Applied Autonomous Robots I
Fall 2012

- BUG Algorithms
- Potential Fields
What’s Special About Bugs?

- Many planning algorithms assume global knowledge
- Bug algorithms assume only *local* knowledge of the environment and a global goal
- Bug behaviors are simple:
  - Follow a wall (right or left)
  - Move in a straight line toward goal

- Three kinds of bugs:
  - Bug 1 and Bug 2 assume essentially tactile sensing
  - Tangent Bug deals with finite distance sensing
A Few General Concepts

- Workspace $W$
  - $\mathbb{R}(2)$ or $\mathbb{R}(3)$ depending on the robot
  - could be infinite (open) or bounded (closed/compact)
- Obstacle $O_i$
- Free workspace $W_{free} = W \setminus \bigcup_i O_i$
The Bug Algorithms

Insect-inspired

- known direction to goal
  - robot can measure distance $d(x,y)$ between pts $x$ and $y$
- otherwise local sensing
  - walls/obstacles & encoders
- reasonable world
  - finitely many obstacles in any finite area
  - a line will intersect an obstacle finitely many times
  - Workspace is bounded

Courtesy of Howie Choset, G. D. Hager, and Z. Dodds
Buginner Strategy

“Bug 0” algorithm

- known direction to goal
- otherwise local sensing
  - walls/obstacles & encoders

Some notation:
- $q_{\text{start}}$ and $q_{\text{goal}}$
- “hit point” $q_{H_i}$
- “leave point” $q_{L_i}$

A path is a sequence of hit/leave pairs bounded by $q_{\text{start}}$ and $q_{\text{goal}}$
Buginner Strategy

“Bug O” algorithm

- known direction to goal
- otherwise local sensing
  - walls/obstacles & encoders

1) head toward goal
2) follow obstacles until you can head toward the goal again
3) continue
Buginner Strategy

1) head toward goal
2) follow obstacles until you can head toward the goal again
3) continue
Bug Zapper

What map will foil Bug 0?

"Bug 0" algorithm
1) head toward goal
2) follow obstacles until you can head toward the goal again
3) continue
A Better Bug?

But add some memory!

- known direction to goal
- otherwise local sensing
  - walls/obstacles
  - encoders
But some computing power!

- known direction to goal
- otherwise local sensing
  walls/obstacles & encoders

"Bug 1" algorithm

1) head toward goal
2) if an obstacle is encountered, circumnavigate it and remember how close you get to the goal
3) return to that closest point (by wall-following) and continue

Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987
Bug 1 More Formally

- Let $q^L_0 = q_{\text{start}}$, $i = 1$
- repeat
  - repeat
    - from $q^L_{i-1}$ move toward $q_{\text{goal}}$
    - until goal is reached or obstacle encountered at $q^H_i$
    - if goal is reached, exit
    - repeat
      - follow boundary recording pt $q^L_i$ with shortest distance to goal
      - until $q_{\text{goal}}$ is reached or $q^H_i$ is re-encountered
      - if goal is reached, exit
      - Go to $q^L_i$
      - if move toward $q_{\text{goal}}$ moves into obstacle
        - exit with failure
      - else
        - $i = i + 1$
        - continue
Another Step Forward?

Call the line from the starting point to the goal the \( m\text{-line} \)

“Bug 2” Algorithm

Courtesy of Howie Choset, G. D. Hager, and Z. Dodds
A Better Bug?

Call the line from the starting point to the goal the *m-line*

---

**“Bug 2″ Algorithm**

1) head toward goal on the *m-line*
2) if an obstacle is in the way, follow it until you encounter the *m-line* again.
3) Leave the obstacle and continue toward the goal

---

Courtesy of Howie Choset, G. D. Hager, and Z. Dodds
A Better Bug?

"Bug 2" Algorithm

1) head toward goal on the m-line
2) if an obstacle is in the way, follow it until you encounter the m-line again.
3) Leave the obstacle and continue toward the goal
A Better Bug?

“Bug 2” Algorithm

1) head toward goal on the m-line
2) if an obstacle is in the way, follow it until you encounter the m-line again closer to the goal.
3) Leave the obstacle and continue toward the goal

Better or worse than Bug1?
Bug 2 More Formally

- Let $q^L_0 = q_{\text{start}}, i = 1$
- repeat
  - repeat
    - from $q^L_{i-1}$ move toward $q_{\text{goal}}$ along the m-line
    - until goal is reached or obstacle encountered at $q^H_i$
    - if goal is reached, exit
  - repeat
    - follow boundary
    - until $q_{\text{goal}}$ is reached or $q^H_i$ is re-encountered or m-line is re-encountered, $x$ is not $q^H_i$, $d(x, q_{\text{goal}}) < d(q^H_i, q_{\text{goal}})$ and way to goal is unimpeded
    - if goal is reached, exit
    - if $q^H_i$ is reached, return failure
  - else
    - $q^L_i = m$
    - $i = i + 1$
    - continue
Head-to-Head Comparison

Draw worlds in which Bug 2 does better than Bug 1 (and vice versa).

Bug 2 beats Bug 1

Bug 1 beats Bug 2

Courtesy of Howie Choset, G. D. Hager, and Z. Dodds
Bug 1 vs. Bug 2

- BUG 1 is an \textit{exhaustive search algorithm}
  - it looks at all choices before committing

- BUG 2 is a \textit{greedy algorithm}
  - it takes the first thing that looks better

- In many cases, BUG 2 will outperform BUG 1, but

- BUG 1 has a more predictable performance overall
A More Realistic Bug

- As presented: global beacons plus contact-based wall following

- The reality: we typically use some sort of range sensing device that lets us look ahead (but has finite resolution and is noisy).

- Let us assume we have a range sensor
Raw Distance Function

\[ \rho(x, \theta) = \min_{\lambda \in [0, \infty]} d(x, x + \lambda [\cos \theta, \sin \theta]^T), \]

such that \( x + \lambda [\cos \theta, \sin \theta]^T \in \bigcup_i \mathcal{W}_i \).

\[ \rho : \mathbb{R}^2 \times S^1 \rightarrow \mathbb{R} \]

Saturated raw distance function

\[ \rho_R(x, \theta) = \begin{cases} 
\rho(x, \theta), & \text{if } \rho(x, \theta) < R \\
\infty, & \text{otherwise.} 
\end{cases} \]
Intervals of Continuity

- Tangent Bug relies on finding endpoints of finite, continuous segments of $\rho_R$
Motion to Goal Transitions
from Moving Toward Goal to “Following Obstacles”

Currently, the motion-to-goal behavior “thinks” the robot can get to the goal
Motion to Goal Transitions

Among Moving Toward Goal to “Following Obstacles”

Now, it starts to see something --- what to do?
Ans: Choose the pt $O_i$ that minimizes $d(x,O_i) + d(O_i,q_{goal})$

--- t to the goal
Minimum Heuristic Example

At x, robot knows only what it sees and where the goal is,

so moves toward $O_2$. Note the line connecting $O_2$ and goal pass through obstacle

so moves toward $O_4$. Note some “thinking” was involved and the line connecting $O_4$ and goal pass through obstacle

Choose the pt $O_i$ that minimizes $d(x, O_i) + d(O_i, q_{goal})$
Motion to Goal Example

Choose the pt $O_i$ that minimizes $d(x, O_i) + d(O_i, q_{goal})$
Transition From Motion-to-Goal

Choose the pt $O_1$ that minimizes $d(x, O_1) + d(O_1, q_{goal})$

Problem: what if this distance starts to go up?

Ans: start to act like a BUG and follow boundary

$M$ is the point on the “sensed” obstacle which has the shortest distance to the goal

Followed obstacle: the obstacle that we are currently sensing

Blocking obstacle: the obstacle that intersects the segment

$$(1 - \lambda)x + \lambda q_{goal} \quad \forall \lambda \in [0, 1]$$

They start as the same
Boundary Following

Move toward the $O_i$ on the followed obstacle in the “chosen” direction

Maintain $d_{\text{followed}}$ and $d_{\text{reach}}$

$M$ is the point on the “sensed” obstacle which has the shortest distance to the goal

Followed obstacle: the obstacle that we are currently sensing

Blocking obstacle: the obstacle that intersects the segment

They start as the same

Courtesy of Howie Choset, G. D. Hager, and Z. Dodds
Example: Zero-Range Sensor

1. Robot moves toward goal until it hits obstacle 1 at H1
2. Pretend there is an infinitely small sensor range and the Oi which minimizes the heuristic is to the right
3. Keep following obstacle until robot can go toward obstacle again
4. Same situation with second obstacle
5. At third obstacle, the robot turned left until it could not increase heuristic
6. D\text{followed} is distance between M3 and goal, d\text{reach} is distance between robot and goal because sensing distance is zero
Example: Finite Range Sensor
Example: Infinite Range Sensor
Tangent Bug

- Tangent Bug relies on finding endpoints of finite, conts segments of \( \rho_R \)

Now, it starts to see something --- what to do?
Ans: Choose the pt \( O_i \) that minimizes \( d(x, O_i) + d(O_i, q_{goal}) \)

“Heuristic distance”
Tangent Bug

- Tangent Bug relies on finding endpoints of finite, conts segments of $\rho_R$

Problem: what if this distance starts to go up? Ans: start to act like a BUG and follow boundary
The Basic Ideas

• A motion-to-goal behavior as long as way is clear or there is a visible obstacle boundary pt that decreases heuristic distance

• A boundary following behavior invoked when heuristic distance increases.

• A value $d_{\text{followed}}$ which is the shortest distance between the sensed boundary and the goal

• A value $d_{\text{reach}}$ which is the shortest distance between blocking obstacle and goal (or my distance to goal if no blocking obstacle visible)

• Terminate boundary following behavior when $d_{\text{reach}} < d_{\text{followed}}$
Summary

- Bug 1: Safe and reliable
- Bug 2: better in some cases, worse in others
- Target Bug: supports range sensing
Schedule

- Sample *complete* maps will be provided last week
- Final partial map will be made available 11/26
- Project Presentations – in class on 12/3
  - 10-15 min / team
- Competition Day == Final Exam Day **12/13**
- Reschedule Final Exam to 12/13 10:30-12:30 am
- Final Report Due Date 12/10 @ 11:59 am
- 1 Page Competition Post-Mortem == Final Exam Day 11:59 EST
Final Project Presentations

- 12/7 – Powerpoint
- 10-15 minutes
  - Your strategy
    - Perception: sensor suite
    - Navigation: localization, motion planning, and motion control
  - Demonstrations
  - Preliminary results
Another Method of Thinking

- Think of the goal as the bottom of a bowl
- The robot is at the rim of the bowl
- What will happen
The General Idea

- Both the bowl and the spring analogies are ways of storing potential energy

- The robot moves to a lower energy configuration

- A potential function is a function $U : \mathbb{R}^m \rightarrow \mathbb{R}$

- Energy is minimized by following the negative gradient of the potential energy function:

$$\nabla U(q) = DU(q)^T = \left[ \frac{\partial U}{\partial q_1}(q), \ldots, \frac{\partial U}{\partial q_m}(q) \right]^T$$

- We can now think of a vector field over the space of all $q$'s ...
  - at every point in time, the robot looks at the vector at the point and goes in that direction
Attractive/Repulsive Potential Field

\[ U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q) \]

- \( U_{\text{att}} \) is the “attractive” potential – move to the goal
- \( U_{\text{rep}} \) is the “repulsive” potential – avoid obstacles
Attractive Potential

**Conical Potential**

\[ U(q) = \zeta d(q, q_{\text{goal}}). \]

\[ \nabla U(q) = \frac{\zeta}{d(q, q_{\text{goal}})} (q - q_{\text{goal}}). \]

**Quadratic Potential**

\[ U_{\text{att}}(q) = \frac{1}{2} \zeta d^2(q, q_{\text{goal}}), \]

\[ F_{\text{att}}(q) = \nabla U_{\text{att}}(q) = \nabla \left( \frac{1}{2} \zeta d^2(q, q_{\text{goal}}) \right), \]

\[ = \frac{1}{2} \zeta \nabla d^2(q, q_{\text{goal}}), \]

\[ = \zeta (q - q_{\text{goal}}), \]
The Repulsive Potential

\[ U_{\text{rep}}(q) = \begin{cases} \frac{1}{2} \eta \left( \frac{1}{D(q)} - \frac{1}{Q^*} \right)^2, & D(q) \leq Q^*, \\ 0, & D(q) > Q^* \end{cases} \]

whose gradient is

\[ \nabla U_{\text{rep}}(q) = \begin{cases} \eta \left( \frac{1}{Q^*} - \frac{1}{D(q)} \right) \frac{1}{D^2(q)} \nabla D(q), & D(q) \leq Q^*, \\ 0, & D(q) > Q^* \end{cases} \]
The Total Potential

\[ U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q) \]

\[ F(q) = -\nabla U(q) \]
Potential Fields
Gradient Descent

- A simple way to get to the bottom of the potential

\[ \dot{c}(t) = -\nabla U(c(t)) \]

- A critical point is a point \( x \) s.t. \( q^* \) where \( \nabla U(q^*) = 0 \)
  - Equation is stationary at a critical point
  - Max, min, saddle
  - Stability?
The Hessian

- For a 1-D function, how do we know if we are at the unique max or unique min?

- The Hessian is an nxn matrix of second derivatives

- If the Hessian is non-singular ($\text{Det}(H) \neq 0$), the critical point is a unique pt.
  - If $H$ is positive definite $\rightarrow$ minimum
  - If $H$ is negative definite $\rightarrow$ maximum
  - If $H$ is indefinite $\rightarrow$ saddle point
Gradient Descent

Gradient Descent:
- \( q(0) = q_{\text{start}} \)
- \( i = 0 \)
- while \( \nabla U(q(i)) \neq 0 \) do
  - \( q(i+1) = q(i) - \alpha(i) \nabla U(q(i)) \)
  - \( i = i + 1 \)
In Practice: Computing Distances
Computing Distances: Use a Grid

- Use a discrete version of space and work from there

- One way to do this: Brushfire Algorithm
  - Need to define a grid on the workspace
  - Need to define connectivity (4 or 8)
  - Obstacles start with a 1 in grid; free space is zero
Brushfire Algorithm

- Initially: create a queue $L$ of cells on the boundary of all obstacles

- While $L$ Not Empty
  - Pop the top element of $L$
  - If $d(t) = 0$
    - Set $d(t) = 1 + \min_{t' \in N(t), d(t) \neq 0} d(t')$
    - Add all $t' \in N(t)$ with $d(t) = 0$ to $L$ (at the end)

- The result is a distance map $d$ where each cell holds the minimum distance to an obstacle

- Gradient of the distance is found by taking differences with neighboring cells.
Brushfire Algorithm: Example
Potential Functions Question

• How do we know that we have only a single (global) minimum

• We have two choices:
  • not guaranteed to be a global minimum: do something other than gradient descent (what?)
  • make sure only one global minimum (a navigation function, which we’ll see later).
The Wavefront Planner

- Apply the brushfire algorithm starting from the goal

- Label the goal pixel 2 and add all zero neighbors to L
  - While L is NOT empty
    - pop the top element of L, t
    - set $d(t)$ to $1 + \min_{t' \in N(t), d(t) > 1} d(t')$
    - Add all $t' \in N(t)$ with $d(t)=0$ to L (at the end)

- The result is now a distance for every cell
  - gradient descent is again a matter of moving to the neighbor with the lowest distance value
The Wavefront Planner: Setup
The Wavefront in Action (Part 1)

- Starting with the goal, set all adjacent cells with “0” to the current cell + 1
  - 4-Point Connectivity or 8-Point Connectivity?
  - Your Choice – use 8-pt in this example
The Wavefront in Action (Part 2)

- Now repeat with the modified cells
  - This will be repeated until no 0’s are adjacent to cells with values $\geq 2$
  - 0’s will remain when regions are unreachable
The Wavefront in Action (Part 3)

- Repeat again …
The Wavefront in Action (Part 4)

- And again …
The Wavefront in Action (Part 5)

- And again until

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

MEM380: Applied Autonomous Robots Fall 2012
The Wavefront in Action (Done)

- You’re done
  - Remember, 0’s should only remain if unreachable regions exist
The Wavefront, Now What?

- To find the shortest path, according to your metric, simply always move toward a cell with a lower number
- The numbers generated by the wavefront planner are roughly proportional to their distance to the goal
Wavefront Overview

- Divide the space into a grid
- Number the squares starting at the start in either 4 or 8 point connectivity starting at the goal, increasing until you reach the start.
- Your path is defined by any uninterrupted sequence of decreasing numbers that lead to the goal.
Potential Functions Question

- How do we know that we have only a single (global) minimum

- We have two choices:
  - not guaranteed to be a global minimum: do something other than gradient descent (what?)
  - make sure only one global minimum (a navigation function, which we’ll see later).
Potential Functions w/ Guarantees?

- Is there a systematic way of constructing potential functions to guarantee no local minima?

- Answer: YES -> Navigation Functions
  - A function $\phi: Q_{\text{free}} \rightarrow [0,1]$ is called a navigation function if it
    - is smooth (or at least $C^2$)
    - has a unique minimum at $q_{\text{goal}}$
    - is uniformly maximal on the boundary of the free space
    - is Morse

- A function is Morse if every critical point (a point where the gradient is zero) is isolated.

- Question: How?
Sphere Worlds: Definition

- Suppose that the world is a sphere of radius $R_0$ centered at $q_0$ containing $n$ obstacles of radius $r_i$ centered at $q_i$, $i=1..N$
  - $\beta_0(q) = -d^2(q,q_0) + R_0^2$
  - $\beta_i(q) = d^2(q,q_i) - r_i^2$

- Define $\beta(q) = \prod \beta_i(q)$ (Repulsive)
  - Note: This is 0 on any obstacle boundary, + in the free space, - inside an obstacle

- Define $\gamma_k(q) = (d(q,q_{goal}))^{2\kappa}$
  - Note: This is 0 at the goal and increasing as we move away from
  - $\kappa$ controls the growth
Sphere Worlds

- Consider \( \frac{\gamma \kappa}{\beta}(q) \)
  - This is 0 ONLY at the goal
  - This goes to \( \infty \) at the boundary of any obstacle
  - By increasing \( \kappa \), we can make the gradient at any direction point toward the goal
  - It is possible to show that the only stationary point is the goal with positive definite Hessian because \( \partial \gamma \kappa / \partial q \) dominates \( \partial \beta / \partial q \)
    - Therefore no local minima

- In short, following \(-\nabla \frac{\gamma \kappa}{\beta}(q)\) is guaranteed to get to the goal (for large enough value of \( \kappa \))
Sphere World: An Example

- One problem: the value of \( \frac{\gamma \kappa}{\beta}(q) \) may be very large

- A solution: introduce a “switching function”

\[
\sigma_\lambda(x) = \frac{x}{\lambda + x} \quad \lambda > 0
\]

- Now define \( s(q, \lambda) = \left( \sigma \circ \frac{\gamma \kappa}{\beta} \right)(q) = \left( \frac{\gamma \kappa}{\lambda \beta + \gamma \kappa} \right)(q) \)
  - This bounds the value of the function
  - HOWEVER, \( s(q, \lambda) \) may turn out NOT to be Morse
Sphere World: An Example

- A Solution: introduce a “sharpening function” \( \zeta_\kappa(x) = x^{\frac{1}{\kappa}} \)

\[
\varphi(q) = \left( \zeta_\kappa \circ \sigma_1 \circ \frac{\gamma_\kappa}{\beta} \right) = \frac{d^2(q, q_{goal})}{\left[ (d(q, q_{goal}))^{2\kappa} + \beta(q) \right]^{\frac{1}{\kappa}}}
\]

- For large enough \( \kappa \), this is a navigation function on a sphere world!!