Sound detection on Webots and with a real e-puck

Leïla Fleury, Léa Pascal & Roxane Pott

02.06.2015
Outline

1) Goal

2) Procedure

3) Difficulties encountered

4) Improvements

5) Conclusion
1) Goal

Finding a sound source with an e-puck on Webots and in the real world

- Sound detection
- Sound localization
- Travel to the sound

→ In a quick and smooth way
2) Procedure

2.1) Sound recording

2.2) Fourier Transform

2.3) Determination of the sound source direction using phase shift

2.4) Dance

2.5) NaN values
2.1) Sound recording

- Sound sensors

- Implementation on e-puck

```
e_ad_scan_on();
while(!e_ad_is_array_filled());
e_ad_scan_off();
```

2.1) Sound recording

Plot of the registered signal in the time domain
2.2) Fourier Transform
Implementations

// subtract mean from signal
e_subtract_mean(e_mic_scan[k], FFT_BLOCK_LENGTH, LOG2_BLOCK_LENGTH);

// Copy vector into sigCmpx
e_fast_copy(e_mic_scan[k], (int*)sigCmpx, FFT_BLOCK_LENGTH);

// Do FFT. The result is stored in sigCmpx.
e_doFFT_asm(sigCmpx);
2.2) Fourier Transform

Index

\[ f = \frac{\text{index} \times \text{sampling frequency}}{\text{signal length}} \]

On Webots: index=4

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Webots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling frequency [Hz]</td>
<td>33000</td>
</tr>
<tr>
<td>Signal length</td>
<td>128</td>
</tr>
<tr>
<td>Theoretical frequency [Hz]</td>
<td>1030</td>
</tr>
<tr>
<td>Emitted frequency [Hz]</td>
<td>1031.25</td>
</tr>
</tbody>
</table>

On e-puck: index=22

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>E-puck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling frequency of one microphone [Hz]</td>
<td>16834/3</td>
</tr>
<tr>
<td>Signal length</td>
<td>256</td>
</tr>
<tr>
<td>Theoretical frequency [Hz]</td>
<td>440</td>
</tr>
<tr>
<td>Emitted frequency [Hz]</td>
<td>469.33</td>
</tr>
</tbody>
</table>
2.2) Fourier Transform

Plot of the registered signal by the e-puck in the frequency domain

$\rightarrow$ Emitted frequency = 469.33 [Hz]
2.3) Determination of the sound source direction

- The three microphones receive the signal with a phase delay depending on their distance from the source.
- The greater the phase is, the further the source is.
- By comparing two phases, we can determine the direction of the sound source.

2.3) Determination of the sound source direction

Phase calculation

• We use the following equation:

$$\varphi = \tan^{-1} \frac{Im}{Re}$$

• In the C-code, this is expressed by the function atan2():

```c
phi[k] = atan2(sigCmpx[index].imag, sigCmpx[index].real);
```
2.3) Determination of the sound source direction

**Phase shift**

- Phase shift between microphone 1 and 0 mainly used
- The source is to the left $\rightarrow$ the robot turns to the left
- The source is to the right $\rightarrow$ the robot turns to the right
- The source is in front of the robot $\rightarrow$ go straightforward
- The source is behind $\rightarrow$ the robot turns around

\[
\Delta \varphi_{10} = \varphi_0 - \varphi_1
\]

\[
\begin{align*}
\Delta \varphi_{10} & > 0 \\
\Delta \varphi_{10} & < 0 \\
\Delta \varphi_{10} & = 0 \\
\Delta \varphi_{10} & = NaN
\end{align*}
\]

2.3) Determination of the sound source direction

Phase shift

\[ d\phi_{10} = \text{NaN} \]

\[ d\phi_{10} > 0 \]

\[ d\phi_{10} = 0 \]

\[ d\phi_{10} < 0 \]

2.3) Determination of the sound source direction

Phase shift

WEBOTS

if \((d\phi_{10} < 0.2 \land d\phi_{10} > -0.2 \land d\phi_{20} < -0.2)\)
//if the source is just in front
{
    wb_differential_wheels_set_speed(1000, 1000);
}
else if \((d\phi_{10} < 0.2 \land d\phi_{10} > -0.2 \land d\phi_{20} > 0.2)\)
//if the source is just behind
{
    wb_differential_wheels_set_speed(-1000, 1000);
}
2.4) Dance

if (e_get_prox(0) > 300 || e_get_prox(7) > 300 || e_get_prox(1) > 300 || e_get_prox(6) > 300)
{
    // stop near the sound source and without collision
    e_set_speed_left(0);
    e_set_speed_right(0);
    dance_time = 1;
}

if (dance_time) {
    e_set_speed_left(-1000); // turn around
    e_set_speed_right(1000);

    for (i=0; i<8; i++) {
        // switch on all the leds
        e_set_led(i,1);
        sprintf(buffer,"Dance time !\n\n\n");
        e_send_uart1_char(buffer, strlen(buffer));
        while(e_uart1_sending());
    }
}
2.5) NaN values

double no_sound[3];

    for(k=0;k<3;k++)
    {  no_sound[k]=0;  }

while(no_sound[0] == 0 || no_sound[1] == 0 || no_sound[2] == 0)
    {
        [...] // Register the sound, do the FFT, calculate the phase phi[k]
        if(phi[k] - phi[k] == 0){
            no_sound[k] = 1; }
        else {
            no_sound[k]=0;
        }
    } // Don’t move before one value is different of NaN
    e_set_speed_left(0);
    e_set_speed_right(0);
3) Difficulties encountered

- Hardware difficulties
- Incoherencies about phase method
- Very short distance for sound perception
- Bluetooth quite unstable
- Index → trial-and-error method first
4) Improvements

• Using the bearing angle
• Smoother trajectory :
  – determine quadrants as a function of the value of the phase shift
• NaN solutions : conditions and filters
• Find a method to improve the phase coherence or reduce the number of decisions made by the robot → average on several phase calculation
5) Conclusion

• The e-puck finds the source on Webots and in the real world

• The challenging part was the real e-puck implementation
  – variability of the phase
  – noisy environment
  – ...

• We learned/get better ...
  ... how to deal with simulated and real implementation
  ... in C coding
  ... in signal analysis
References

• http://www.e-puck.org/

• https://www.cyberbotics.com/overview

Thank you for your attention!