Lecture 15: Coordination I

Autonomous Robots, 16-200
Carnegie Mellon University Qatar
Overview

- Logistics
- Path Planning with A*
- Movie
- Coordination
Logistics

- HW #3 is due tonight at midnight
- Graded research assignments will be available outside Bernardine’s office by 6p.m. today
- Lab #4 is posted and is due November 17th
- We will be out of Qatar over Eid, but you can contact us via email
- Submit your project topics TODAY if you haven’t done so already. We will send you feedback shortly. Start your projects today!!!
Plan

- **Today:**
  - A* and Coordination I

- **November 9th:**
  - Overall comments on research assignment and Coordination II

- **November 10th Lab:**
  - Work on projects and Lab 4
A*  

- Often used to search for the lowest cost path from the \textit{q\_start} to the \textit{q\_goal} location in a graph of cells/voronoi/visibility/quadtree

- Problem setting:
  - A graph of nodes, \( n_s \) is start, \( n_g \) is goal
  - A cost metric for moving from node to node
  - Want to find the path from \( n_s \) to \( n_g \) with the minimum cost
Remember Best-first Search?

- Define a heuristic $h(n)$
  - Estimates cost of shortest path from node $n$ to the goal
  - E.g. Euclidean distance to goal
- Greedy search always chooses to follow child with lowest $h(.)$ value
  - Can produce non-optimal paths!
- Also called depth-first
Remember Best-first Search?

- **Define a** heuristic $h(n)$
  - Estimates cost of shortest path from node $n$ to the goal
  - E.g. Euclidean distance to goal
- **Greedy search always chooses to** follow child with lowest $h(.)$ value
  - Can produce non-optimal paths!
- **Also called depth-first**

Path length=$13$!!!
A*  

- Define $g(n)$
  - Cost to reach node $n$ from start
- Greedy search by following child with lowest $f(n)$
  - $f(n) = g(n) + h(n)$
- Provided $h(n)$ is admissible, A* will be complete and optimal
  - Performance will depend on how good $h(n)$ is
- Admissibility:
  - $h(n)$ must never overestimate the cost
A* Example

\[
\begin{array}{cccc}
9 & 8 & 7 & 6 \\
8 & S & 4 & 5 \\
7 & 6 & 3 & 4 \\
6 & 5 & 2 & 3 \\
5 & 6 & 1 & 2 \\
4 & 3 & 2 & G \\
5 & 4 & 3 & 1 \\
6 & 5 & 4 & 3 \\
\end{array}
\]
**A* Example**

\[
f(n) = h(n) + g(n)
\]

<table>
<thead>
<tr>
<th></th>
<th>8+1</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>8+1</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>8+1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Queue of children to consider
A* Example

```
g(n)  h(n)
```

```
<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>8+1</th>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8+1</td>
<td></td>
<td>5+2</td>
<td></td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>7+1</td>
<td>1(7)</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>5+2</td>
<td></td>
<td>4</td>
<td>G</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
### A* Example

<table>
<thead>
<tr>
<th>9</th>
<th>8+1</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8+1</td>
<td>S</td>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7+2</td>
<td>1(7)</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6+3</td>
<td>2(7)</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

$h(n)$

The image depicts a grid with numbers and letters, with arrows indicating the path and the function $h(n)$.
**A* Example**

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>8+1</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8+1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>7+2</td>
<td>1(7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>3(9)</td>
<td>2(7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>5+4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
### A* Example

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>8+1</th>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8+1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7+2</td>
<td>1(7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3(9)</td>
<td>2(7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4(9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5+4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
### A* Example

<table>
<thead>
<tr>
<th></th>
<th>8+1</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8+1</td>
<td>S</td>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7+2</td>
<td>1(7)</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3(9)</td>
<td>2(7)</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4(9)</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5(9)</td>
<td>6(9)</td>
<td>2+7</td>
<td></td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>5+6</td>
<td>4+7</td>
<td>3</td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
## A* Example

An interesting thing just happened!!!
# A* Example

<table>
<thead>
<tr>
<th>9+1</th>
<th>1(9)</th>
<th>7+2</th>
<th>6</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8+1</td>
<td><strong>S</strong></td>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7+2</td>
<td>1(7)</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3(9)</td>
<td>2(7)</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4(9)</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5(9)</td>
<td>6(9)</td>
<td>7(9)</td>
<td>G</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5+6</td>
<td>4+7</td>
<td>3+8</td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
## A* Example

<table>
<thead>
<tr>
<th></th>
<th>9+1</th>
<th>1(9)</th>
<th>2(9)</th>
<th>6+3</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>8+1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7+2</td>
<td>1(7)</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3(9)</td>
<td>2(7)</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4(9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5(9)</td>
<td>6(9)</td>
<td>7(9)</td>
<td></td>
<td></td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>5+6</td>
<td>4+7</td>
<td>3+8</td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td></td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
**A* Example**

Path Length = 10!!
Coordination

The problem:

- How do we make 2 or more robots work together to execute a task?
- Want to execute the task better than if robots do not coordinate

Hey, can you help me move this box? I can’t do it myself

NASA’s RoboNaut
Movie

USC – Interaction Lab – Maja Mataric and Brian Gerkey

Highlights from experiments performed with the pusher-watcher algorithm, implemented using the MURDOCH task allocation system
Motivation

Multirobot applications

Remote Operations

Warehouse Management

Production Plants

Intelligent Environments

Automated Construction

Education and Entertainment
Motivation

Multirobot applications

Urban Reconnaissance

Urban Search And Rescue

Agriculture

Hazardous Exploration

Hazardous Clean-Up
Motivation

Why Multiple Robots?

- Task requirement
- Improved efficiency
- Simplified robot design *(cooperating specialists)*
- Improved localization
- Improved robustness *(redundancy)*
- Wider variety of possible solutions
Motivation

Coordinating multiple robots to cooperate in dynamic environments is a difficult problem to solve!
Dynamic Environments

- Many unknowns
- Changing obstacles
- Changing task requirements
- Changing resources
- Limited resources
Required Characteristics

1. Robustness to Failures
2. Response Speed
3. Solution Efficiency
4. Handling imperfect Information
5. Handling Imperfect Communication
6. Efficient Resource Utilization
7. Efficient Task Allocation
8. Efficient Role Adoption
9. Handling New Input
10. Fluid Addition/Loss of Robots
11. Handling Heterogeneous Teams
12. Extensibility
13. Flexibility/Easily Applied to New Applications
14. Handling Tight Coordination
15. Scalability
16. Learning and On-line Adaptation
17. Proven Implementation on Robots
Coordination

- Robot A’s choice of action depends on:
  - Robot B’s actions
  - Robot B’s sensing
  - Robot B’s communication

- Examples:
  - Multi-robot exploration/mapping
  - Multi-robot surveillance/sensor networks
  - Construction
  - Passing in ball sports
  - Furniture moving, formation control
Coordination Approaches

- **Fully Centralized**

- **Fully Distributed**

- **Hybrids**
Common Components of Coordination

- Robustness to failure
- Task allocation
- Role assignment
- Action synchronization (Temporal coordination)
In the real world things always break!
Categories of Failure

1. Communication Failure
2. Partial Robot Malfunctions
3. Robot Death
Task Allocation

- Who is going to take responsibility for each task?
- Can one robot execute each task?
- Can the robots exchange tasks?
- Who is finally accountable to the operator?
- Some tasks can be decomposed into single-robot tasks:
  - Mapping
  - Exploration
Role Assignment

- What happens when tasks are tightly coordinated?
- Then we need role assignment
  - Which robot will play each role?
- Static role assignment:
  - Locker Room Agreements [Stone, 98]
- Dynamic role assignment:
  - Work out best assignment given state of world
  - Can be a hard problem!
- Example domain: soccer
Action Synchronization

- What if the roles require robots to synchronize their actions?
  - Tightly coupled
    - Many actions synchronized, short time ranges
    - E.g. Marching bands, passing
  - Loosely coupled
    - Fewer actions, long time ranges
    - E.g. complete painting after house is built
- Can be modeled as coordinated state machines
  - We call these *Plays* [Bowling, et al. 04]
Plays: Coordinated State Machines

Role 1
- Get ball
- Pass
- Move
  - Ale’oop
  - Rebound

Role 2
- Find Free Space
- Receive
- Dribble
- Shoot

Synchronize
Putting It All Together

- Some tasks require coordination at all three levels!
- Example: Treasure Hunt!
  - Assign tasks: search for treasure in area “x”
  - Assign roles: Robot A should search while robot B makes sure they don’t get lost
  - Synchronize actions: Robots A and B need to work together!
- Next time we will talk about how to do this in more detail
- Also, what if your team consists of both humans and robots?
End of slides!

Happy Holidays!