Graph-based formation with real e-pucks

Stéphane Bouquet
Florent-Valéry Coen
Stanislas Cottard
Formation Control

- Control equations depend on the Laplacian matrix $L$ of the connected graph

\[
L = I \cdot W \cdot I^T
\]

\[
u_i(t) = K_u \cdot \bar{e}_i(e_{i,j}(t), \alpha_{i,j}(t)) \cdot \cos(\bar{\alpha}_i(e_{i,j}(t), \alpha_{i,j}(t)))
\]

\[
\omega_i(t) = K_w \cdot \bar{\alpha}_i(e_{i,j}(t), \alpha_{i,j}(t))
\]

\[
\bar{e}_{x,i}(t) = \sum_{j=1}^{\Delta_i} \left[-L_{i,j} \cdot (e_{i,j}(t) \cdot -b_{i,j}) \cos(\alpha_{i,j}(t))\right]
\]

\[
\bar{e}_{y,i}(t) = \sum_{j=1}^{\Delta_i} \left[-L_{i,j} \cdot (e_{i,j}(t) - b_{i,j}) \sin(\alpha_{i,j}(t))\right]
\]

\[
\bar{e}_i(e_{i,j}(t), \alpha_{i,j}(t)) = \sqrt{\bar{e}_{x,i}(t)^2 + \bar{e}_{y,i}(t)^2}
\]

\[
\bar{\alpha}_i(e_{i,j}(t), \alpha_{i,j}(t)) = \text{arctan} 2(\bar{e}_{y,i}(t), \bar{e}_{x,i}(t))
\]
Webots Implementation

• Advantages: messages buffer and automatic ID

• Communication via ping messages

```
// send a ping message
void send_ping(void)
{
    char out[10];
    strcpy(out, robot_name); // in the ping message we send the name of the robot.
    wb_emitter_send(emitter2, out, strlen(out)+1);
}
```

```
void process_received_ping_messages(void)
{
    ...
    while (wb_receiver_get_queue_length(receiver2) > 0)
    {
        inbuffer = (char*) wb_receiver_get_data(receiver2);
        message_direction = wb_receiver_get_emitter_direction(receiver2);
        message_rssi = wb_receiver_get_signal_strength(receiver2);
        ...
        wb_receiver_next_packet(receiver2);
    }
}
```

• Typical message:
  – Direction vector \( <x, y> \)
  • Used to compute theta
  – Signal strength
  • Distance from receiver to emitter = \( 1/ \sqrt{\text{signal strength}} \)
e-puck Implementation

Communication

• e-pucks can not receive and send messages at the same time
• Balance between listening and emitting state
• The e-puck listens during 25 iterations and then sends a message with its ID

```c
listen_time++;

// ircomListen();
IrcomMessage imsg;
ircomPopMessage(&imsg);

if (imsg.error == 0)
{
   int val = (int) imsg.value;
   ...
}
else if (imsg.error > 0)
{
   btcomSendString("Receive failed \n");
}

// no message available in the queue for 25 iterations = sending
else if (imsg.error == -1 && listen_time >= 25)
{
   // to avoid perfect sync listen_time is reset at 0 + rand()
   listen_time = 0 + rand()%12+1;
   for(i=0;i<1;i++)
   {
      for(j = 0; j < 11000; j++)asm("nop ");
      ircomSend(robot_id);
      while (ircomSendDone() == 0);
   }
```
e-puck Implementation

Distributed ID Attribution

- Unlike in Webots, each e-pucks isn’t automatically associated with its own unique ID
- Algorithm to exchange their ID based on their initial position
- Assumption of an initial square-shaped oriented formation
- Direction of the incoming messages is analysed to decide the position of the robot leading its ID recognition

<table>
<thead>
<tr>
<th>Sensor</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0.297</td>
<td>0.873</td>
<td>1.571</td>
<td>2.618</td>
<td>3.666</td>
<td>4.712</td>
<td>5.411</td>
<td>5.986</td>
</tr>
</tbody>
</table>
e-puck Implementation

Distributed ID Performance

Initialization stats

- Initialization speed proportional to the color intensity

Number of messages:
- 200
- 300
- 400
- 500
e-puck Implementation

Formation Control
• Almost like Webots, only the communication changes
• e-puck receives messages with the ID of the emitter
• Update its relative position array as soon as possible
• Message format
  – Value
    • Other robot ID
  – Distance (range)
  – Direction
    • Used directly as theta
• Theta and the range are directly received, unlike on Webots
Performances

Fitness function

- Reward a good forward speed $u$
- Penalise a poor formation
  - Bad distances between the robots
    - More weight on vertices between a robot and the leader
  - Penalise too high rotational speed $\omega$
    - Synonymous to a correction of the robot alignment
- Normalized by the worst case
  - Distance greater than 20 cm, small $u$ and high $\omega$

$$fitness = 1 - \left( \frac{\sum_{i=1}^{4} \frac{range[i] - B[id][i]}{20 - B[id][i]} \cdot 10 \cdot \frac{range[0] - B[id][0]}{(20 - B[id][0])} \cdot (u - 1.5)}{3.1538 \cdot 1.5} \right)$$
Performances

Parameters

• Parameters are equals in both implementations to allow a comparison
  
  – Ku = 5
  – Kw = 0.3
  
  – The leader has no rotational speed and a forward speed almost constant although slightly dependent to the computed u
  – Distance between robots = 12 cm or 12*Sqrt(2) in the diagonal

```c
// Give the leader a straight speed to move the formation
if(robot_id == 0)
{
    u = 0.05*u + 1.5;
    w = 0;
}
```

– The Laplacian matrix was set as

\[
L = \begin{bmatrix}
  3.7 & -1.3 & -1.1 & -1.3 \\
  -1.3 & 3.1 & -1 & -0.8 \\
  -1.1 & -1 & 3.1 & -1 \\
  -1.3 & -0.8 & -1 & 3.1
\end{bmatrix}
\]
Performances

Results

• Runs of $t = 60 \text{ sec}$
• Big difference between real e-pucks and Webots simulation
• Stabilisation at $t = 10$ on Webot, real e-pucks fitness increases as well
Performances

Results

• The fitness of e-puck 2 varies a lot, especially near the end of the 60 seconds

• Due to its position on the diagonal, it is more likely for him to loose some messages and be inaccurate in its positioning

• Difference curve more unstable comparing to the others
Thank you