Sound detection with a real e-puck

Steve Hottinger, Sebastian Strobl, Gaël Germano
EPFL, 7th June 2016
1. Introduction

2. Goals

3. Methodology
   - General equations
     - Phase shift method
     - Bearing angle method
   - Webots
   - Real e-puck

4. Implementation on the Webots / real e-puck

5. Results

6. Difficulties

7. Conclusion
Introduction

• SIS course project

• Signal processing to localize sound sources
  • Example of applications:
    - To look for people lost in the mountains
    - To spot mosquitoes and kill them with laser (Lenz, 2009)
Goals

• Capture and analyze sound
• Move to the sound source
• Stop when it finds the sound source
• Simulation to real robot transition
General equations to localize sound source

Phase shift method

How to compute $\Delta \phi$?
How to relate it to $\theta$?
General equations to localize sound source

Phase shift method

Signal form:

\[ x(t) = \sum_{j=0}^{N-1} R_j \ast \cos(2\pi f_j \ast t + \varphi_j) \]

The phase at microphone i for the sound source frequency (index k):

\[ \varphi_{k,i} = \tan^{-1} \left( \frac{B}{A} \right) \]

With B the imaginary part of Fourier coefficient and A the real part

Phase shift between microphones:

\[ \Delta \varphi_{1,0} = \varphi_1 - \varphi_0 \]
\[ \Delta \varphi_{2,0} = \varphi_2 - \varphi_0 \]
\[ \Delta \varphi_{2,1} = \varphi_2 - \varphi_1 \]

In this example:

\[ \Delta \varphi_{2,1} = \varphi_2 - \varphi_1 = 0 \]
\[ \Delta \varphi_{1,0} = \varphi_1 - \varphi_0 > 0 \]
General equations to localize sound source

Bearing angle method

How to compute $\Delta \phi$ ?
How to relate it to $\theta$ ?
General equations to localize sound source

Bearing angle method

Bearing angle: angle between e-puck forward direction and sound source.

\[
\cos \theta = \frac{c \Delta T}{d} = \frac{c \Delta \phi}{2\pi f d} \quad \text{as} \quad \Delta T = \frac{\Delta \phi}{\omega}
\]

with : \(\Delta \phi = \phi_i - \phi_j\) calculated before

So:

\[
\theta = \cos^{-1} \left( \frac{c \Delta \phi}{2\pi f d} \right)
\]
As we can see in the figure 5, the index 4 is the one with the higher magnitude. Since we have no other source of sound, this corresponds to the sound emitted by the sound source.

On the controller, the sound is processed with a Fast Fourier Transform (FFT). The phase is calculated like developed in equation 11.

To find the good conditions to give order to the wheels, we follow the scheme from Chapter III. An empirical analysis using Matlab to systematize the procedures is done to find the conditions. The two following conditions must be fulfilled to move the e-puck.

<table>
<thead>
<tr>
<th>Move</th>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Speed left</th>
<th>Speed right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right</td>
<td>$1.1 &lt; \theta_{1,2} &lt; 1.6$</td>
<td>$1.2 &lt; \theta_{2,0} &lt; 1.7$</td>
<td>750</td>
<td>250</td>
</tr>
<tr>
<td>Left</td>
<td>$0.8 &lt; \theta_{0,1} &lt; 1.3$</td>
<td>$1.6 &lt; \theta_{2,0} &lt; 2.0$</td>
<td>250</td>
<td>750</td>
</tr>
<tr>
<td>In front</td>
<td>$1.4 &lt; \theta_{1,2} &lt; 1.7$</td>
<td>$1.7 &lt; \theta_{2,0} &lt; 2.0$</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Behind</td>
<td>$1.2 &lt; \theta_{1,2} &lt; 1.7$</td>
<td>$1.2 &lt; \theta_{2,0} &lt; 1.6$</td>
<td>-250</td>
<td>250</td>
</tr>
</tbody>
</table>

$ f_k = \frac{k \times f_s}{N} $

Sampling frequency $f_s = 33’000$
Size of the sample, $N = 128$
Methodology – e-puck

\[ f_k = \frac{k \cdot f_s}{N} \]

Sampling frequency \( f_s = 16'384 \)
Size of the sample, \( N = 256 \)
On the Webots simulator, it is hard to do better when the sound source is behind. The sound takes over, and the robot might indeed go even faster than real time if they are not to be synchronised between the different microphone sources.

However when we tried to implement the same method on the e-puck, there were a lot of NaN values in a Matlab code. These values come because of the bad implementation of the microphone phase shift as explained in Chapter IV. A third condition is used to move the robot in the case when there are some possible improvements.

Methodology – e-puck

Matlab code to find conditions

<table>
<thead>
<tr>
<th>Move</th>
<th>Condition</th>
<th>Speed left wheel</th>
<th>Speed right wheel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>$dphi_{1,0} &lt; 0$</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td>Right</td>
<td>$dphi_{1,0} &gt; 0$</td>
<td>1000</td>
<td>500</td>
</tr>
<tr>
<td>Right</td>
<td>Else</td>
<td>500</td>
<td>-500</td>
</tr>
</tbody>
</table>
Methodology – e-puck

Stopping the e-puck when it finds the sound source

```c
for (m=0; m<NB_SENSORS; m++)
{
    ds_value[m] = e_get_prox(sensor[m]);
}
if(e_get_prox(0)>300 || e_get_prox(7)>300 || e_get_prox(1)>300 || e_get_prox(6)>300)
{
    e_set_speed_left(0);
    e_set_speed_right(0);
    for (i=0; i<8; i++) //8
    {
        e_set_led(i, 1);
        wait(10000);
        e_set_led(i, 0);
        wait(10000);
    }
}
else
{
    for (i=0; i<8; i++)
    {
        e_set_led(i, 0);
    }
}
```
Results Webots
Results e-puck

Let's show!
Difficulties

1. Understand well equations

2. Implementation on the real e-puck
   • Finding the good frequency
   • NaN problems
   • Theta conditions problem
Conclusion

• Difference between simulation and real life implementation

• Possibility to find the good frequency by itself

• Possibility to make the difference between an obstacle and the sound source

• Satisfaction when it finally work!
References


Questions ???