22. Fuzzy Logic

- Fuzzy Logic
- Odometry

“I think there is a world market for maybe five computers” Thomas Watson, chairman of IBM, 1943

Things that can go bad

- Hitting the wall
  - Think of three ways to tell if you hit the wall
  - Corrective measures
- Wrong-way Dayo
  - Think of ways to reduce the chances
  - Three repairs -> disqualification
- Other robots in the way
  - Can you distinguish a robot from a wall?
  - Strategy for passing

Comparison

- Incremental
  + Simple, stable
  - Slow response
- PID or PI
  + Theory, fast response
  - Needs empirical tuning, depends on load
- Fuzzy Logic
  Maps human intuition into rules
  + Fast, good when you have expert knowledge
  + Abstractive approach
  - Needs empirical tuning

Fuzzy Membership Set

- Membership set, variable, set
  - Value specifying levels of truth
  - Collection describes the entire system

Not at all...a little bit...somewhat...mostly...pretty much...definitely

- Examples for a speed control system
  - TooSlow
  - SlowingDown
  - SpeedOK
  - SpeedConstant
  - TooFast
  - SpeedUp
Fuzzy Membership Set

• Lab 7 example membership sets
  – Too close to the right wall
  – Distance to the right wall is ok
  – Too far away from the right wall
  – Too close to the left wall
  – Distance to the left wall is ok
  – Too far away from the left wall
  – Open space to 30 degrees to the right
  – Open space to straight ahead
  – Open space to 30 degrees to the left

Speed Controller

• Desired state
  – $X^*$ is the desired tach period

• Physical plant
  – $X$ real state variable, actual period

• State estimator, data acquisition
  – $X'$ measured tach period

Fuzzy approach

• Preprocessor
  – Crisp inputs (variables with units)

• Fuzzification
  – Input membership sets

• Fuzzy rules
  – Output membership sets

• Defuzzification
  – Crisp outputs (variables with units)

• Postprocessor and actuator output

Fuzzy approach

• Preprocessor, crisp inputs
  – $E = X^* - X'$ error in motor period
  – $D = X'(n) - X'(n-1)$ acceleration

```c
unsigned char Ts;  // Desired Speed in 3.9 rpm units
unsigned char T;   // Current Speed in 3.9 rpm units
unsigned char Told; // Previous Speed in 3.9 rpm units
char D;            // Change in Speed in 3.9 rpm/time units
char E;            // Error in Speed in 3.9 rpm units

void CrispInput(void){
  E=Subtract(Ts,T);
  D=Subtract(T,Told);
  Told=T;  /* Set up Told for next time */
}
```
Fuzzy approach

- Preprocessor, crisp inputs
  - $E = X^* - X'$ error in motor period
  - $D = X'(n) - X'(n-1)$ acceleration

- Fuzzification

  - **Slow**
    - True if the motor is spinning too slow
  - **OK**
    - True if the motor is spinning at the proper speed
  - **Fast**
    - True if the motor is spinning too fast
  - **Up**
    - True if the motor speed is getting larger
  - **Constant**
    - True if the motor speed is remaining the same
  - **Down**
    - True if the motor speed is getting smaller.

Fuzzification

```c
#define TE ???
#define TD ???
#define TN ???

void InputMembership( void ){
    if (E <= -TE) { /* E <= -TE */
        Slow = 255; OK = 0; Fast = 0;
    } else
        if (E < 0) { /* -TE < E < 0 */
            Slow = (255*(-E))/TE;
            OK = 255-Slow;
            Fast = 0;
        } else
            if (E < TE) { /* 0 < E < TE */
                Slow = 0;
                Fast = (255*E)/TE;
                OK = 255-Fast;
            } else { /* E >= TE */
                Slow = 0;
                Fast = 255;
            }
    }
    if (D <= -TD) { /* D <= -TD */
        Up = 255; Constant = 0; Down = 0;
    } else
        if (D < 0) { /* -TD < D < 0 */
            Up = (255*(-D))/TD;
            Constant = 255-Up;
            Down = 0;
        } else
            if (D < TD) { /* 0 < D < TD */
                Up = 0;
                Down = (255*D)/TD;
                Constant = 255-Down;
            } else { /* D >= TD */
                Up = 0;
                Down = 255;
                Constant = 0;
            }
}
```

Fuzzy rules

- If **OK** and **Constant** then **Same**
- If **OK** and **Up** then **Decrease**
- If **Fast** and **Constant** then **Decrease**
- If **Fast** and **Up** then **Decrease**
- If **OK** and **Down** then **Decrease**
- If **Slow** and **Constant** then **Increase**
- If **Slow** and **Down** then **Increase**
Fuzzy rules

\[
\begin{align*}
\text{Same} &= (\text{OK} \text{ and} \text{Constant}) \\
\text{Decrease} &= (\text{OK} \text{ and} \text{Fast} \text{ and} \text{Constant}) \text{ or} (\text{Fast} \text{ and} \text{Up}) \\
\text{Increase} &= (\text{OK} \text{ and} \text{Slow} \text{ and} \text{Constant}) \text{ or} (\text{Slow} \text{ and} \text{Down})
\end{align*}
\]

- and operation is minimum
- or operation is the maximum

\[
\begin{align*}
\text{unsigned char static min}(\text{unsigned char} \ u1, \text{unsigned char} \ u2) \{ \\
\text{if} (u1 > u2) \text{ return} (u2); \\
\text{else return} (u1);
\}
\]

\[
\begin{align*}
\text{unsigned char static max}(\text{unsigned char} \ u1, \text{unsigned char} \ u2) \{ \\
\text{if} (u1 < u2) \text{ return} (u2); \\
\text{else return} (u1);
\}
\]

Defuzzification

\[
\Delta N = \frac{\text{Decrease} \cdot (-TN) + \text{Same} \cdot 0 + \text{Increase} \cdot TN}{\text{Decrease} + \text{Same} + \text{Increase}}
\]

\[
\text{long } dN; \text{ void CrispOutput(void)}(\text{dN}=\text{TN} \cdot (\text{Increase} - \text{Decrease}) / (\text{Decrease} + \text{Same} + \text{Increase});)
\]

Fuzzy Logic Controller

\[
\text{void Timer0A_Handler(void)}(\text{T = SE(); \text{// estimate speed, set T, 0 to 255}} \\
\text{CrispInput(); \text{// Calculate E,D and new Told}} \\
\text{InputMembership(); \text{// Sets Fast,OK,Slow,Down,Constant,Up}} \\
\text{OutputMembership(); \text{// Sets Increase,Same,Decrease}} \\
\text{CrispOutput(); \text{// Sets dN}} \\
\text{N = max(0, min(N+dN, 255)); \text{// output to actuator, Program 8.4}} \\
\text{PWM0_Duty(N); \text{// output to actuator, Program 8.4}} \\
\text{TIMER0_ICR_R = TIMER_ICR_CAECINT; \text{// ack}}
\]
Odometry

Constants

- Number of slots/rotation, \( n=32 \)
- Wheel diameter, \( d=886 \) (0.01cm)
- Wheelbase, \( w=1651 \) (0.01cm)
- Wheel circumference, \( c=\pi d=2783 \) (0.01 cm)

Measurements

- LCount the number of left slots in \( \Delta t \)
- RCount the number of right slots in \( \Delta t \)
- Counts vary from -28 to +28 each \( \Delta t \)

Simple cases

- \(-28 \leq m \leq +28 \) each \( \Delta t \)

<table>
<thead>
<tr>
<th>LCount</th>
<th>RCount</th>
<th>Motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m )</td>
<td>( m )</td>
<td>straight line motion in the current direction</td>
</tr>
<tr>
<td>( 0 )</td>
<td>( m )</td>
<td>pivot about stopped left motor</td>
</tr>
<tr>
<td>( m )</td>
<td>( 0 )</td>
<td>pivot about stopped right motor</td>
</tr>
<tr>
<td>( m )</td>
<td>( -m )</td>
<td>pure rotation about cog</td>
</tr>
</tbody>
</table>
**Derivations**

- $c = \text{circumference}$
- $n = \text{number of slots per rotation}$
- $w = \text{wheelbase}$
- $L_l = L_{\text{Count}} \cdot c / n$ the arc distance traveled by the left wheel (0.01cm)
- $R_r = R_{\text{Count}} \cdot c / n$ the arc distance traveled by the right wheel (0.01cm)

Assume $L_l, R_r$ positive
Assume $R_r > L_l$

\[
\frac{L}{L_l} = \frac{(L + w)}{R_r}
\]

\[
\frac{L}{L_l} - \frac{L}{R_r} = \frac{w}{R_r}
\]

\[
L = \frac{w \cdot L_l}{R_r - L_l}
\]

\[d\theta = \frac{L_l}{L} = \frac{R_r}{(L + w)}\]

**Odometry**

- Needs very accurate sensors
- Errors accumulate
- OK for relative travel from known position
  - periodic absolute knowledge of position

\[
L_l = L_{\text{Count}} \cdot c / n \\
R_r = R_{\text{Count}} \cdot c / n \\
L = (w \cdot L_l) / (R_r - L_l) \\
d\theta = \frac{(100 \cdot L_l)}{L} \\
dz = \frac{((d\theta / 2) \cdot (L + w / 2))}{100} \\
x = x + dz \cdot \cos(\theta) \\
y = y + dz \cdot \sin(\theta) \\
\theta = \theta + d\theta \\
x = x + dz \cdot \cos(\theta) \\
y = y + dz \cdot \sin(\theta)
\]

**First part of move**

**Second part of move**