Optimized simulated flocking algorithm for e-pucks

Team paard:  - Diego Antognini
            - Damien Doy
            - Martin d’Hoffschmidt
Plan

Goal of the project
Reynolds Limited
Kikker controller
Metrics
PSO
Introduction

Goal: implement a scalable, efficient and robust flocking controller

We started with the Reynolds controller from the laboratory 4

Used unlimited range communication, not scalable
Unlimited communication
Limited range communication
Reynolds limited

Needs to know a global state of the robots

A robot should know the relative position of every other robots
Reynolds limited: algorithm

- send own broadcast with id, position(0, 0) and timestamp+1
- when receive message, if timestamp stored ≤ timestamp received ignore
- timestamp stored ← timestamp received
- calculate position emitter, add with position in message
- broadcast message with timestamp, id and relative position
Limited range communication

Broadcast message: pos(0.0, 0.0), id=0, pos(0.0, 0.1), id=0, pos(0.0, 0.2), id=0, pos(0.0, 0.3), id=0

Robot id: 0 1 2 3
Scalability: Reynolds limited

We limited the range of the robots to the bare minimum (10 cm, Ø of epuck)

Less communication, less computation, more battery

Same performance as unlimited Reynolds
Demos: Reynolds limited communication
Kikker controller: body-fixed reference frame

- Real axis $\rightarrow$ x axis, front
- Imaginary axis $\rightarrow$ y axis, left
- Each e-puck equipped with a bearing
- Able to compute distance to another e-puck, range
Kikker controller: desired direction

- Compute the direction we want to go to

\[ \mathbf{a} = \frac{\mathbf{h} + \beta \mathbf{b} + \gamma \mathbf{c} + \delta \mathbf{d}}{\| \mathbf{h} + \beta \mathbf{b} + \gamma \mathbf{c} + \delta \mathbf{d} \|} \]

- Heading direction vector \( \rightarrow \) \( \mathbf{h} \)
- Proximal control vector \( \rightarrow \) \( \mathbf{b} \)
- Cohesion control vector \( \rightarrow \) \( \mathbf{c} \)
- Migration control vector \( \rightarrow \) \( \mathbf{d} \)
Kikker controller: heading direction

- Average sum of the alignment with the other robots

\[ h = \frac{\sum_{k \in \mathcal{N}_r} \exp(j \theta_k)}{\| \sum_{k \in \mathcal{N}_r} \exp(j \theta_k) \|} \]

- The angle is computed with regard to the north
- Each robot's transmits it own angle
- Communication can either be global or local
Kikker controller: cohesion controller

- Cohesion control vector computed as

\[ \mathbf{c} = \sum_{j \in \mathcal{N}_r} g_j \exp(j \psi_j) \]

- Compute the angle and distance of the \( j \text{th} \) neighbour robot

- Compute the virtual force with regard to the reference distance

\[ g_j = \begin{cases} + (d_k - d_{ref})^2 & \text{if } d_k \geq d_{ref} \\ - (d_k - d_{ref})^2 & \text{if } d_k < d_{ref} \end{cases} \]
Kikker controller: motion control

- **Forward velocity**
  \[ u = \begin{cases} 
  a \cdot a_c & \text{if } a \cdot a_c \geq 0 \\
  0 & \text{otherwise} 
  \end{cases} \]

- **Angular velocity**
  \[ \omega = K_p (\beta - \alpha) \]

- **Proportional controller to reach desired heading**
Kikker controller: obstacles video
Metrics

- **Orientation**

\[ o[t] = \frac{1}{N} \left| \sum_{k=1}^{N} \exp(j \psi[t]) \right| \]

- **Velocity**

\[ v[t] = \frac{1}{v_{max}} \max \left[ \text{proj}_\Phi (\bar{x}[t] - \bar{x}[t - 1]) \right] \]

- **Entropy**

\[ s[t] = \int_{0}^{\infty} \sum_{k=1}^{M} p_k(h) \log_2(p_k(h)) dh \]

Average orientation

Proportion of robots in cluster K of max distance of h

Average velocity of the center of mass along the direction of the migration urge
Metrics

- Instant performance
  \[ p[t] = o[t] v[t] s[t] \]

- Overall performance
  \[ \bar{p} = \frac{1}{T} \sum_{t=1}^{T} p[t] \]
PSO - What to optimize

Braitenberg with neural networks (22 parameters)

Direct braitenberg weight (12 parameters)

Kikker controller (5 parameters - normalized)

<table>
<thead>
<tr>
<th>Name</th>
<th>lower bound</th>
<th>upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.01</td>
<td>2.0</td>
</tr>
<tr>
<td>$K_p$</td>
<td>0.1</td>
<td>1.5</td>
</tr>
<tr>
<td>$d_{ref}$</td>
<td>0.05</td>
<td>0.50</td>
</tr>
</tbody>
</table>
PSO - How we optimize

Noise resistance : multiple evaluation of a run (avg)

Fitness \[ p = \frac{1}{T} \sum_{t=1}^{T} p(t) \]

Last of a run \( \sim 5 \) minutes in Webots

15 particles
PSO - Results

Braitenberg neural network: Didn’t converge after \( \sim 20 \) iterations

Braitenberg direct weight: Didn’t converge after \( \sim 20 \) iterations

Kikker controller: Didn’t converge but get interesting results

<table>
<thead>
<tr>
<th>Parameters</th>
<th>performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.124 \pm 0.018</td>
</tr>
<tr>
<td>( \beta = .38, \gamma = .33, \delta = .69, K_p = 1.00, d_{ref} = .34 )</td>
<td>0.100 \pm 0.020</td>
</tr>
<tr>
<td>( \beta = .38, \gamma = .29, \delta = .87, K_p = .94, d_{ref} = .36 )</td>
<td>0.111 \pm 0.020</td>
</tr>
</tbody>
</table>
Conclusion

Unlimited Reynolds: not implementable on real robots

Limited Reynolds: saves computation, saves energy, more scalable

Kikker: another approach with interesting results

PSO: Difficult and very long to optimize
Thank you for your attention

Questions ?
Kikker controller: proximal control

- Proximal control vector computed as

\[ \mathbf{p} = \frac{1}{N_s} \sum_{k=1}^{N_s} f_k \exp(j\phi_k) \]

- Compute the virtual force with IR sensor values

- Direction of the \( kth \) virtual force with respect to the \( kth \) sensor

\[ f_k = -\frac{(o_k)^2}{C} \]