Behavior-Based Paradigm, Part II

September 17, 2002

Class Meeting 8
Today’s Objectives

• Wrap-up: Motor schemas

• Potential Field Techniques

• Brief Review for Exam #1

• Makeup class rescheduling (for those who can’t make the makeup times)
Recall from Last Time: Motor Schemas

- Developed by Arkin in 1980s
- Based on biology’s schema theory
- Behavioral responses are all represented as vectors generated using a potential fields approach
- Coordination is achieved by vector addition

![Diagram showing the relationship between Sensory Input, BEHAVIOR, Perceptual Schema, Motor Schema, and Pattern of Motor Actions.]
Recall: Perception-Action Schema Relationships

Motor Schemas

Environmental Sensors

ES1

ES2

ES3

PS = Perceptual Schema
PSS = Perceptual Subschema
MS = Motor Schema
ES = Environmental Sensor

Vector \( \Sigma \)

Robot

Motors
Output of Motor Schemas Defined as Vectors

- **Output Vector**: consists of both orientation and magnitude components
- \( V_{\text{magnitude}} \) denotes magnitude of resultant response vector
- \( V_{\text{direction}} \) denotes orientation
Motors Schemas Achieve Behavioral Fusion via Vector Summation

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

Behavioral fusion
Recall: Defined Motor Schemas

- Move-ahead
- Move-to-goal
- Avoid-static-obstacle
- Dodge
- Escape
- Stay-on-path
- Noise
- Follow-the-leader
- Probe
- Dock
- Avoid-past
- Move up, move-down, maintain altitude
- Teleautonomy

Each of these can be defined as a potential field of output vector responses.
We’ll Come Back to Motor Schema Examples…
First: Potential Fields

**Introduction to Potential Fields:**

- Potential field: array (or field) of vectors representing space

- Vector \( \mathbf{v} = (m, d) \): consists of magnitude \( m \) and direction \( d \)

- Vector represents a force

- Typically drawn as an arrow:

  - Length of arrow = \( m \) = magnitude
  - Angle of arrow = \( d \) = direction

\[ V \text{ magnitude} \]
\[ V \text{ direction} \]
Potential Fields (con’t.)

• Vector space is 2D world, like bird’s eye view of map
• Map divided into squares, creating (x,y) grid
• Each element represents square of space
• Perceivable objects in world exert a force field on surrounding space
Five Primitive Potential Fields

Uniform

Perpendicular

Attraction

Repulsion

Tangential
Magnitude Profiles

• Change in velocity in different parts of the field

(See your text for 3D versions of these profiles)

Field closest to an attractor/repellor will be stronger
Programming a Single Potential Field

- Repulsive field with linear drop-off:

\[
\begin{align*}
V_{direction} &= 180^\circ \\
V_{magnitude} &= \begin{cases} 
\frac{(D - d)}{D} & \text{for } d \leq D \\
0 & \text{for } d > D
\end{cases}
\end{align*}
\]

where \(D\) is max range of field’s effect
typedef struct {
    double magnitude;
    double direction;
} vector;

vector repulsive(double d, double D)
{
    if (d <= D)
    {
        outputVector.direction = -180;  // turn around
        outputVector.magnitude = (D-d)/D;  // linear dropoff
    }
    else
    {
        outputVector.direction = 0.0;
        outputVector.magnitude = 0.0;
    }
    return outputVector;
}
Pseudocode for Runaway Behavior

```plaintext
vector runaway( )
{
    double reading;
    reading = readSonar();  // perceptual schema
    vector = repulsive(reading, MAX_DISTANCE);  // motor schema
    return Voutput;
}

while (robot == ON)
{
    Vrunaway = runaway(reading);  // motor schema
    turn(Vrunaway.direction);
    forward(Vrunaway.magnitude * MAX VELOCITY);
}
```
Important Note:

*Entire Field Does Not Have to Be Computed*

- Only portion of field affecting robot is computed
- Robot uses functions defining potential fields at its position to calculate component vector
Combining Fields/Behaviors

- Compute each behavior’s potential field
- Sum vectors at robot’s position to get resultant output vector
Issues with Combining Potential Fields

• Impact of update rates:
  – Lower update rates can lead to “jagged” paths

• Robot treated as point:
  ➔ Expect robot to change velocity and direction instantaneously (can’t happen)

• Local minima:
  – Vectors may sum to 0.
The Problem of Local Minima

• If robot reaches local minima, it will just sit still
Solutions for Dealing with Local Minima

• Inject noise, randomness:
  – “Bumps” robot out of minima

• Include “avoid-past” behavior:
  – Remembers where robot has been and attracts the robot to other places

• Use “Navigation Templates” (NaTs):
  – The “avoid” behavior receives as input the vector summed from other behaviors
  – Gives “avoid” behavior a preferred direction

• Insert tangential fields around obstacles
Return to Motor Schemas: Example Motor Schema Encodings

• Move-to-goal (ballistic):
  \[ V_{\text{magnitude}} = \text{fixed gain value} \]
  \[ V_{\text{direction}} = \text{towards perceived goal} \]

• Avoid-static-obstacle:
  \[
  V_{\text{magnitude}} = \begin{cases} 
  0 & \text{for } d > S \\
  \frac{S-d}{S-R} * G & \text{for } R < d \leq S \\
  \infty & \text{for } d \leq R 
  \end{cases}
  \]

  where
  \[ S = \text{sphere of influence of obstacle} \]
  \[ R = \text{radius of obstacle} \]
  \[ G = \text{gain} \]
  \[ d = \text{distance of robot to center of obstacle} \]
More Motor Schema Encodings

• Stay-on-path:

\[ V_{\text{magnitude}} = \begin{cases} 
  P & \text{for } d > (W/2) \\
  \frac{d}{W/2} \ast G & \text{for } d \leq (W/2) 
\end{cases} \]

where:

- \( W \) = width of path
- \( P \) = off-path gain
- \( G \) = on-path gain
- \( D \) = distance of robot to center of path

\( V_{\text{direction}} \) = along a line from robot to center of path, heading toward centerline
More Motor Schema Encodings (con’t.)

• Move-ahead:
  \( V_{\text{magnitude}} = \text{fixed gain value} \)
  \( V_{\text{direction}} = \text{specified compass direction} \)

• Noise:
  \( V_{\text{magnitude}} = \text{fixed gain value} \)
  \( V_{\text{direction}} = \text{random direction changed every } p \text{ time steps} \)
Motor Schema-Based Robots

Primarily at Georgia Tech:

- HARV
- George
- Ren and Stimpy
- Buzz
- Io, Callisto, Ganymede
- Mobile manipulator

Callisto
Example of Using One Motor Schema (Behavior) Per Sensor

- Khepera Robot:

  ![Image of Khepera Robot]

  = IR sensors
Consider “Box Canyon”

- Generate repulsive field for each sensor
- Sum the sensors to get the resultant direction

Result: Robot can back out of box canyon without needing “model” of walls
while (robot==ON)
{
    vector.mag = vector.dir = 0.0;  //initialize to 0
    for (i=0; i<=numberIR; i++)
    {
        vectorCurrent = Runaway(i);  //accept a sensor number
        vectorOutput = VectorSum(tempVector,vectorCurrent);
    }
    turn(vector.direction);
    forward(vector.magnitude*MAX VELOCITY);
}
Sequencing of Motor Schemas

• Can sequence motor schemas if one activity needs to be completed before another.
• Recall Foraging – FSA diagram:
Stimulus-Response Diagram for Schema-Based Foraging

FORAGE

WANDER
- Noise
- Avoid-static-obstacle
- Avoid-static-obstacle

ACQUIRE
- Move-to-goal
- Avoid-static-obstacle
- Avoid-static-obstacle
- Noise

DELIVER
- Move-to-goal
- Avoid-static-obstacle
- Avoid-static-obstacle
- Noise

Σ
Sequencer
Design in Motor-Schema-Based Systems

- Characterize problem in terms of motor behaviors needed to accomplish task
- Decompose motor behaviors to most primitive level, using biological studies when feasible
- Develop formulas to express robot’s reaction to perceived environmental events
- Conduct simple simulation studies to assess desired behaviors’ approximate performance
- Determine perceptual requirements needed to satisfy motor schema inputs
- Design specific perceptual algorithms
- Integrate control system onto robot
- Test and evaluate
- Iterate and expand as needed
Summary of Architectural Design Issues

• Analysis vs. Synthesis

• Top-down (knowledge-driven) vs. Bottom-up (data-driven) Design

• Domain relevance vs. Domain independence

• Understanding intelligence vs. Intelligent machines
The Class So Far (in 1 slide)...
Focus on Paradigms and Architectures

- History of Robotics
- Robot Software Design Strategies
- Robotic Paradigms and Architectures
  - Hierarchical
    - Nested Hierarchical Controller
    - NIST RCS
    - STRIPS
  - Reactive and Behavior-Based
    - Biological foundations
    - Subsumption
    - Motor-Schemas
    - Potential Fields
  - (Later: Hybrid)
    - Behavior encodings, combinations, etc.
    - Pros, cons, evaluation of various architectures
    - Programming implementations/pseudocodes of above
Exam #1

- Will cover all lecture notes (except today’s topics specific to potential fields)

- Will cover all extra readings (3 to date)

- Will cover Murphy text, chapters 1 – 4  EXCEPT section 4.4 (potential fields)
Reminder: Makeup Classes

• Meet 3:40 – 6:10:
  – Next Tuesday, September 24th
  – Thursday, October 3rd

• No class:
  – Thursday, September 26th
  – Tuesday, October 1st

• TODAY: Meet at end of class if you can’t make one of these extended classes!! We’ll pick another makeup time for these groups.