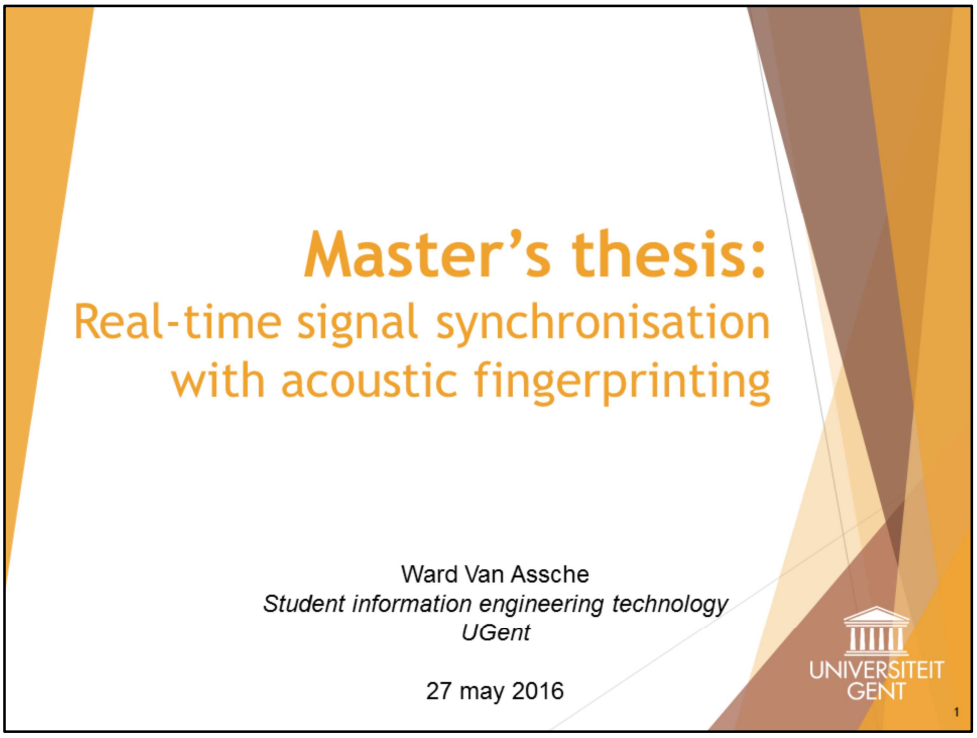
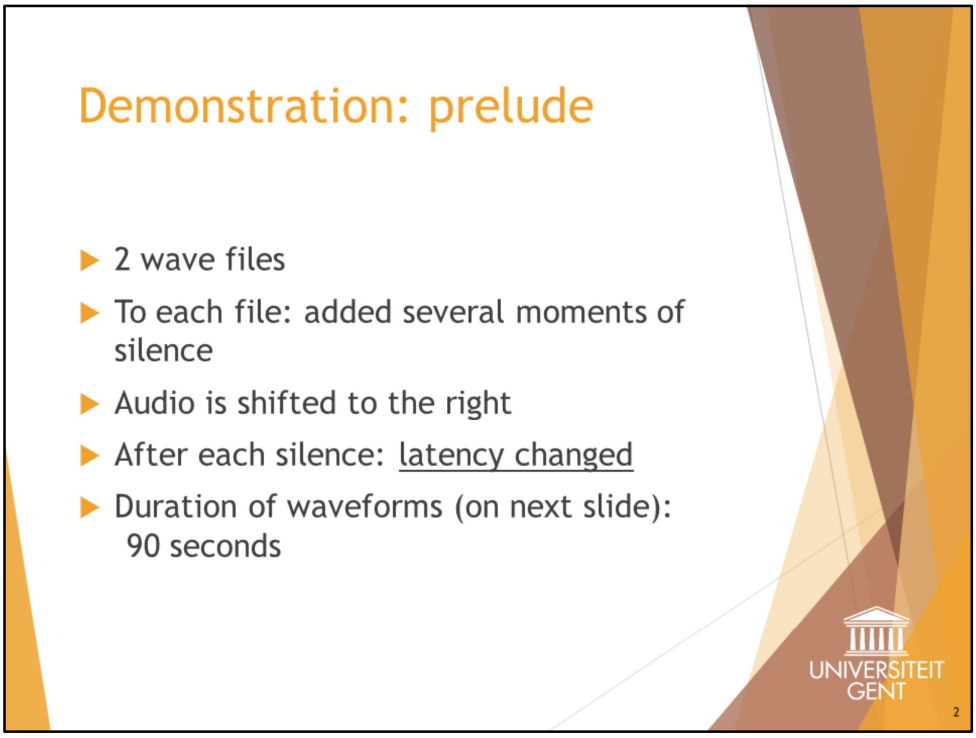
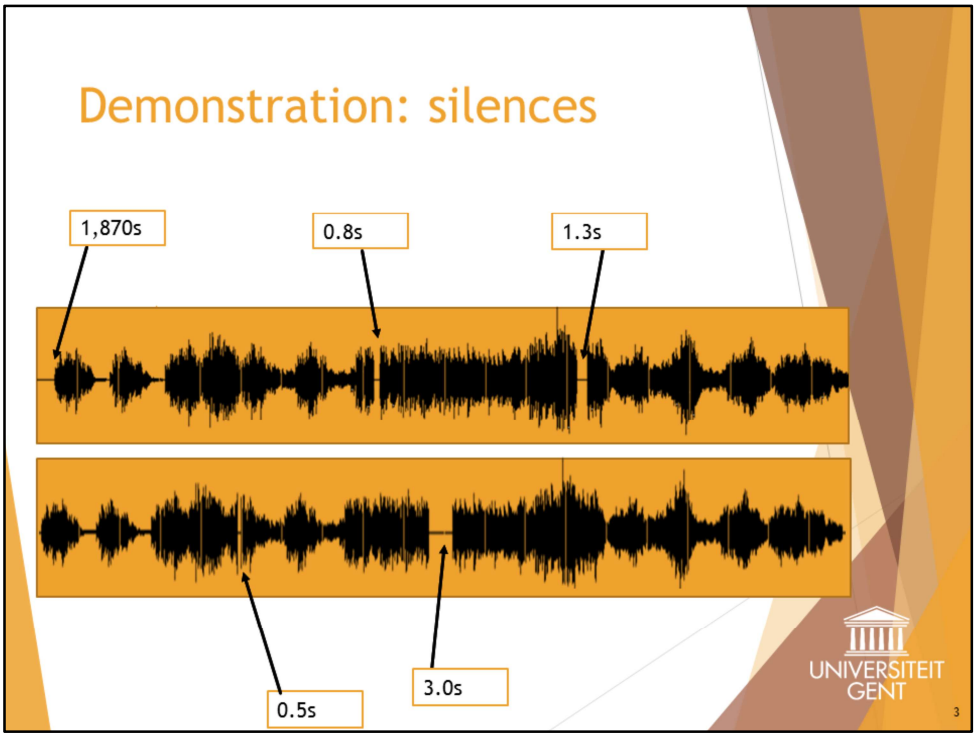


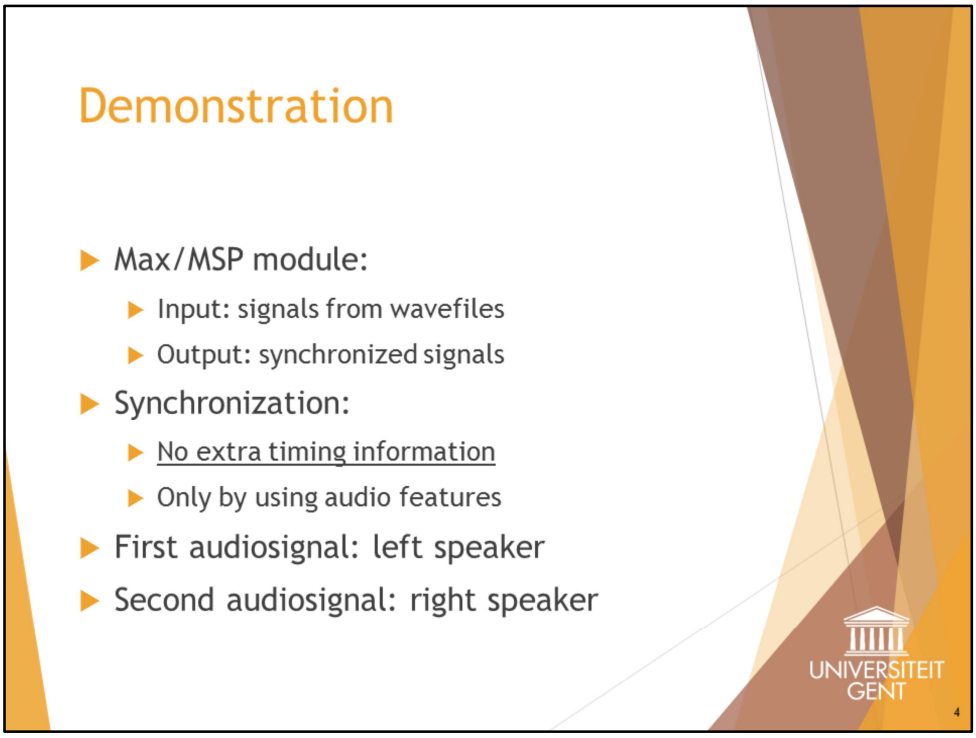
Hi, my name is Ward Van Assche and I'm going explain to you what I've created in the research for my Master's thesis.
I'm going to start with a small demonstration. Firstly I will sumarize what you are going to see during the demo.



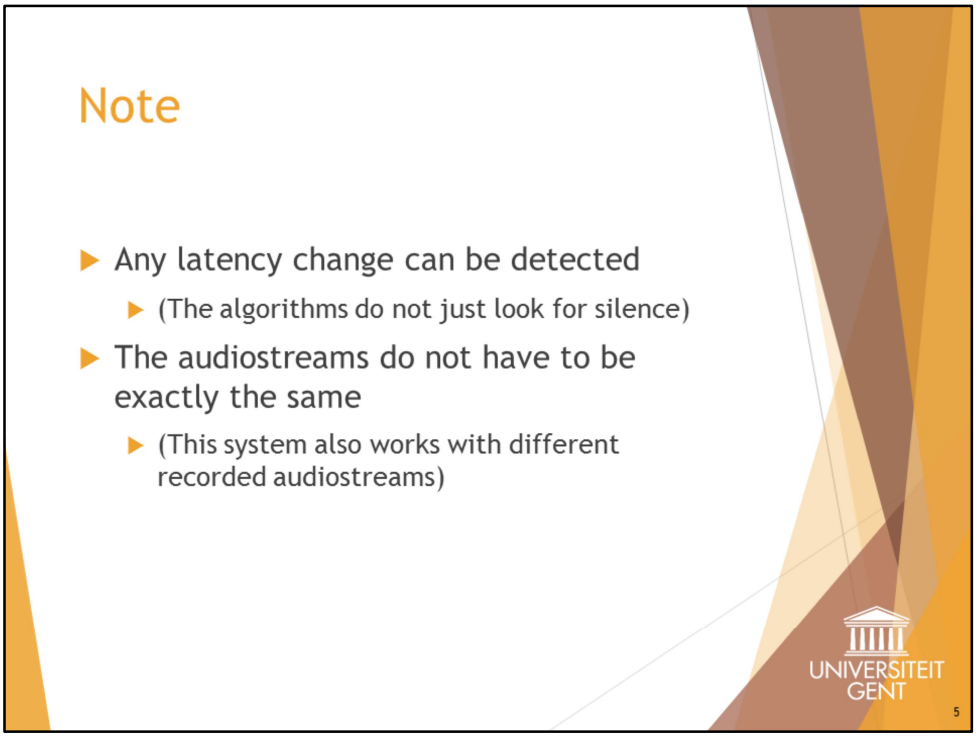
For the demo I took two wavefiles and added some silences at random places in both files. After adding the silences the sound is shifted to the right, no samples are lost. Because of this the latency between the files will change after each silence.



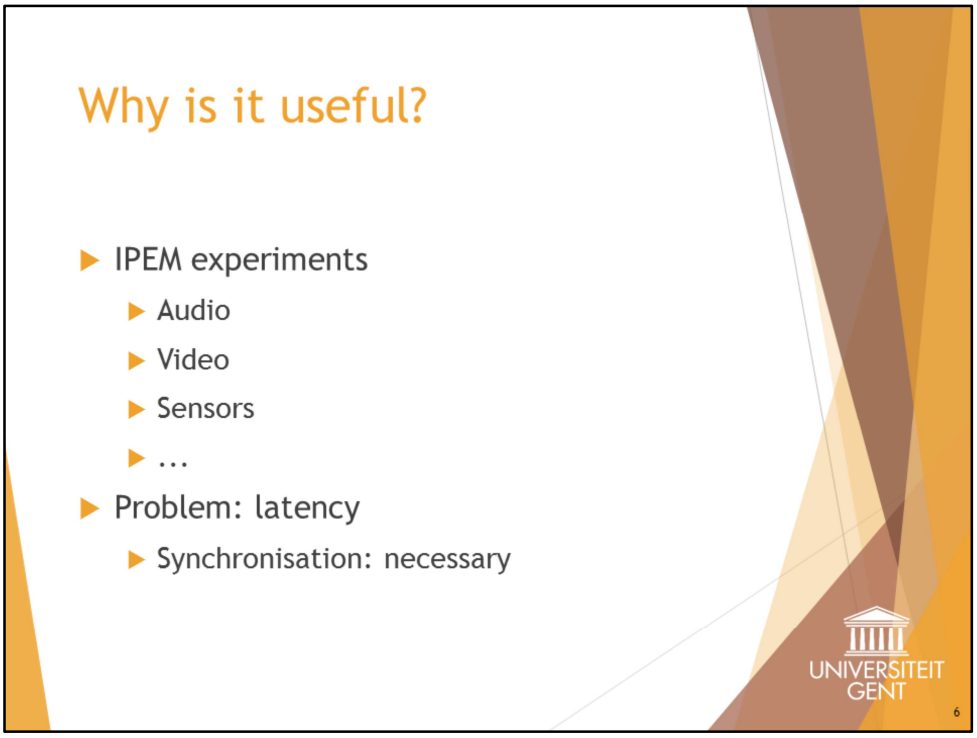
On this slice you can see the added silences. You can use this slide to have an idea what is happening during the demo.



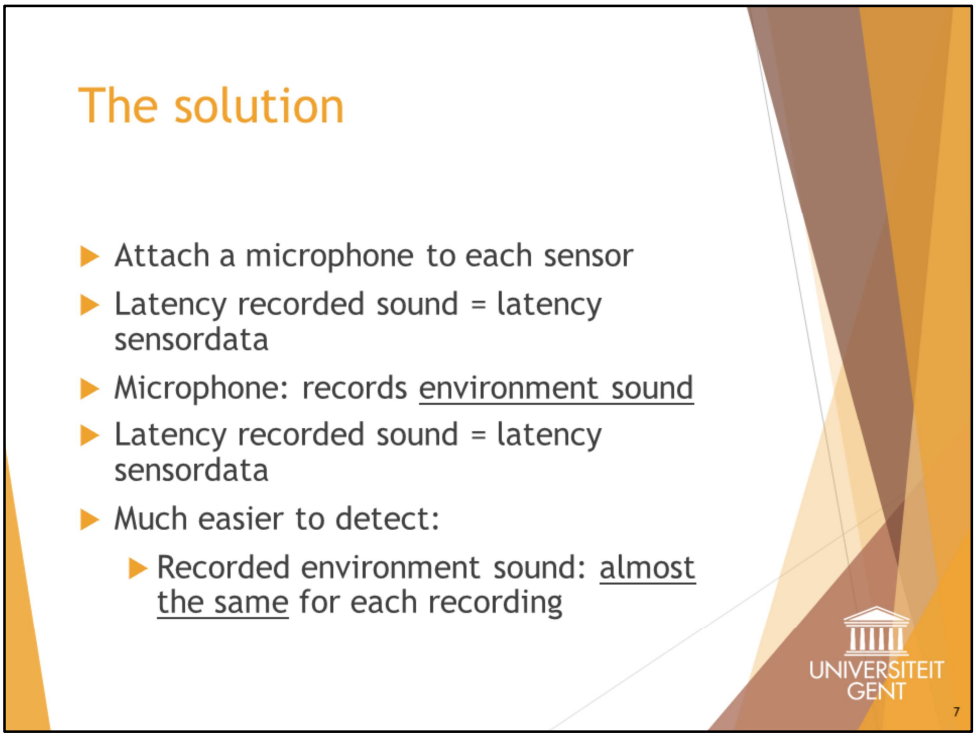
The wavefiles will be used as input for the Max/MSP module I will show during the demo. The module is responsible for the synchronization and will send the synchronized signals as output. The module doesn't have access to extra timing information.
The first synchronized signal will be sent the left speaker. The second synchronized signal to the right speaker.



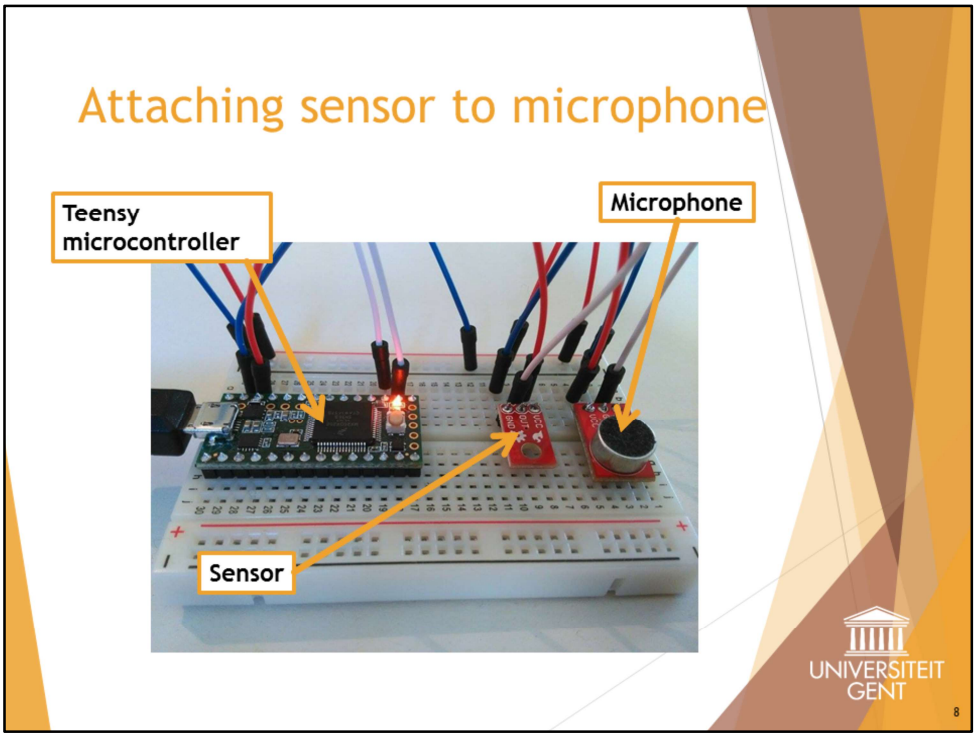
Quite impressive isn't it? I just want to note that the used algorithms don't just look for silence. Any latency change caused by any reason can be detected. It's also important to know that the audiostreams don't have to be exactly the same. The system works with recorded streams from different kinds of microphones.



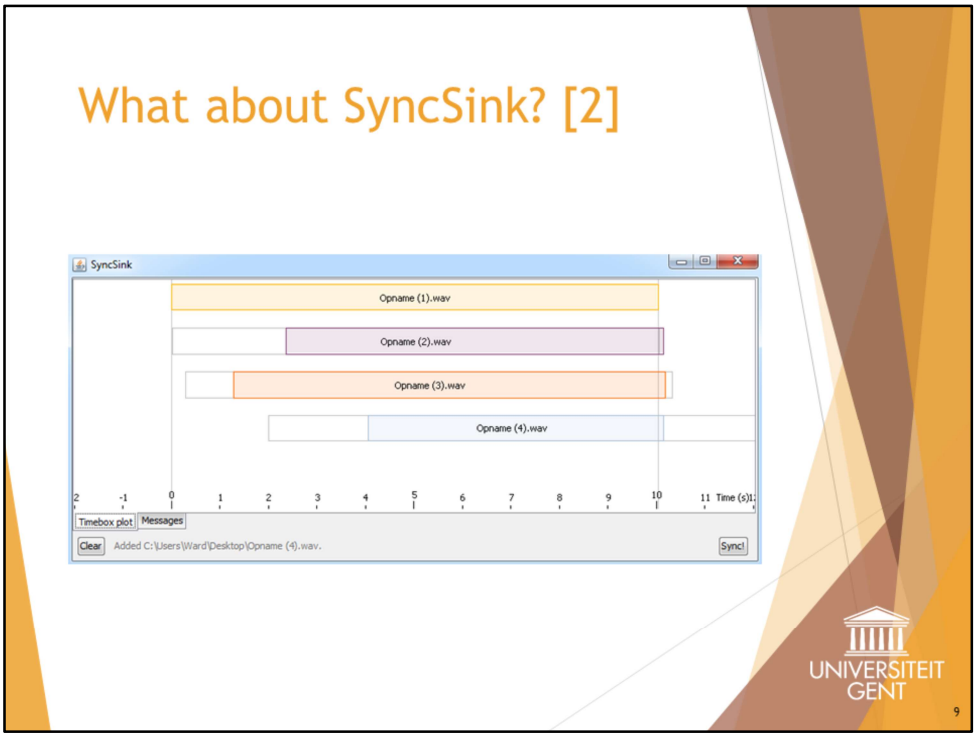
You are now probably wondering why this is useful. So, at IPEM a lot of experiments are done by using various sensors like accelerometers and pressure sensors. The problem is each sensor has its own unpredictable latency. Before analyzation is possible the data has to be synchronized.



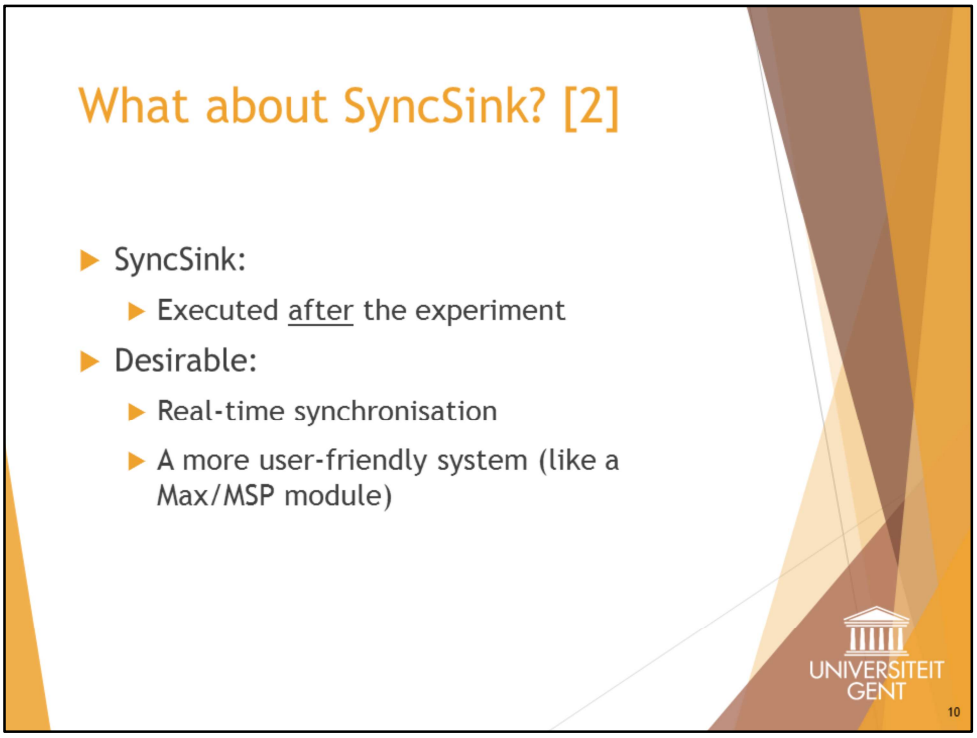
This can be done by attaching a microphone to each sensor so the latency of the sensor and microphone are the same. The difficulty of the initial problem can be reduced by recording the environment sound. This because the environment sound is the same for each microphone.



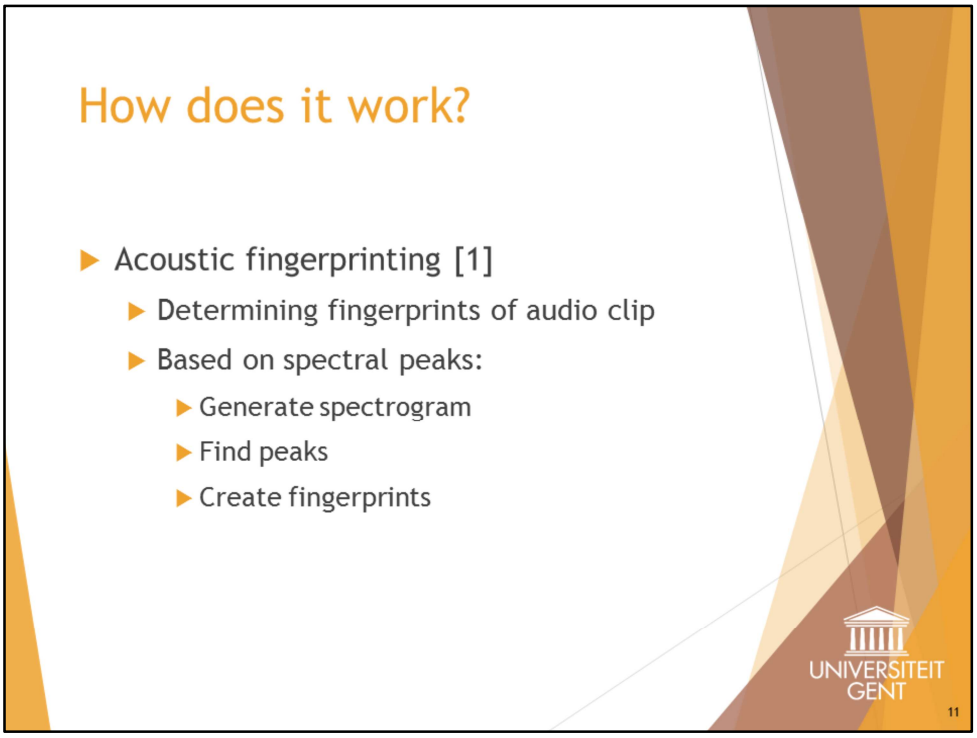
This is an example of a sensor and microphone set-up. By attaching them to the same microcontroller the latency between them is negligble.
(neglidgable)



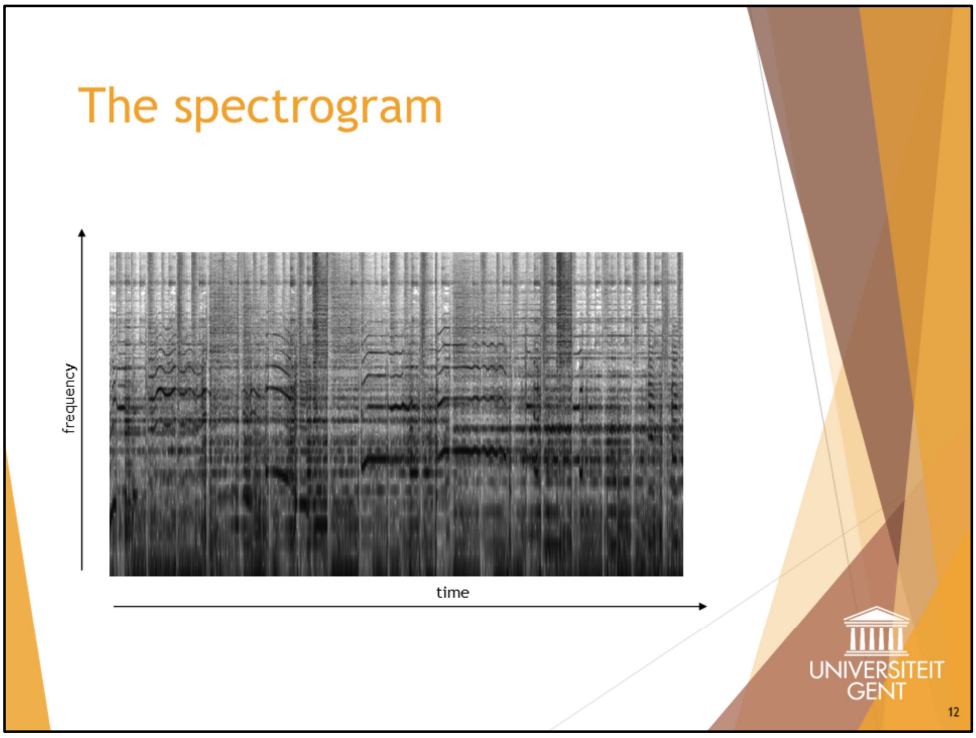
Some people probably already used SyncSink. It's a system using the same fundamentals I just explained. But there are some remarkable differences with what I've developed.



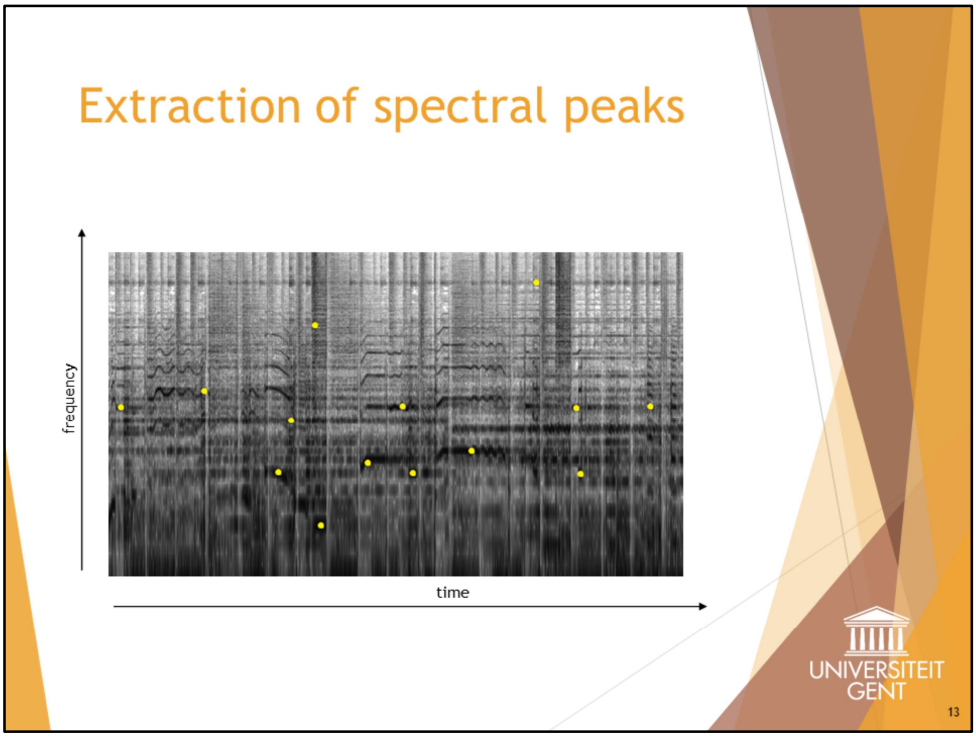
SyncSink has to be manually executed after the experiment. This is not very user-friendly and can be time-consuming.
For my master's thesis Joren asked me to develop a system which can do this in real-time. After developing the system I wrapped the system in a Max/MSP module.



The latency between the audiostreams can be detected by using the acoustic fingerprinting algorithm.
In order to do this the fingerprints of each audio fragment have be determined.
This is done by connecting the spectral peaks of the spectrograms of the audio fragments.
An example will make this terrible explanation more clear.

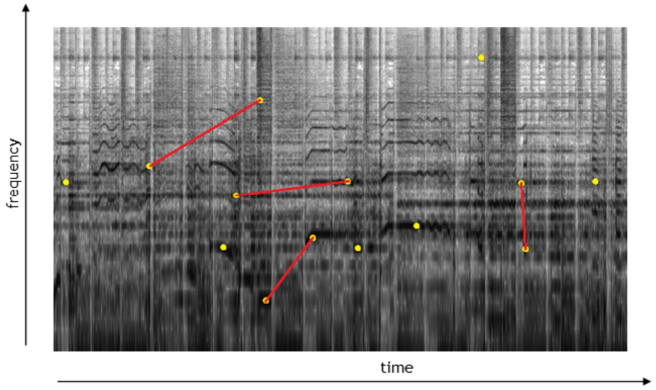


So, this is an example of a spectrogram. A spectrogram is a graph where the energy of each frequency-range is plotted against the time. On this slide the dark places have more energy than the less dark places.
The spectral peaks are places where the energy is higher than the energy of the neighbours. On the next slide I marked some spectral peaks.



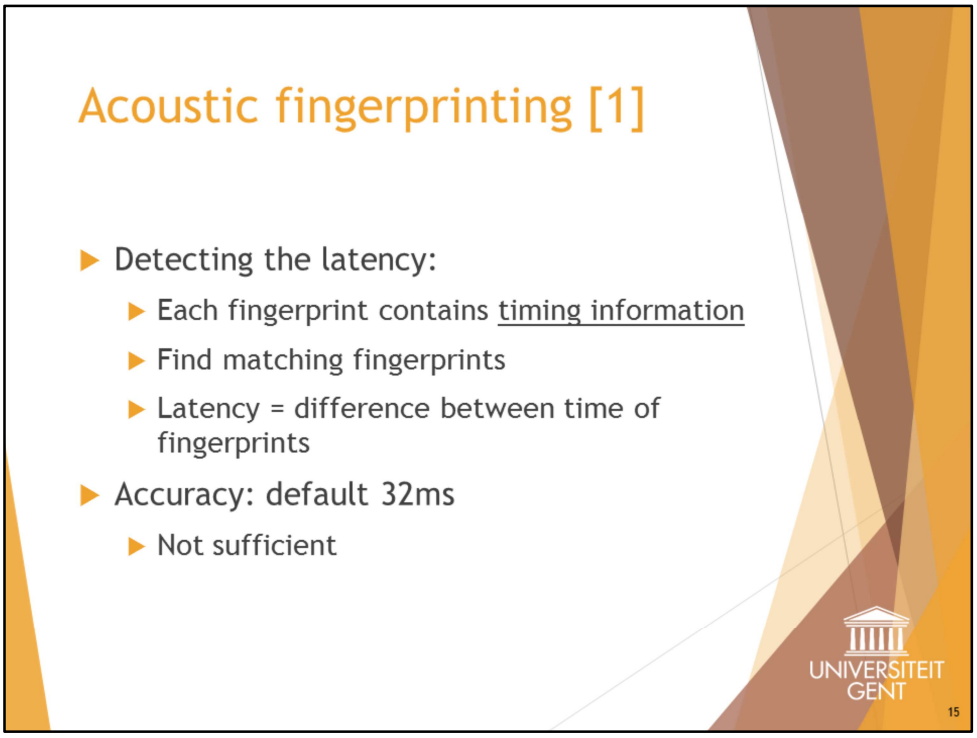
Here you can see some spectral peaks. The fingerprints are nothing more than a connection between two spectral peaks. Which spectral peaks are connected depends of the parameters of the algorithm. On the next slide I have marked some possible fingerprints.

Creation of the fingerprints



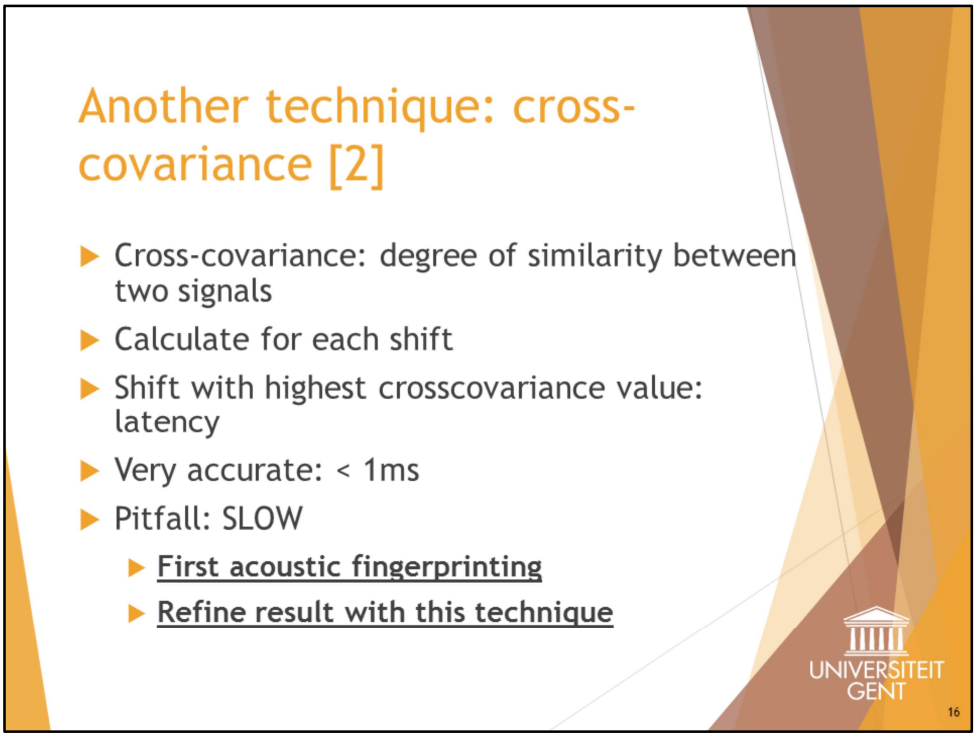
UNIVERSITEIT GENT

14

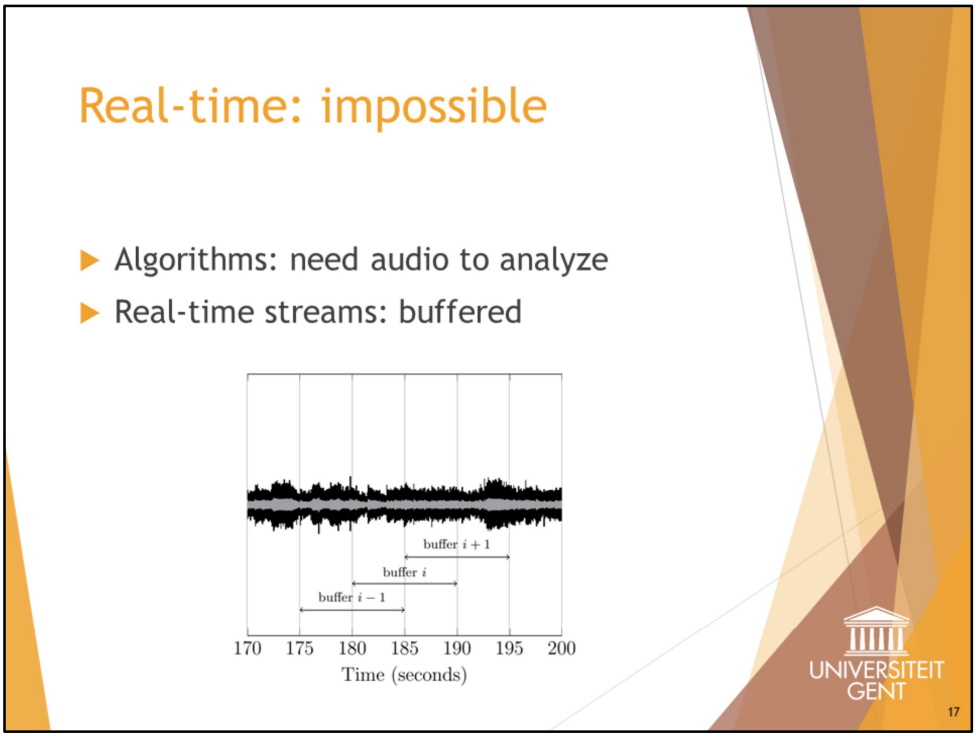


I forgot to say that each fingerprint contains information about the time where the spectral peaks occured. When many fingerprints can be matched this means that the audio streams contain resembling elements. The latency between the audio fragments can be calculated from the timing information of the fingerprints.

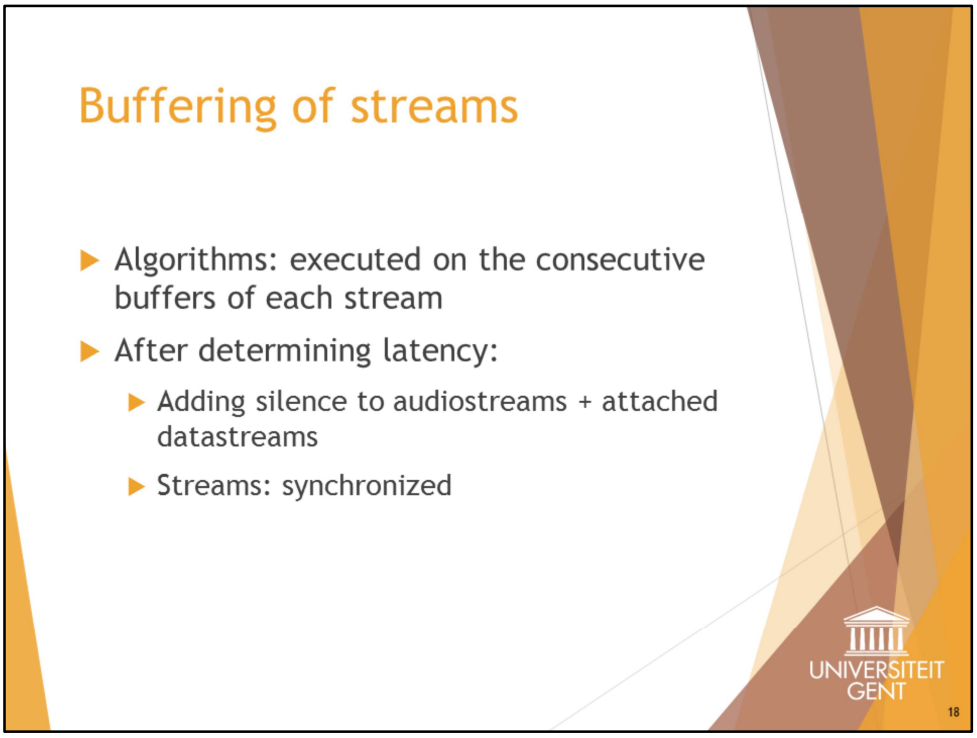
The biggest problem of this system is its accuracy. Fortunately there is another algorithm which can solve this problem.



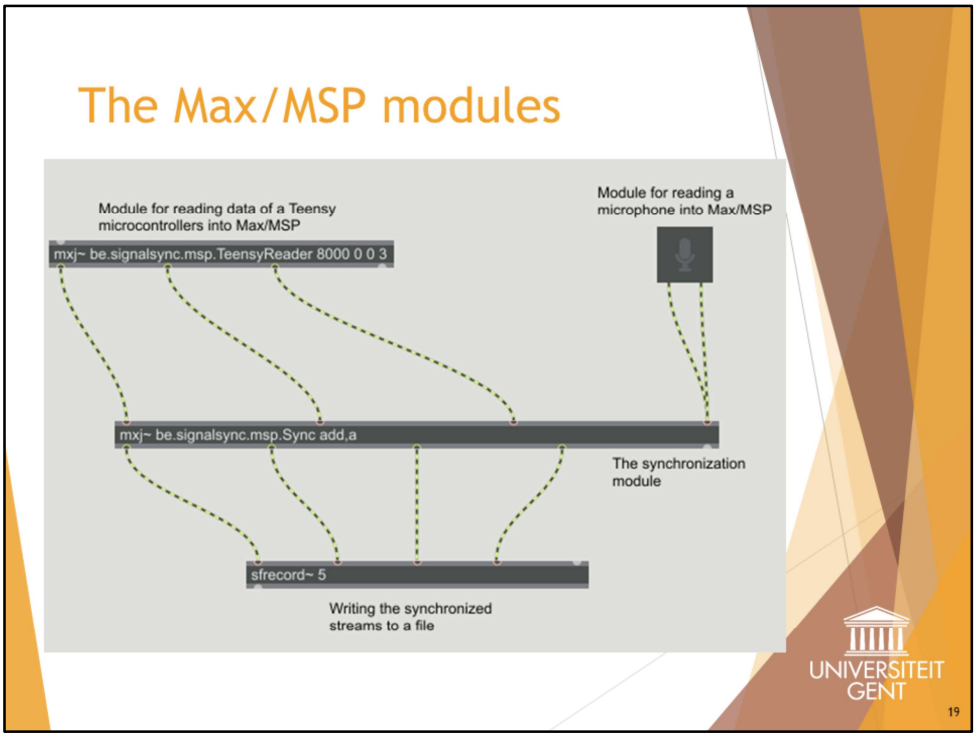
The cross-covariance is a calculation which measures the degree of similarity between two signals. By calculating this number for each possible shift between the signals it's easy to find out the latency. This algorithm is very accurate but also very slow. To solve we can first use the accoustic fingerprinting algorithm. Then the result can be refined by calculating the crosscovariance between the signals.



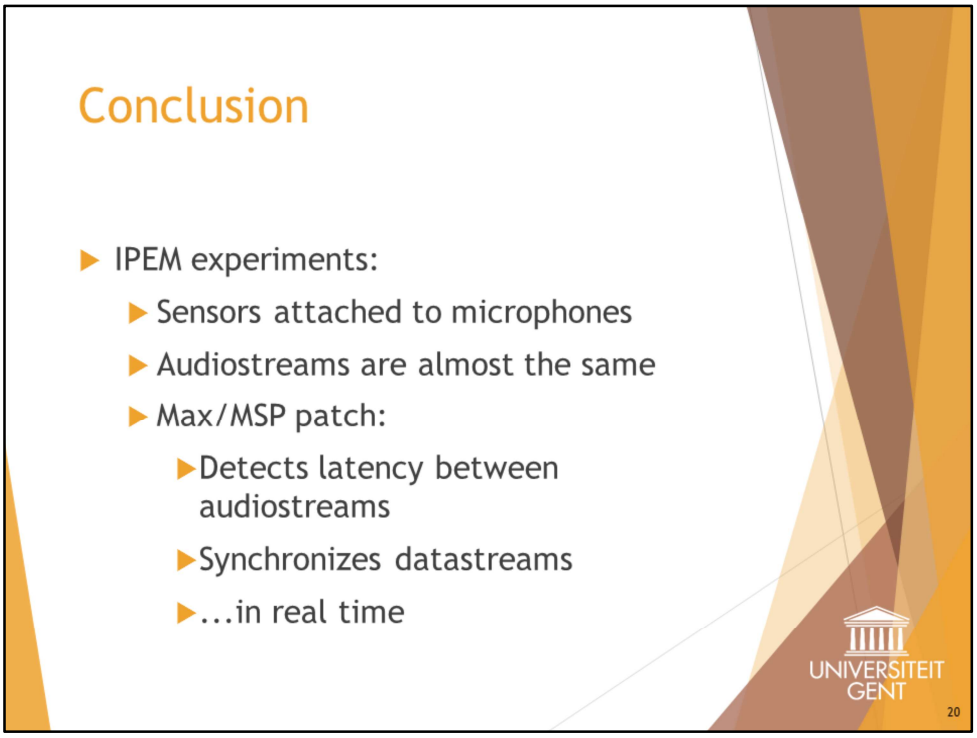
It’s practically impossible to synchronize the streams in real-time. The latency-detecting algorithms need some audio to analyse. That’s why the streams are buffered.



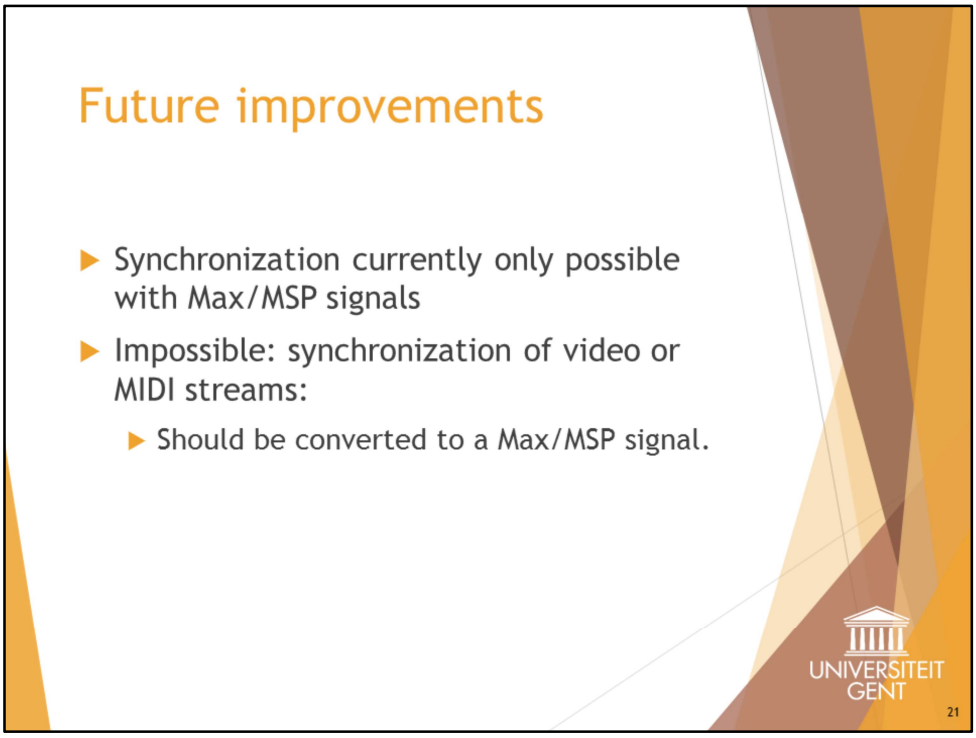
The algorithms are executed on the consecutive buffers. The resulting latencies are converted to the amount of silence which has to be added to each stream. After adding the silence to the streams, the streams are synchronized.



So this is an example of a Max/MSP patch which can be used during an experiment. The TeensyReader patch above reads 1 audiostream and 2 datastreams from a Teensy. The samples are sent to the Sync module. The data will be synchronized with the sound recorded with a microphone. The synchronized streams are written to a file by using the sfrecord module.



So never forget when you are doing an experiment to attach a microphone to your sensors. Because of the recorded environment audio is the same the latency can be detect easily. The latency is used to add whitespace to the attached streams resulting synchronized streams.
All this is possible in real-time.



The current synchronization system only supports Max/MSP signals. Everything can be synchronized but before it has to be converted to a Max/MSP signal. An possible improvement of the current system could be providing support for MIDI and video streams.

Questions?

UNIVERSITEIT GENT

22

References

- [1] Avery L. Wang. An Industrial-Strength Audio Search Algorithm.
  In Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR 2003), pages 7-13, 2003.
- [2] Joren Six and Marc Leman. Synchronizing Multimodal Recordings Using Audio-To-Audio Alignment.
  Journal of Multimodal User Interfaces
- [3] Joren Six and Marc Leman. Panako - A Scalable Acoustic Fingerprinting System Handling Time-Scale and Pitch Modification.
  In Proceedings of the 15th ISMIR Conference (ISMIR 2014), 2014.