Event Handling
With Real E-Pucks Using Threshold-Based Task Allocation

DISTRIBUTED INTELLIGENT SYSTEMS

D. Borgeaud, C. Dutertre, N. Rey
The Task

- Handle as many events as possible during a fixed amount of time.
- Cost: energy for the robots to move
- No communication

Events are materialised by blue cylinders that robots have to reach.
Threshold-Based Algorithm

- Robots go for a task only if the associated stimulus is higher than a threshold
- No communication => local stimulus.
- Each robot scans the environment with its camera
- Stimulus = amount of blue found in an image
Finite State Machine

SCAN

Event detected

Event lost

Event handled

WORK
Scan State

● Idea: as soon as a stimulus < threshold is detected: go to WORK state

● Robot rotates on itself and takes pictures to scan the environment

● Small capacity: keep only the middle 1px line of the image

● **Stimulus** = max number of contiguous blue pixels in an image, divided by the image width
Scan State

- Noise reduction:  

- Threshold continuously decreases with time
Work State

- Robot goes for the event

- Correct its trajectory:
  - object on the left of the image => turn left
  - object on the right of the image => turn right

- Collision avoidance using Braitenberg (walls and other robots)
Robots stop and blink when they handle an event.

Capacity of the camera and sensibility of the proximity sensors didn’t allowed us to implement it in a satisfactory manner.
Webots Implementation

1. Capture horizontal lines of 52 pixels

2. Process the line to determine if a pixel is blue or not
   If the biggest component of the pixel is blue, it is considered as blue

1. Determine the largest object in view

2. Compute the stimulus associated to the object
   size of the object / image width
Webots Experimental Setup

- 5 e-pucks
- 2m x 2m arena
- 10 events
- 240 seconds
- 50 repetitions per threshold
- Threshold range (0.025 step): [0.10, 0.25]
- Base-line: threshold = 0
Webots Simulations (1)
Webots Simulations (2)
Webots Simulations (3)
Real Experimental Setup

- 3 e-pucks
- 1.2m x 0.6m arena
- Arena separated in 63 boxes
- 8 events
Adaptation To Real World

Limited computation capacity
- Reduced image frame rate
- Reduced angle of view
  ➔ Reduced velocity

Noisy measurements
- On both the camera and proximity sensors

Smaller arena
- Reduced number of events
Main issue: the camera

Tried different capturing methods

- 120 pixels horizontal line
- 60 pixels horizontal line with 2x zoom
- 30 pixels horizontal line with 5x zoom

→ No improvement (Quality and size of the image)

Reduced angle of view → loses sight of cylinders easily.

Main issue is that there is basically 2 states, 
“cylinder in view, stimulus = 1” or 
“No cylinder in view, stimulus = 0”.
**Detection distances**

<table>
<thead>
<tr>
<th>Distance</th>
<th>Success rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 cm</td>
<td>100 %</td>
</tr>
<tr>
<td>15 cm</td>
<td>80 %</td>
</tr>
<tr>
<td>20 cm</td>
<td>20 %</td>
</tr>
</tbody>
</table>
Demonstration / video
Conclusion

Successful implementation of a threshold based algorithm in Webots
→ Detect events in a satisfactory manner
→ Correct trajectory in function of the object position
→ Able to compare different threshold value

Very challenging to adapt the webot controller to a e-puck.
→ Threshold useless due to the limitation of the camera
→ E-puck are still able to navigate in the environment
Questions ?