Distributed Localization and Mapping with a Robotic Swarm

İhsan Ecemiş
Icosystem Corporation

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHC030042.

Credits:

Icosystem:

Joe Rothermich, Paolo Gaudiano, Carl Anderson, Joe D'Angelo

iRobot

HRL

USC

Agenda

Mission & Project Background

Software Tools

Hardware Platform

Algorithms

Results

DARPA's Vision (D. Gage/SDR)

Building Clearing Mission

Drop robots into unknown building

Robots explore and build map

Identify “item of interest”

Detect intruders

Functional Requirements

SDR Mission Robot Tasks:

- Navigation
  - wall following, collision avoidance, …
- Localization
- Mapping
- Intrusion Detection

Why Swarms?

Swarm Robotics

- Robustness
- Scalability
- Low cost
- Portability

Difficult to control a single robot: how do you control a swarm?

Icosystem's Role

Develop swarm control software

Design distributed control algorithms

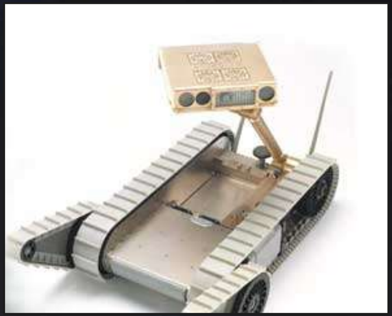
Implement in simulation

Transfer to *Swarmbot* platform

Collaborators: iRobot

Founded in 1991 by Rod Brooks, Colin Angle, and Helen Greiner

Develops industrial, consumer, military and research robots



Collaborators: HRL

Owned by The Boeing Company, General Motors and Raytheon Company

Applied research in the electronics & information sciences

Creates new products and services for space, telecommunications, defense & automotive applications.

Software Tools

Player/Stage (USC & HRL)

Open source platform for experiments in mobile robotics and sensor networks

Player:

- Provides network access to robots and sensors
- Includes **device drivers for robot hardware**

Stage:

- **Simulates** large populations of robots & sensors
- Defines custom robots with parameterized devices

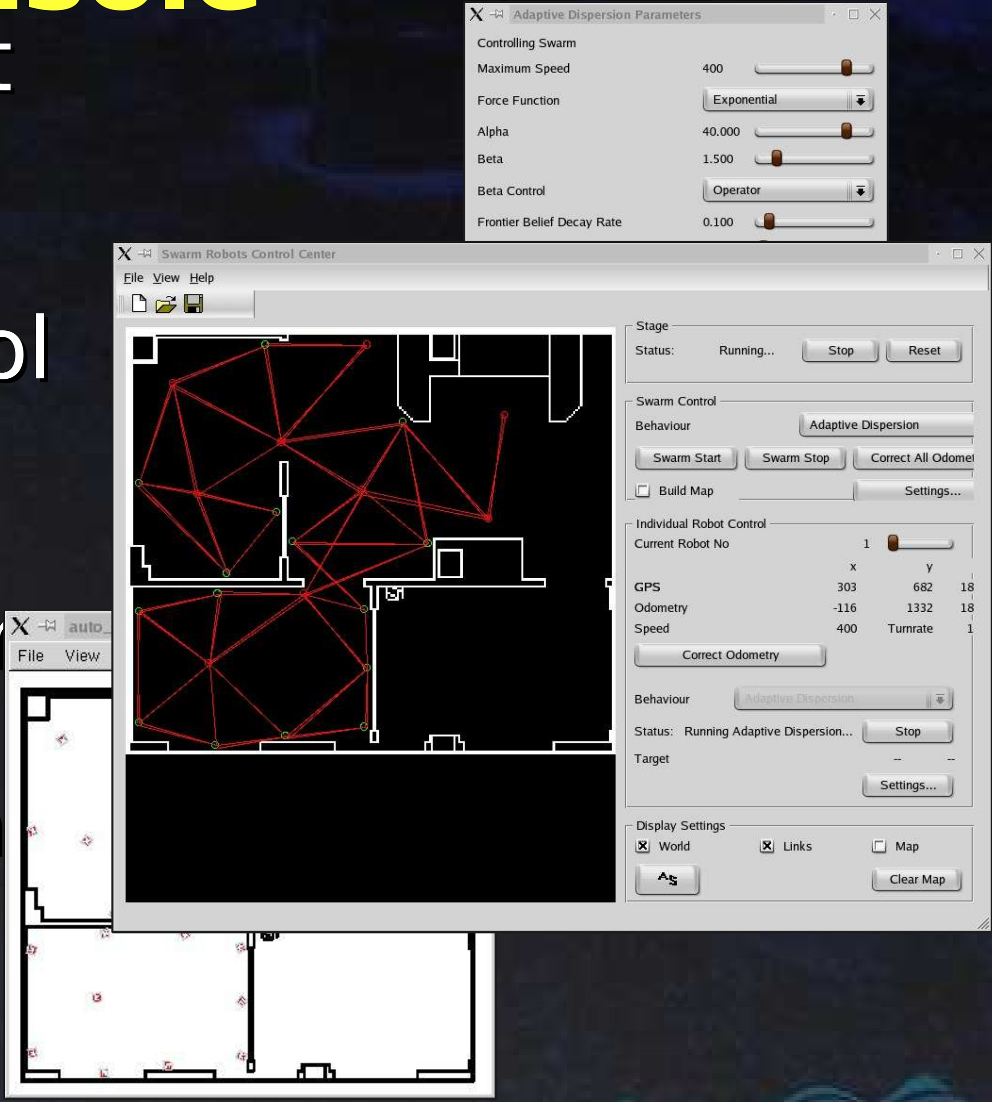
Swarm Operator Control Console

Player/Stage client

Single-robot or swarm-level control tool

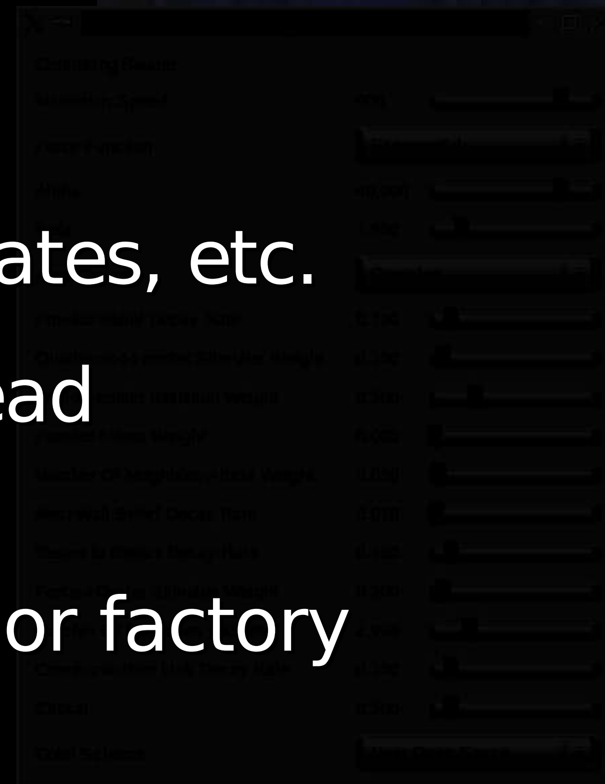
Library of behavior

Real-time or off-lin
Visualization tool



SOCC Features

- C++ (>25K lines), Qt GUI
- Visualization of mapping, communications links, robot states, etc.
- Self-generating code, auto thread handling / window creation
- Highly modular, flexible behavior factory
- Hardware interface



Hardware Platform

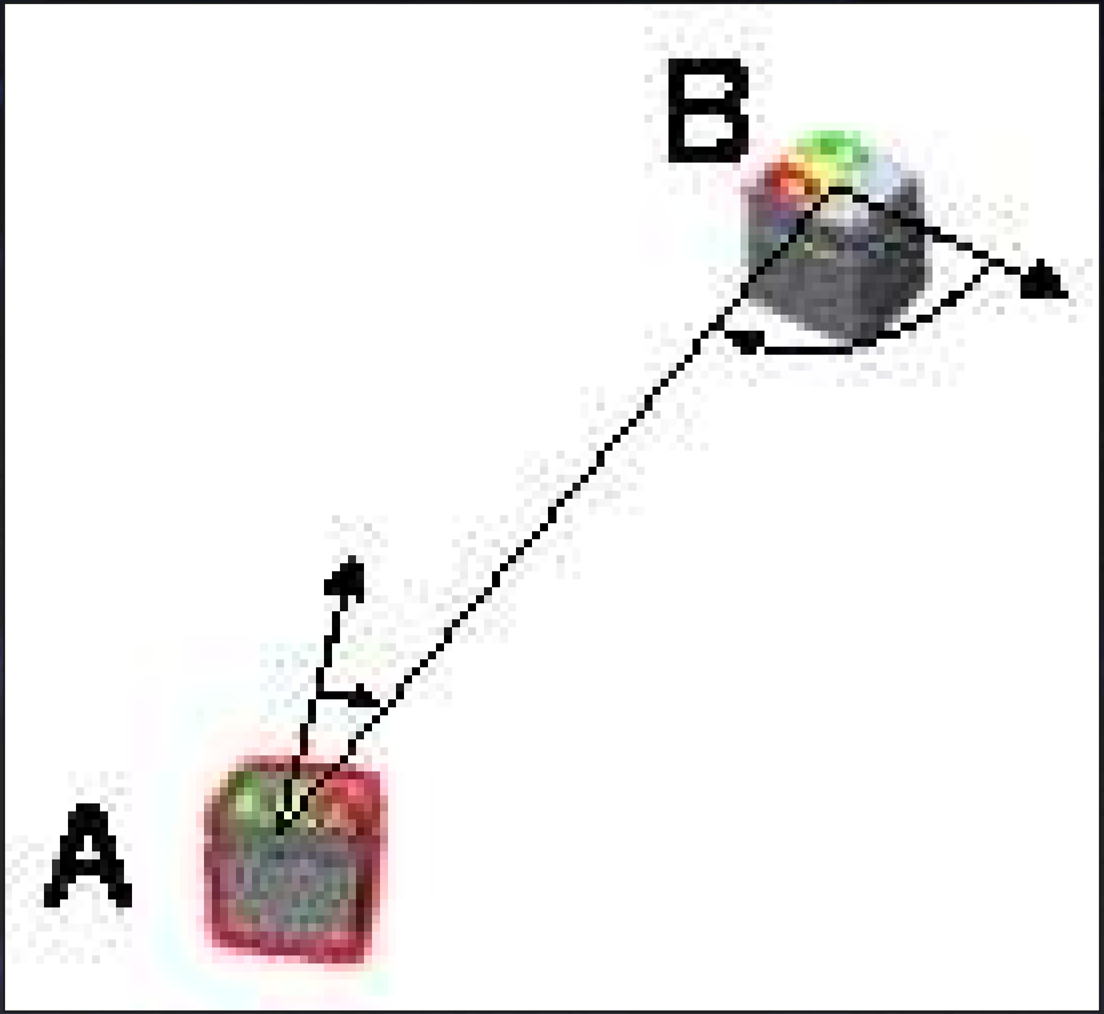
iRobot's *Swarmbots*

- Small footprint (12x12x13cm)
- Bump skirt (8 corner sensors)
- IR comms allow for:
  - Reciprocal localization
  - Local communications
  - Pheromone comms
  - "Radar"

Reciprocal localization

- LEDs and sound for *swarmish* comms

Reciprocal Localization



Hardware Constraints

Limited Memory: 64KB (OS/code) + 512KB data

Low IR Comms Bandwidth: 60 bytes/sec

No radio communication

Cross-compiler limitations

Tight behavior cycle: 25msec

Sensory limitations and noise

Sensory Limitations

No range to obstacles

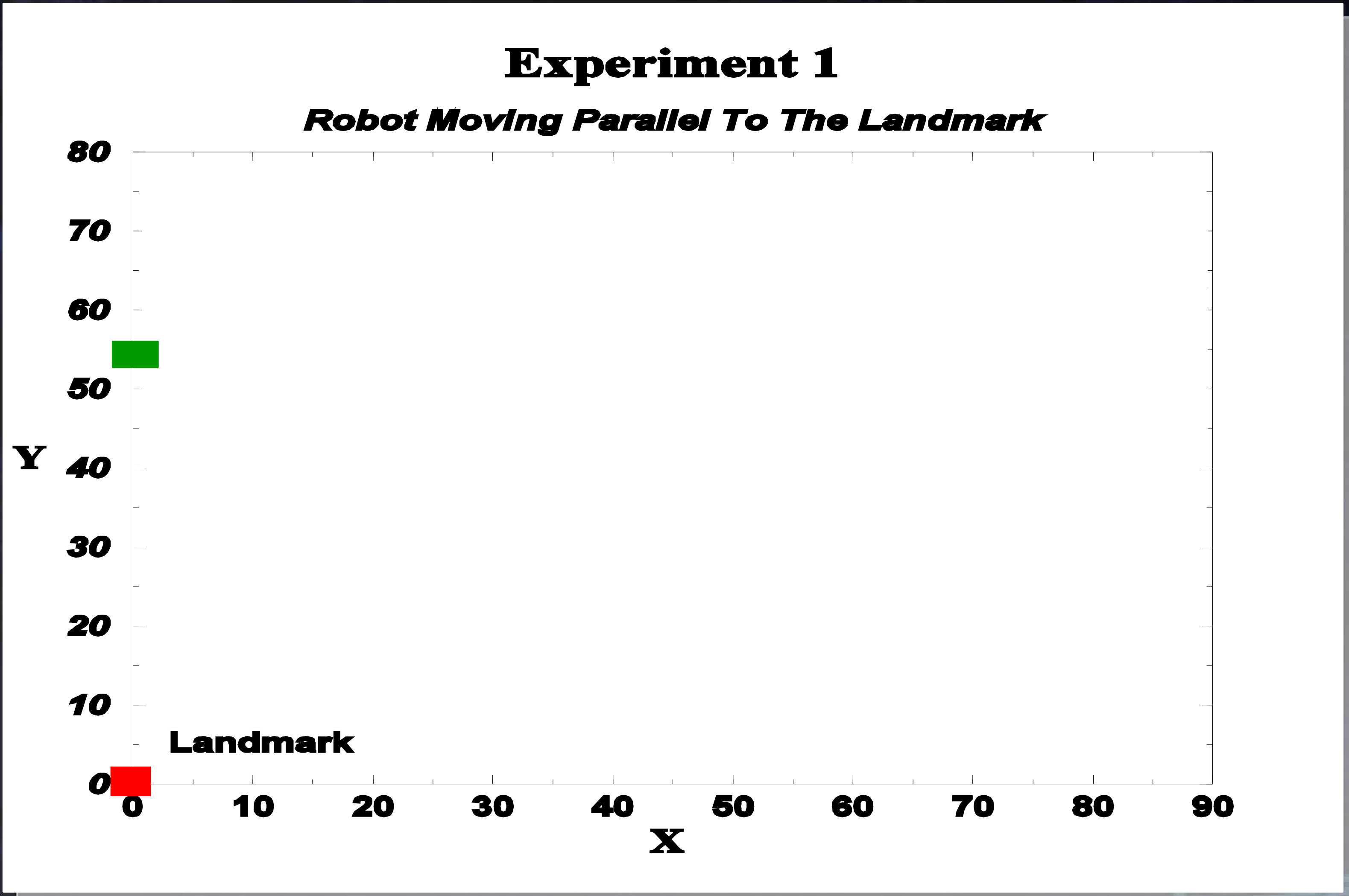
IR range to other bots (<1m)

Non-uniform, noisy localization

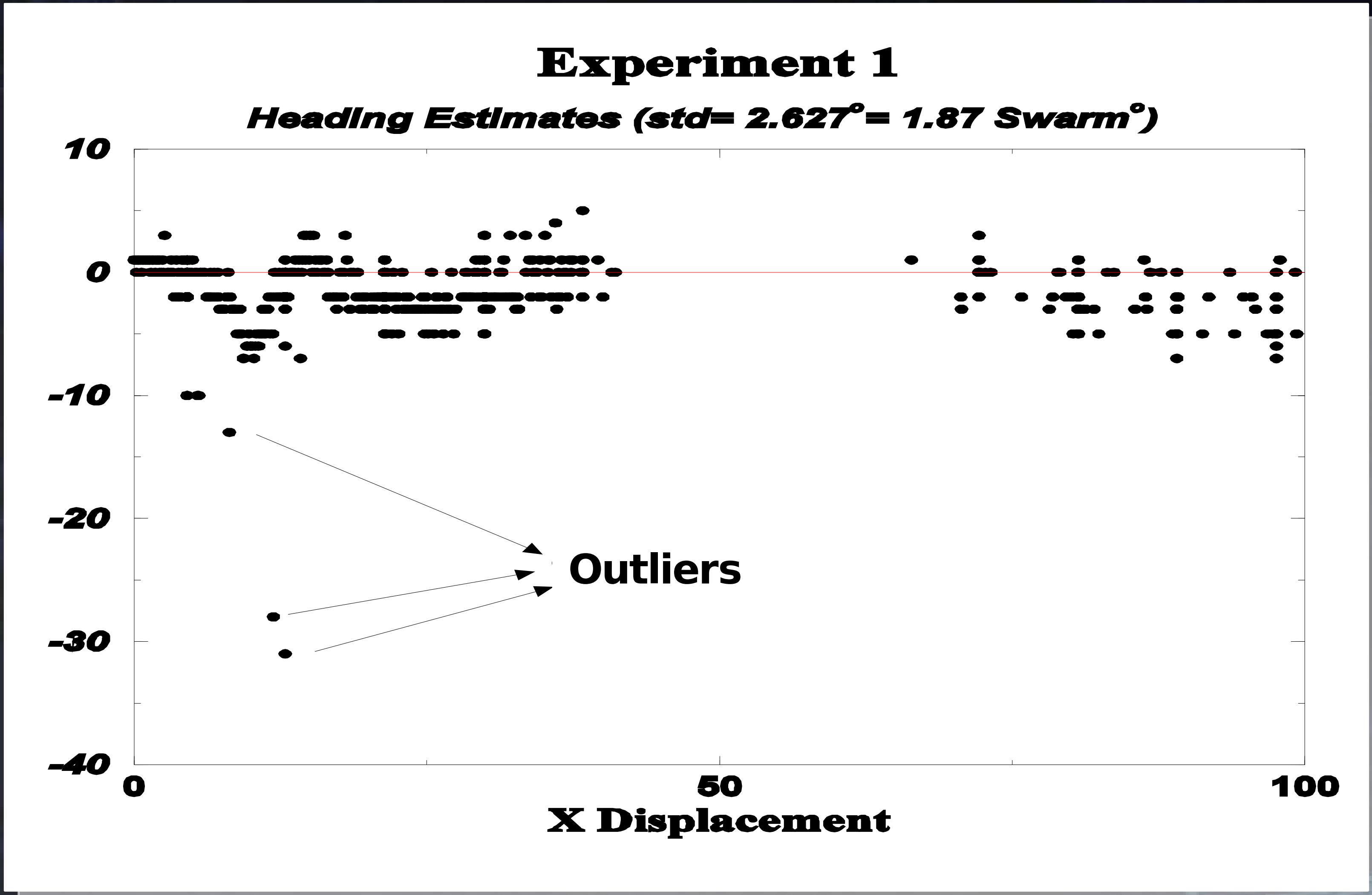
Corrupted data packets (no handshake)

These are "limitations" only in the context of high-accuracy tasks such as localization and mapping

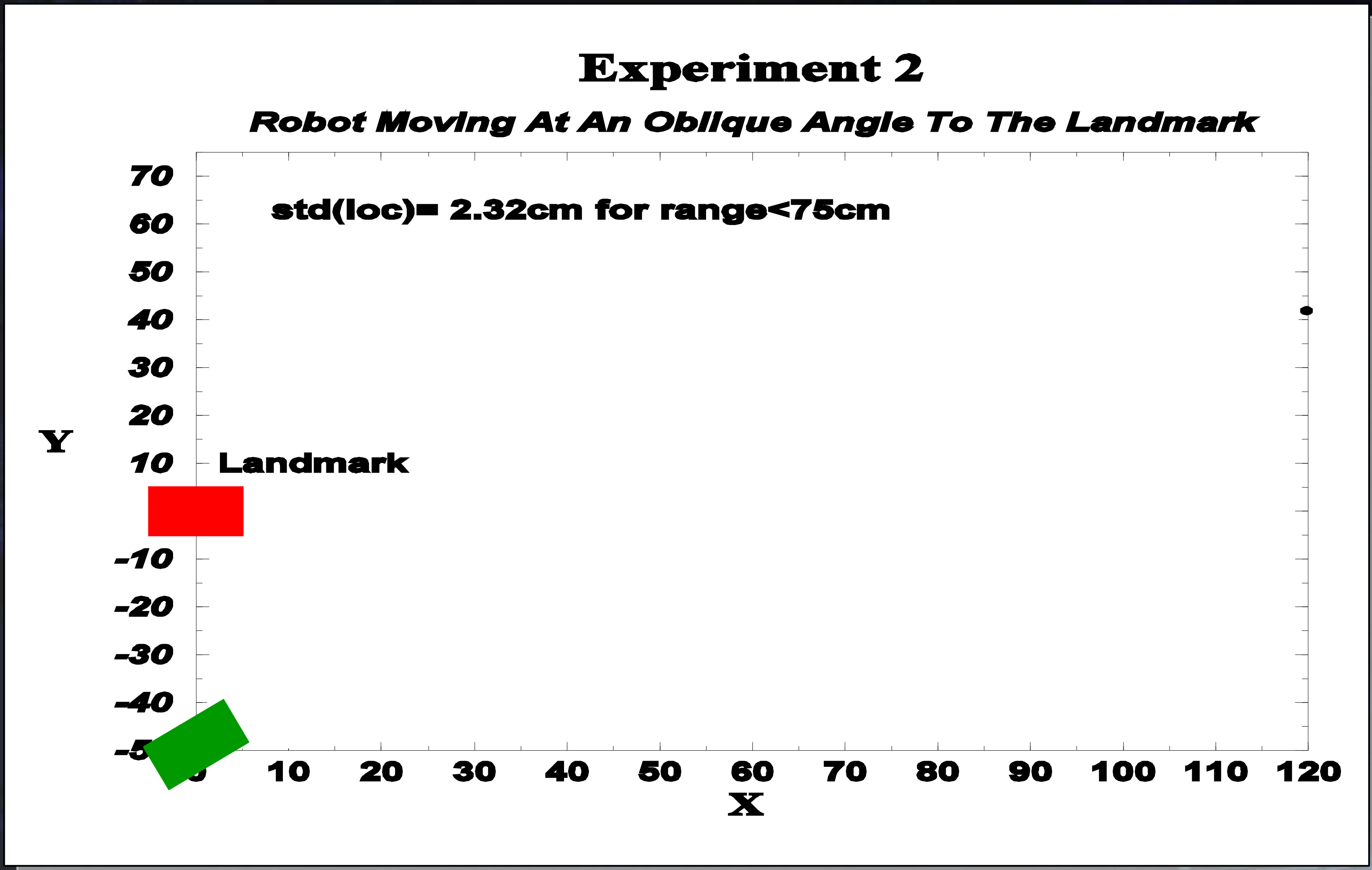
Sensory Noise Experiments



Sensory Noise Experiments



Sensory Noise Experiments



Algorithms

Combining Behaviors

- SDR Mission Robot Tasks:
  - Navigation
    - wall following, collision avoidance, …
  - Localization
  - Mapping
  - Intrusion Detection

Implement each behavior on a single robot

Combine behaviors on a single robot

Implement combined behaviors on swarm

Navigation Behaviors

Waypoint navigation

Dispersion

Robot avoidance

Wall following

Breadcrumbs

Adaptive Dispersion

- Use IR to locate neighbors
- Analog function for attraction/repulsion
- One-parameter dispersion control
- Smooth, scalable functionality

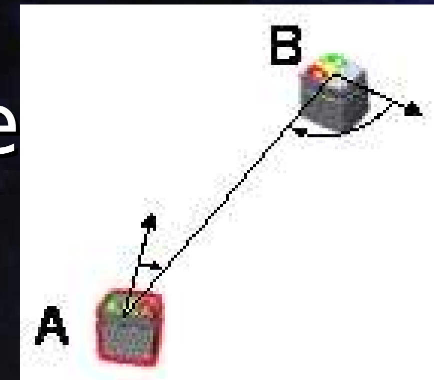
Swarm Behaviors

Collaborative Localization

Dynamic Task Allocation

Swarm Mapping

Collaborative Localization



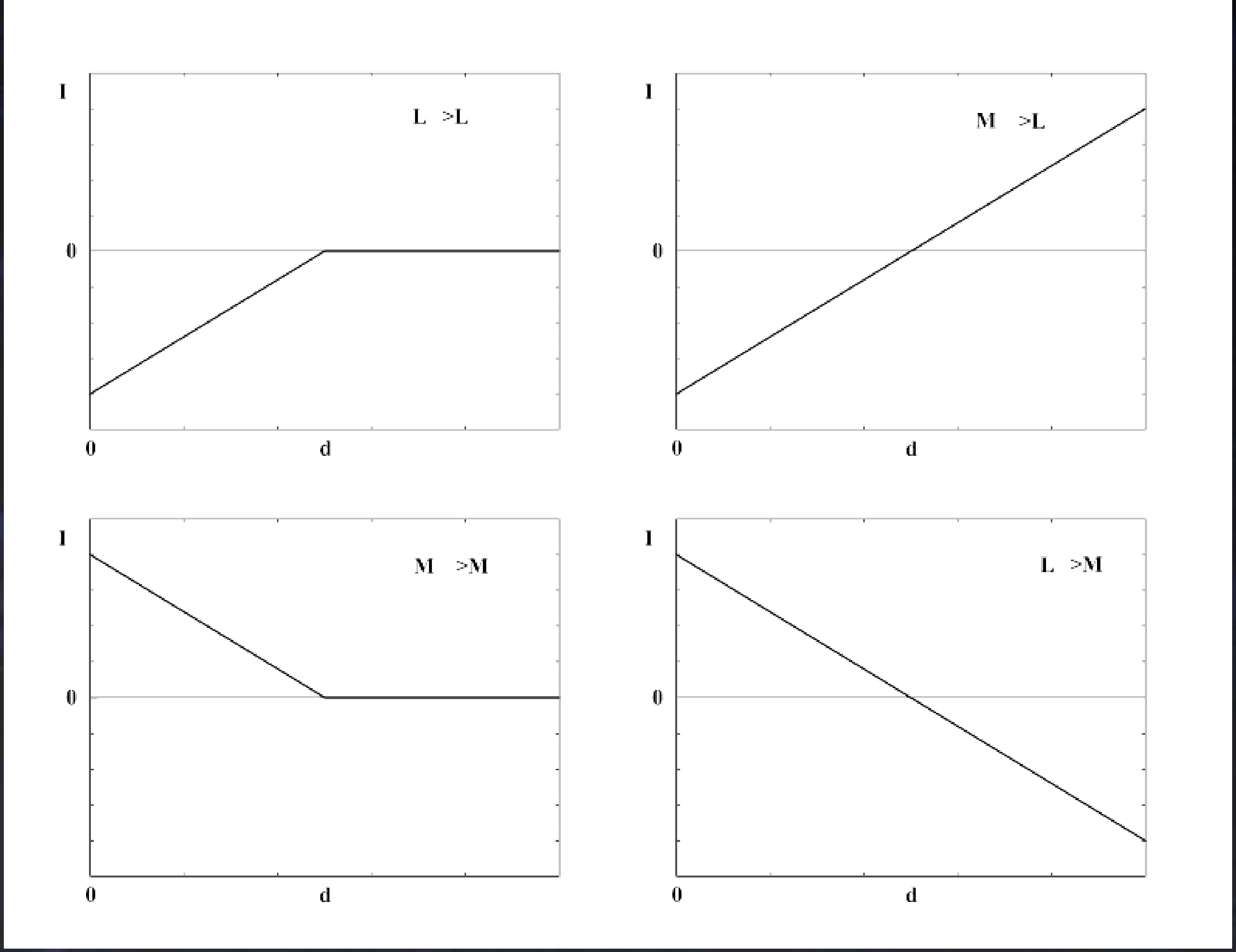
- Use your neighbors to compute estimated position and confidence
- Combine estimates with odometry
- Keep one or more robots as *landmarks* to provide accurate localization†
- Use principles of *task allocation* for dynamic selection of

†Related to work by Rekleitis *et al.*, Kurazume & Hirosi, but with many robots

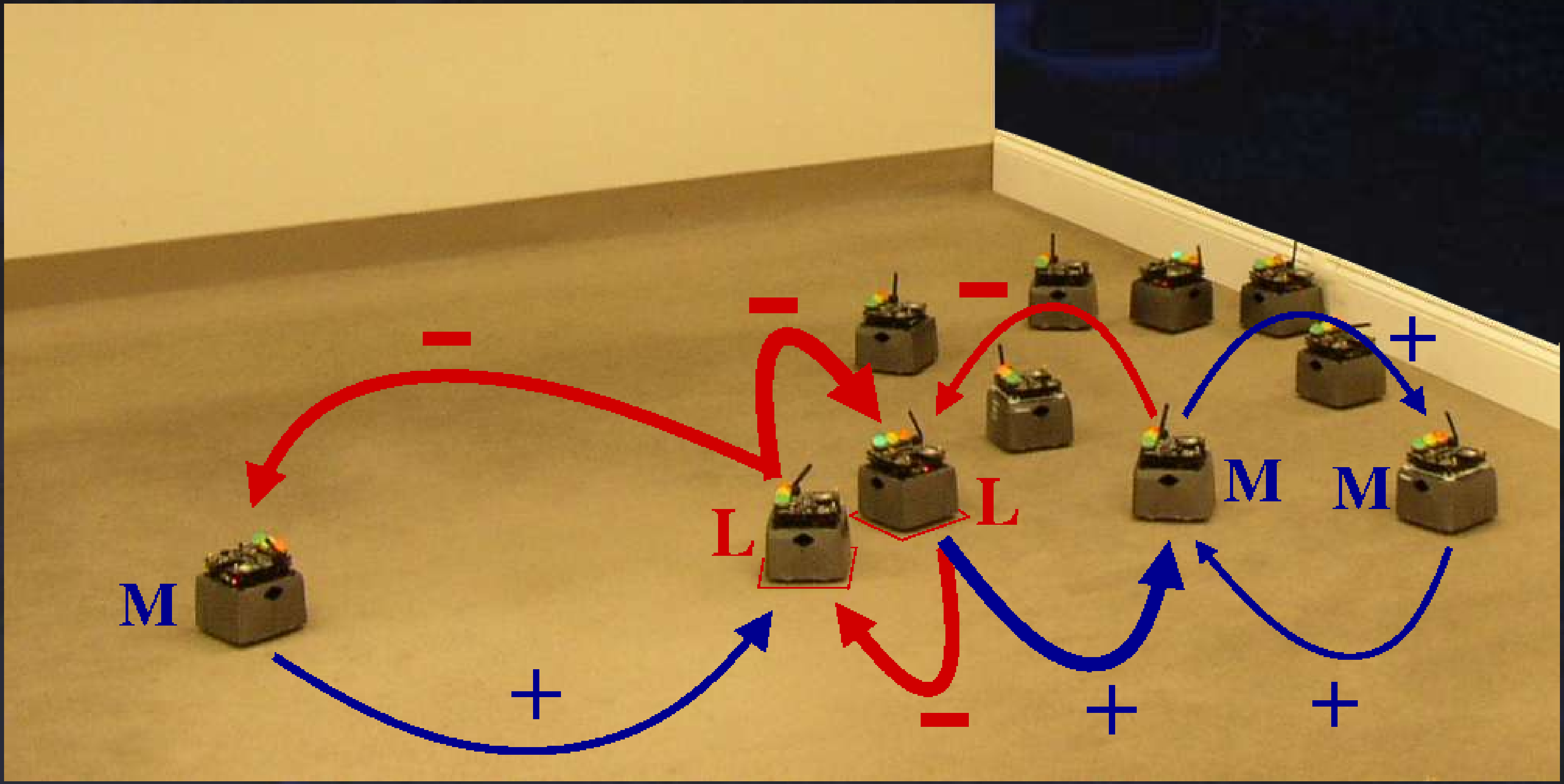
Dynamic Task Allocation

- Decentralized assignment of tasks to robot: e.g., Mover or Landmark
- “Neural network” approach
  - Leaky integrators represent desire to move
  - Robots compete to move (interactions)
  - Guarantees dynamic task switching
- Some hard-coded constraints

Weight Functions



Robot Interactions



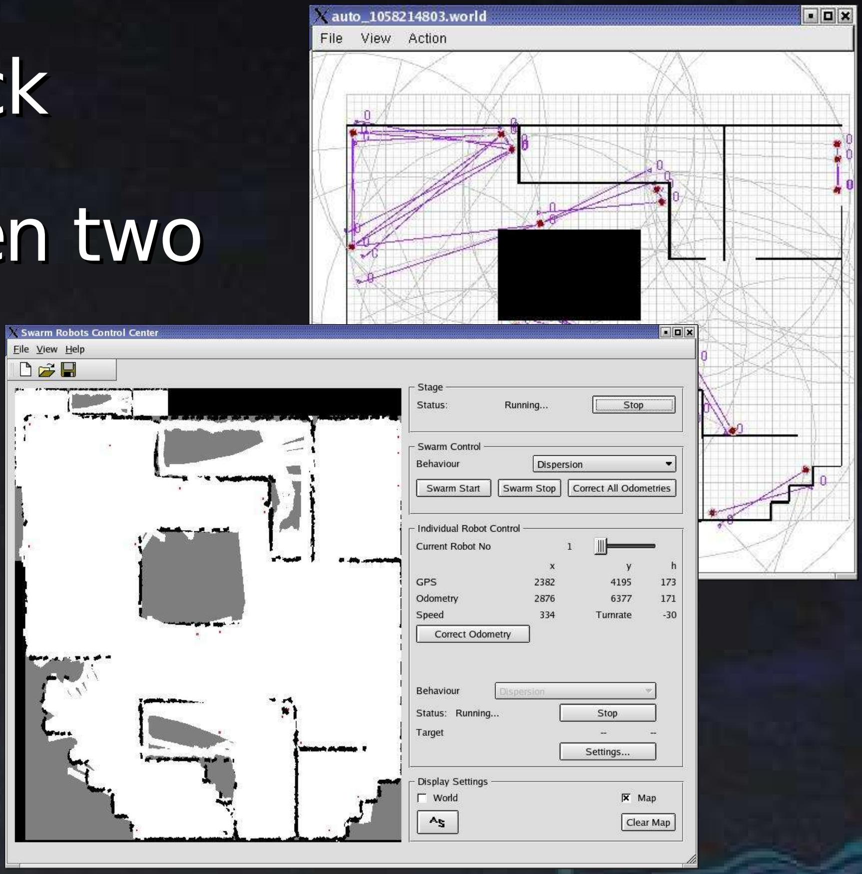
Task Allocation Features

- Decentralized
- No preset number of landmarks
- Robust (robots may die/be added)
- Scalable

Swarm Mapping: Simulation

Leverage limited sensors to create map:

- Robots disperse, bump along walls
- Bumps are marked black
- Space covered / between two robots is white
- Robots create local map in global frame of reference



Swarmbot Mapping: Challenges

Low memory

Noisy sensors

Low-bandwidth communications

Corrupted data

No communication to operator

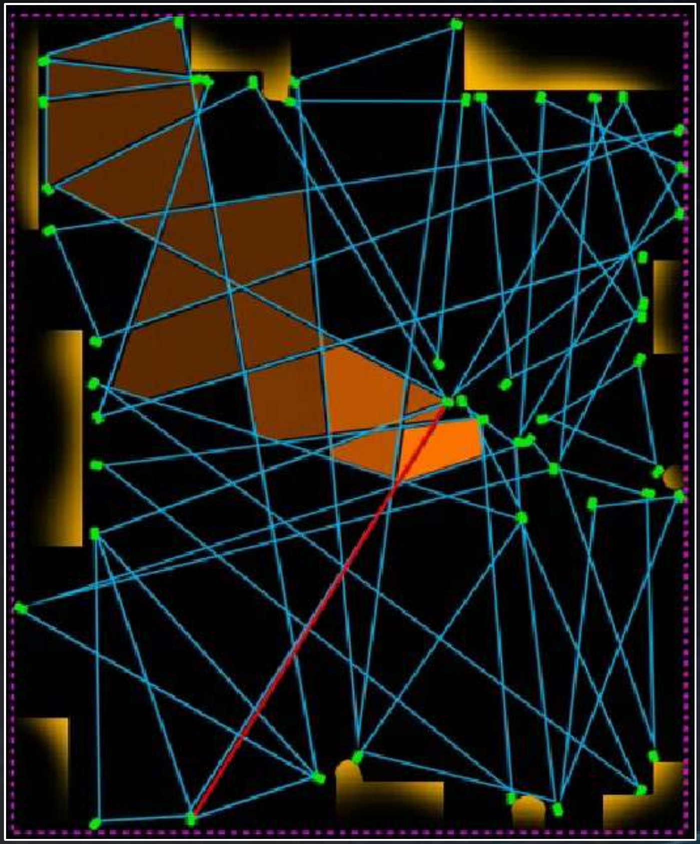
Swarmbot Mapping: Challenges

Analogy: collaborative mapping with blindfold, flexible cane and Morse code

Surprise: IT WORKS!

Intrusion detection

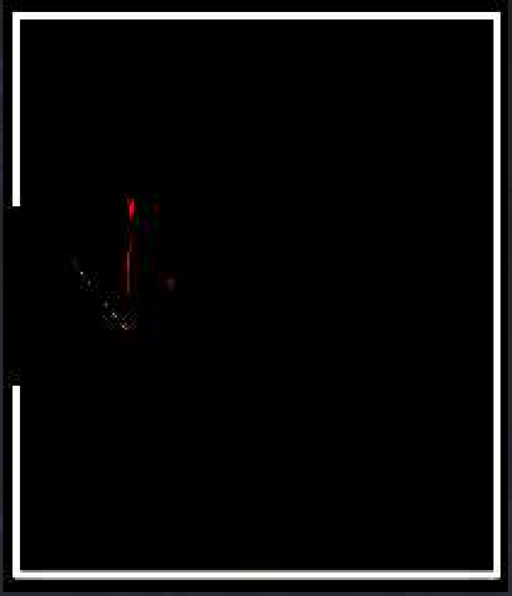
Objective: identify the location of an intruder moving through a grid of robots.



Icosystem's approach

Use strictly local information (nearest neighbors) to detect broken links

Constrain intruder location to triangles



Conclusions

- Developed algorithms for collaborative localization, dynamic task allocation and swarm mapping
- Successful transition from software to hardware
- The robot swarm is able to carry out tasks that are impossible for a single robot
- Large swarm mapping example
- Leveraged existing software/hardware
- Overcame technical limitations