Wrap-up of Simulator Overview + Swarming / Flocking / Schooling

January 21, 2003

Class Meeting 3
Announcements

• Paper presentations:
  – Reading list distributed – available on course web page
  – Send me your preferences by Friday morning (Jan. 24).
  – Assignments will be made by class time on Tuesday, Jan. 28.
  – Refer to syllabus for expectations for paper presentations
Announcements (con’t.)

• TA Update:
  – 1/2-time: Jeff Barnett (barnett@cs.utk.edu)
  – 1/3-time: William Duncan (duncan@cs.utk.edu)
Outline

• Questions on Assignment #1?

• Complete quick review of Nomad 200 simulator

• Swarming / flocking / schooling
Wrap-up: Quick Review of Nomad200 Simulator

• (refer also to slides from end of last class)
Behavior-Based / Reactive Approach to Robot Control: Based on Biological Paradigm

- Several layers of sensing / control / action operating in parallel
- Combined output yields “emergent” behavior of system
Common Behavior-Based Approach for Robotics: Motor Schemas

- Motors schemas achieve behavioral fusion via vector summation
- Output of each behavior is a vector giving desired direction of robot motion
- All vectors are summed to give resultant robot direction of motion

\[ R = \sum (G_i \cdot R_i) \]

Behavioral fusion

More on this next time…
Typical mobile robot implementation architecture

- Essentially: PC on wheels/legs/tracks/...
Implication for Robot Control Code

• Two options:
  – Separate threads with appropriate interfaces, interrupts, etc.
  – Single process with “operating-system”-like time-slicing of procedures

• Usually: combination of both

• For now: let’s examine single process with “operating-system”-like time-slicing of procedures
Simple program:
Follow walls and obey human operator commands

• Assume we have the following functions needed:
  – Communications to operator interface -- commands such as “stop”, “go”, etc.
  – Sonar: used to follow wall and avoid obstacles

Want all of this to happen “in parallel” on single processor
Typical “single process” control approach to achieve functional parallelism

```c
int wall_follower() { /* one time slice of work */
int obstacle_avoider() { /* one time slice of work */
int communicator() { /* one time slice of work */
int controller_arbiter() { /* decides what action to take */

main()
{
    while (forever)
    {
        wall_follower();
        obstacle_avoider();
        communicator();
        controller_arbiter();
    }
}
```

*Note of caution: dependent upon programmer to ensure individual functions return after “time slice” completed*
Control Commands to Nomad200 Simulator

vm(translation, wheel_rotation, turret_rotation)

Robot continues to execute the given velocity commands until another
command issued.

Thus, duration of “time slice” is important.
In Robot Design Choices, Must Consider Real-World Challenges

Software Challenges:

• **Autonomous**: robot makes majority of decisions on its own; no human-in-the-loop control (as opposed to *teleoperated*)

• **Mobile**: robot does not have fixed based (e.g., wheeled, as opposed to *manipulator arm*)

• **Unstructured**: environment has not been specially designed to make robot’s job easier

• **Dynamic**: environment may change unexpectedly

• **Partially observable**: robot cannot sense entire state of the world (i.e., “hidden” states)

• **Uncertain**: sensor readings are noisy; effector output is noisy

*Let’s look at these in more detail…*
Examples and Effect of Unstructured Environment

• Examples of unstructured environment:
  – Nearly all natural (non-man-made) environments:
    • Deserts
    • Forests
    • Fields
  – To some extent, man-made environments not specifically designed for robots

• Impact:
  – Difficult to make assumptions about sensing expectations
  – Difficult to make assumptions about environmental characteristics
Example of Taking Advantage of **Semi-Structured Environment**

- If in most man-made buildings, assume perpendicular walls; allows straightening of “warped” walls caused by accumulated error.
Sources and Effect of Dynamic Environment

• Sources of dynamic environment:
  – Other robots/agents in the area
    • Teammates
    • Adversaries
    • Neutral actors
  – Natural events (e.g., rain, smoke, haze, moving sun, power outages, etc.)

• Impact:
  – Assumptions at beginning of mission may become invalid
  – Sensing/Action loop must be tight enough so that environment changes don’t invalidate decisions
Example of Effect of Dynamic Environment

**Possible control code:**

```python
while (forever) do:

    { free = check_for_obstacle_to_goal();
      if (free)
        move_straight_to_goal();
        sleep(a_while);
    }
```

briefly
Causes and Effect of **Partially Observable Environment**

- **Causes of partially observable environment:**
  - Limited resolution sensors
  - Reflection, occlusion, multi-pathing, absorption

- **Impact:**
  - Same actions in “same” state may result in different outcomes

**Example:**
Glass walls--laser sensors tricked
Sources and Effect of Uncertainty/Noise

• Sources of sensor noise:
  – Limited resolution sensors
  – Sensor reflection, multi-pathing, absorption
  – Poor quality sensor conditions (e.g., low lighting for cameras)

• Sources of effector noise:
  – Friction: constant or varying (e.g., carpet vs. vinyl vs. tile; clean vs. dirty floor)
  – Slippage (e.g., when turning or on dusty surface)
  – Varying battery level (drainage during mission)

• Impact:
  – Sensors difficult to interpret
  – Same action has different effects when repeated
  – Incomplete information for decision making
Example of Effect of Noise on Robot Control Code: “Exact” Motions vs. Servo Loops

Two possible control strategies:

(1) “Exact” motions:
   - Turn right by amount \( \theta \)
   - Go forward by amount \( d \)

(2) Servo loop:
   - If to the left of desired trajectory, turn right.
   - If to the right of desired trajectory, turn left.
   - If online with desired trajectory, go straight.
   - If error to desired trajectory is large, go slow.
   - If error to desired trajectory is small, go fast.
Consider effect of noise: “Exact” control method

• “Exact” method:

\[ \theta_d, d_1, \theta_1, d_2, \theta_2 \]

Current robot position & orientation

\[ \theta, d, \theta_1, d_1, \theta_2, d_2 \]

\[ \theta_1, d_1: \text{actual angle, distance traveled;} \]

Noise ➔ overshoot goal; have to turn back to goal

\[ \text{Doesn’t give good performance} \]
Consider effect of noise: Servo method

- Servo method:

  - Current robot position & orientation
  - Goal

  Much better performance in presence of noise
Summary of Nomad 200 Simulator Introduction

• Typical robot control involves several activities/behaviors in parallel

• Roughly emulate parallelism through manually time-slicing your code

• Lots of sources of noise make exact control difficult

• For better success, control robot motion via servo-loop control, not exact distance commands
Outline

• Complete quick review of Nomad 200 simulator

→ Swarming / flocking / schooling
Swarming / Flocking / Schooling

• Natural flocks consist of two balanced, opposing behaviors:
  – Desire to stay close to flock
  – Desire to avoid collisions with flock

• Why desire to stay close to flock?
  – In natural systems:
    • Protection from predators
    • Statistically improving survival of gene pool from predator attacks
    • Profit from a larger effective search pattern for food
    • Advantages for social and mating activities
Contrasts in Swarming / Flocking / Schooling

- Many apparent contrasts in swarms/flocks/schools:
  - Made up of discrete agents, yet overall motion seems fluid
  - Simple in concept, yet visually complex
  - Randomly arrayed, yet highly synchronized
  - Seems intentional, with centralized control, yet evidence suggests group motion is only due to aggregate result of individual agents
Computational Complexity of Flocking?

• In natural systems,
  – No indication that flocking is bounded
  – Flocks don’t become “overloaded” or “full” as new agents join
    • E.g.: Herring migration: schools are as long as 17 miles and contain millions of fish
  – Individual natural agent (e.g., bird) doesn’t seem to pay attention to each flockmate
  – In birds, seem to be 3 categories of awareness:
    • Itself
    • 2-3 closest neighbors
    • Rest of flock

⇒ Natural systems use a constant time algorithm for flocking
Approach Closely Related to Particle Systems

• **Particle systems:**
  – Collections of large numbers of individual particles, each having its own behavior
  – Used to model fire, smoke, clouds, spray/foam of ocean waves, etc.
  – Particles are created, age, and die off

• However, need to also add:
  – Geometric objects with local coordinate system, since robots aren’t points
  – Geometric/kinematic models of motion of a body
Generating Flocking and Foraging in Distributed Robotics

• Work of Mataric, 1994

• Fundamental principle:
  – Define *basis behaviors* as general building blocks for synthesizing group behavior

• Behavior:
  – Control law that takes advantage of dynamics of system to effectively achieve and maintain goals

• Basis behaviors:
  – Minimal set of behaviors, with appropriate compositional properties

The Nerd Herd: Mataric, MIT, 1994
Selection of Basis Behaviors

• Difficult to define “optimality” metric or proofs of correctness

• Criteria used for selecting basis behaviors:
  
  – **Necessary**: each achieves, or helps achieve, a relevant goal that cannot be achieved with other behaviors in the set, and cannot be reduced to them

  – **Sufficient**: for accomplishing goals in domain, so that no other behaviors are needed

  – **Additional characteristics**: Simple, local, stable, robust, scalable
Basis Behaviors Selected

• Set of behaviors proposed:
  – Save-wandering: ability of a group to move about while avoiding collisions
  – Following: ability of an agent to move behind another agent, retracing its path and maintaining a line or queue
  – Aggregation: ability of a group to gather so as to establish and maintain some maximum inter-agent distance
  – Dispersion: ability of a group to spread out so as to establish and maintain some minimum inter-agent distance
  – Homing: ability to find a particular region or location
Objective: Combine Basis Behaviors

• Idea:
  – Use basis behaviors in a variety of combinations to enable creation of more complex group behaviors, such as:
    • Flocking
    • Foraging

  – Other possibilities (not shown in Mataric):
    • Surrounding
    • Herding
Robots Used in Study

- “Nerd Herd” collection of 20 robots:
  - 12” long
  - 4 wheels
  - Piezo-electric bump sensors around body
  - Two-pronged gripper for puck-carrying
    - Contact switches on tip of each gripper finger
    - 6 infrared sensors (2 forward, 2 inside, 2 down)
  - Radio system for:
    - Localization
    - Communication
    - Data collection
    - “Kin” recognition
  - Programmed using Behavior Language (from subsumption)
Safe-Wandering Algorithm

• **Avoid-Kin:**
  - Whenever an agent is within $d_{\text{avoid}}$
    - If the nearest agent is on the left
      - Turn right
    - Otherwise, turn left

• **Avoid-Everything-Else**
  - Whenever an obstacle is within $d_{\text{avoid}}$
    - If obstacle is on right only, turn left
    - If obstacle is on left only, turn right
    - After 3 consecutive identical turns, backup and turn
    - If an obstacle is on both sides, stop and wait.
    - If an obstacle persists on both sides, turn randomly and back up

• **Move-Around:**
  - Otherwise move forward by $d_{\text{forward}}$, turn randomly
Following Algorithm

Follow:
- Whenever an agent is within $d_{\text{follow}}$
  - If an agent is on the right only, turn right
  - If an agent is on the left only, turn left

If sufficient robot density, $\text{safe}_{\text{wandering}} + \text{follow}$ yield more complex behaviors:
- e.g., osmotropotaxic behavior of ants: unidirectional lanes
Dispersion Algorithm

Dispersion:
- Whenever one or more agents are within $d_{\text{disperse}}$
  • Move away from Centroid_{disperse}
Aggregation Algorithm

**Aggregate:**
- Whenever nearest agent is outside $d_{aggregate}$
  - Turn toward the local centroid$_{aggregate}$, go.
- Otherwise, stop.
Homing Algorithm

Home:

- Whenever at home
  - Stop
- Otherwise, turn toward home, go.
Perceptual Information, Behavior Selection

- Perceptual information is encoded as a series of predicates:
  - At-home?
  - Have-puck?
  - Crowded?
  - Behind-kin?
  - Sense-puck?
Combining Basis Behaviors to Form Higher-Level Behaviors

• Two types of coordination:

  – Complementary:
    • Outputs are executed concurrently
    • Direct combination (a vector summation process)

  – Contradictory:
    • Outputs can only be executed one at a time
    • Temporal combination (a sequence through behavior states)
Generating Flocking Through Behavior Combinations

• **Flock:**
  - Sum weighted outputs from Safe-Wander, Disperse, Aggregate, and Home

In general, flocking should allow agents to move around obstacles:

Note how flock moves around obstacles, yet stays together as a flock

Work of Reynolds (1987)
What Mobile Robot Capabilities are Needed to Implement Flocking?

• Capabilities:
  – Ability to sense nearby robots:
    • Position, direction and speed of travel
  – Ability to sense nearby obstacles
  – Ability to distinguish between robots and obstacles

• How to obtain?
  – Vision recognition
  – Beacons on robots detected locally (e.g., via infrared, etc.)
  – Global positioning + radio broadcast of global robot positions
Foraging Example

• Foraging:
  – Robot moves away from home base looking for attractor objects
  – When detect attractor object, move toward it, pick it up, and return it to home base
  – Repeat until all attractors in environment have been returned home

• High-level behaviors required to accomplish foraging:
  – *Wander*: move through world in search of an attractor
  – *Acquire*: move toward attractor when detected
  – *Retrieve*: return the attractor to the home base once acquired
Finite State Acceptor (FSA) Diagram for Foraging

- **Start** to **Wander**
- **Wander** to **Acquire**
- **Acquire** to **Halt**
- **Halt** to **Done**
- **Start** to **Release**
- **Release** to **Wander**
- **Acquire** to **Grab**
- **Grab** to **Release**
- **Release** to **Wander**
Mataric’s Approach to Generating Foraging Through Basis Behavior Combinations

• **Foraging:**
  - Whenever crowded? Disperse.
  - Whenever at-home?
    - If have-puck? drop-puck
    - Otherwise disperse
  - Whenever sense-puck?
    - If not have-puck? pickup-puck
  - Whenever behind-kin? Follow.
Movie of Nerd Herd Results (1994)
What Mobile Robot Capabilities are Needed to Implement Foraging?

• Capabilities:
  – Ability to sense nearby robots’ positions
  – Ability to sense nearby obstacles
  – Ability to distinguish between robots and obstacles
  – Ability to sense and grasp pucks
  – Ability to find home
  – Ability to search for pucks

• How to obtain?
  – Vision recognition
  – Beacons on robots detected locally (e.g., via infrared, etc.)
  – Global positioning + radio broadcast of global robot positions
Mataric Approach Similar to Reynold’s Simulated Flocks / Boids (1987)


Separation: steer to avoid crowding local flockmates

Alignment: steer towards average heading of local flockmates

Cohesion: steer to move toward the average position of local flockmates

(Similar to Mataric’s dispersion)

(Similar to Mataric’s aggregation)
Reynold’s Boid Flocks

• Boid neighborhood characterized by:
  – Distance (measured from center of boid)
  – Angle (measured from direction of flight)
Movie of Reynold’s Boids Flocking

Simulated boid flock avoiding cylindrical obstacles
Boids Approach Was Made into Movie

For lots more information on Flocks, Herds, Schools ...

• Great web site:
  http://www.red3d.com/cwr/boids/

  Contains lots of pointers to related literature on this topic
Preview of Next Class

• Key techniques for single autonomous robot control
   – Focus on:
     • Motor schemas
     • Potential fields
     • Behavior combinations

   – Presented by William Duncan

   – (Lynne at Intel the rest of this week)