i Overall power consumption to transfer one bit of data from agent *i* to the base.

$$\delta_i(s_i) = \alpha_3 \int_{\Omega} R(x) p_i(x) dx$$ Data rate originated by the *i*'th agent.

$$G(s) = \sum_{i=1}^{N} c_i \delta_i(s_i)$$ Communication Cost Function

⇨ $\min_{(s_1,\ldots,s_N)} G(s)$

Solving the Optimization Problem

29

- Maximize F(s) and minimize G(s)

⇨ $\max_{(s_1,\ldots,s_N)} J(s)$ with $J(s) = \omega_1 F(s) - \omega_2 G(s)$

- Optimization via partial derivatives $\frac{\partial J}{\partial s_i}$

⇨ $\dot{r}_i = \omega_1 \frac{\partial F}{\partial s_i} - \omega_2 \frac{\partial G}{\partial s_i} =: v_i$ reference trajectory

- Under certain assumptions it is possible to approximate these derivatives with only local information. [9]

[9] Li et al – **Distributed Cooperative Coverage Control of Sensor Networks - 2005**

Summary

30

- Formulated coverage control problem via value function *F(s)*
- Introduced communication cost via cost function *G(s)*
- Optimization of *J(s)* via partial derivatives
- Certain assumptions

⇨ Agent movement with only local information

Theoretical Work and Simulation Results

Problem Formulation
Keep-together function
Exploration
Combining Tasks

Keep Together Function: Motivation

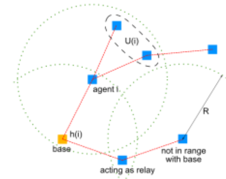
32

- Limited communication range

  Wireless communication is restricted to certain distances within which a reliable communication is possible.

- Communication from agents to base is necessary

  To transmit the sensed information to the base it is necessary that every agent stays connected to its neighbors



⇨ Artificial potential function to keep the agents connected

Keep Together Function: Properties

33

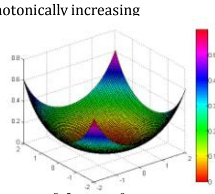
- Potential function in dependence of distance *d*

  $d = \|a - b\|$ Distance from point *a* to point *b*

- Properties of function *f*:
  - $f(d)$ is continous and monotonically increasing
  - $f(0) = 0$
  - $\lim_{d \to R} f(d) = +\infty$

- Example:

  $$f(d) = \frac{d}{(R-d)^n}$$



R= 5 n=2

Application of Keep Together Function

34

- It is necessary to stay in contact with the downstream neighbor and all upstream neighbors.

⇨ Distances to down- and upstream neighbors have to be smaller than R

- Notation

  $d_i = \|s_{h_i} - s_i\|$ Distance from agent *i* to its downstream neighbor h_i

  $f(d_i) = f_i$ Function *f* which depends of d_i is denoted as f_i

---

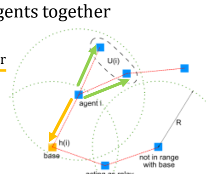
Application of Keep Together Function

35

- Formulation of gradients to keep agents together

  $\left(\frac{\partial f_i}{\partial s_i}\right)^T$ KT-gradient to downstream neighbor

  $-\sum_{j \in U_i} \left(\frac{\partial f_j}{\partial s_i}\right)^T$ KT-gradient to upstream neighbors



- Reference Trajectory

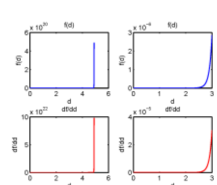
⇨ $$\dot{r}_i = \left(\frac{\partial f_i}{\partial s_i}\right)^T - \sum_{j \in U_i} \left(\frac{\partial f_j}{\partial s_i}\right)^T + v_i$$

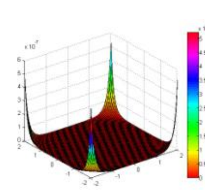
Designing the Keep Together Function

36

- Choose Communication Range *R*
- Keep Together Function should not override other gradients if neighbors are sufficiently close

⇨ Design function f close to zero for d < R and f→∞ for d→R



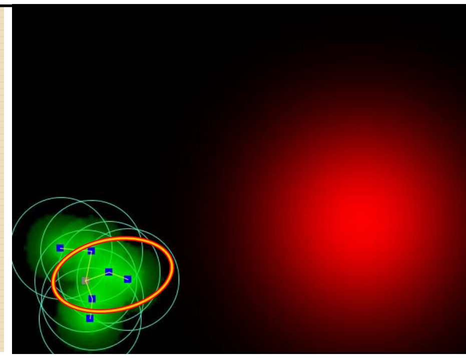


R= 5 n=20

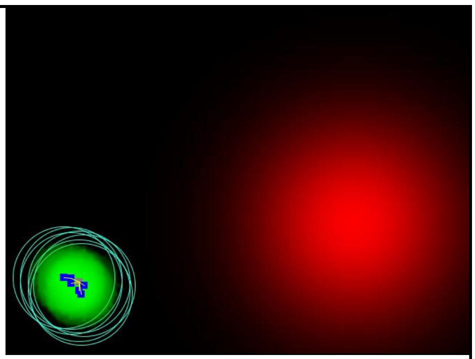
Keep Together Function

37

- Now compare behavior:
  - Optimizing only coverage and communication cost, no Keep-Together Function
  - Additionally considering Keep-Together function



38 Density Function Gradient and Communication Cost



39 With Keep Together Function

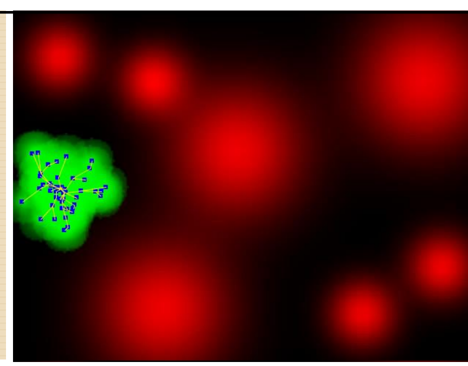
Theoretical Work and Simulation Results

Problem Formulation
Keep-together function
Exploration
Combining Tasks

Review Coverage Control

41

- Coverage control:
  - Maximizing the probability of detecting events.
  - ➡ Most important areas of the mission space are well covered.
- Problem:
  - Areas of the mission space may be „hidden" from the agents.
  - ➡ Not all important areas are covered.

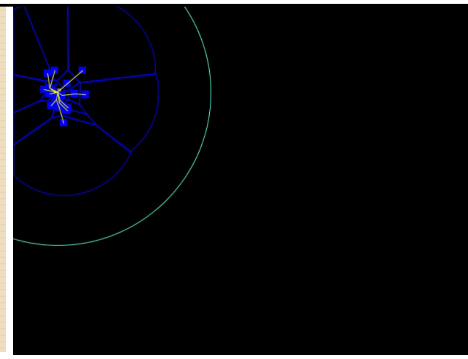


42 Mission space with „hidden" areas

Algorithm for Exploration

43

- Exploring the mission space:
  - Use a deployment algorithm
  - Maximize the area covered by all agents.
- Algorithm uses r-limited Voronoi Cells
  - Moving towards centroid of r-limited Voronoi Cell
  - Radius is given by communication range.
  - Using only local information.



44

Exploration using r-limited Voronoi Cells

Theoretical Work and Simulation Results

Problem Formulation
Keep-together function
Exploration
Combining Tasks

Combining Tasks: Motivation

46

- Now we have two tasks:
  - Covering the most important areas of the mission space to maximize the probability of detecting events.
  - Exploring the complete mission space by maximizing the area covered by all agents.
- Idea: Combine both tasks:
  - First explore the mission space.
  - Then cover the most important areas.
- ➔ Enables the agents to cover areas unreachable if only using coverage control.

Combining Tasks: Motivation

47

- Extended Setup:
  - Switch task when the whole misson space has been explored.
- Problem:
  - How does each agent know, that the whole mission space has been explored?
  - Agents only possess local information.
  - Information about all agents / whole mission space is needed.

Taskswitch via Consensus

48

- Consensus is the state where all agents in the network achieve agreement.
  - In this case, agreement on switching the task.
  - This is equivalent to agreement that the complete mission space is explored.
- Mission space is explored when all agents stop moving.
  - Implement an agreement protocol based on movement of agents.

Taskswitch via Consensus

49

- Add two variables
  - λ_i gives the state of agent i
    - $\lambda_i = 0$ means agent i is moving
    - $\lambda_i = 1$ means agent i has stopped
  - Λ_i is the consensus state of agent i

$$\Lambda_i = \frac{1}{1+|\mathcal{N}_i|}\left(\lambda_i + \sum_{\mathcal{N}_i} \Lambda_j\right)$$

  - $\mathcal{N}_i$ is the set of neighbors of agent i
  - $|\mathcal{N}_i|$ is the number of neighbors of agent i

Taskswitch via Consensus

50

- In every step:
  - First update state λ_i.
  - Then update Λ_i according to the given formula.
- If the mission space has not completely been explored:
  - There are moving agents with state $\lambda_i = 0$
  - For the consensus variable holds $\Lambda_i < 1$.
- If all agents have stopped:
  - $\Lambda_i \to 1$ for each agent.

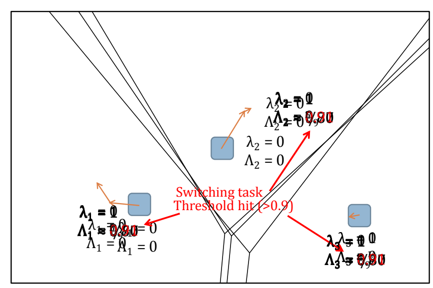
Taskswitch via Consensus

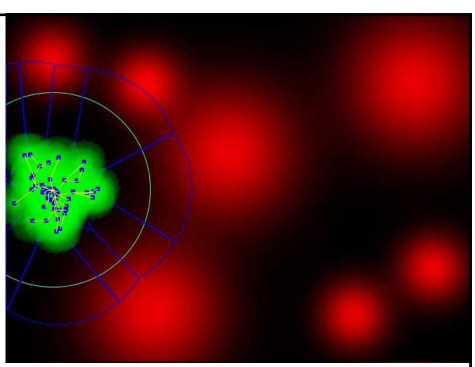
51

- If $\Lambda_i > (1-\varepsilon)$, with $\varepsilon \ll 1$, then
  - Set $\Lambda_i = 1$
  - Switch to coverage task
- Ideally, all agents hit the threshold at the same time.
  - In reality, there is always a small delay between the first and the last agent to hit the threshold.
  - Add a dead-time to each agent before it starts moving according to the coverage task.

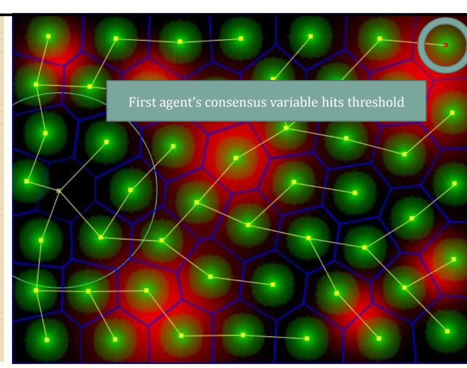
Consensus with ε = 0.1

52

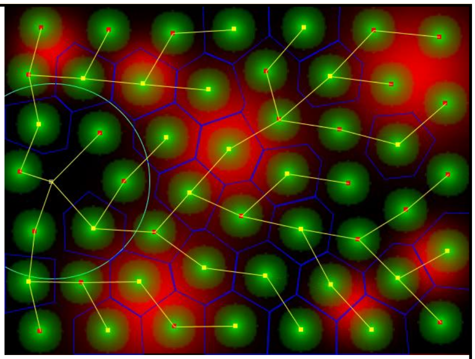




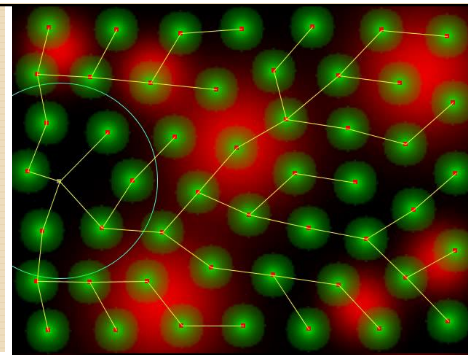
53 Simulation Results



54 Simulation Results



55 Simulation Results



56 Simulation Results

Conclusion

Summary

Outlook

Summary

58

- Extendable and versatile simulation environment for Multi-Agent Systems
- Analysis of joint detection probability coverage algorithms and enhanced functionality
  - Keep-Together Function
- Combining different control tasks for improved behavior
  - Realizing a consensus algorithm for task switching

All Algorithms use only local information

Outlook

59

- Agents
  - Specialized agents, e.g. communication agents
- Modules
  - Communicator
    - Anisotropic Communicator
    - Improved routing protocols
  - Detector
    - Anisotropic Detector
    - Combination of different detectors and sensing tasks
- Controller
  - Different controller techniques
  - Convergence for discrete controller

Outlook

60

- Environment
  - A time dependent density function for event probability
  - Non-rectangular arenas
- Simulator
  - Asynchronity
- Theoretical Work
  - Proof the functionality of the Keep-Together Function

THE END

Thank you for your attention

- Consensus mit agent failure
- Paper iman, cortes
- Relay agent failure -> agents lost in space
- Dead-time for task switch