Ping-pong: Musically Discovering Locations

Hyung Suk Kim[*]
CCRMA
Stanford University
660 Lomita Dr.
Stanford, California 94305
hskim08@ccrma.stanford.edu

Jorge Herrera[*]
CCRMA
Stanford University
660 Lomita Dr.
Stanford, California 94305
jorgeh@ccrma.stanford.edu

Ge Wang
CCRMA
Stanford University
660 Lomita Dr.
Stanford, California 94305
ge@ccrma.stanford.edu

ABSTRACT

A recently developed system that uses pitched sounds to discover relative 3D positions of a group of devices located in a shared physical space is described. The measurements are coordinated over an IP network in a decentralized manner, while the actual measurements are carried out measuring the time-of-flight of the notes played by different devices.

Approaches to sonify the discovery process are discussed. A specific instantiation of the system is described in detail. The melody is specified in the form of a score, available to every device in the network. The system performs the melody by playing different notes consecutively on different devices, keeping a consistent timing, while carrying out the inter-device measurements necessary to discover the geometrical configuration of the devices in the physical space.

1. INTRODUCTION

Ping-pong is a system that uses coordinated pitched sounds to discover the relative 3D positions of a network of devices located in a shared physical space. The initial motivation behind this system is to allow computer music ensembles [6, 9] to easily and musically discover the location of the devices in the group, with sub-metric precision. By doing this, 3D aware interactions are made possible. Pieces like Dahl's *SoundBounce*[3] and *Sparks* by McCurry, Harriman and Sanganeria[1] have used locations of the musicians/devices on stage as a parameter relevant to the performance. In both cases, a "manual" procedure is required for setup, either by forcing the performers to locate themselves at pre-defined positions or by calibrating the system using other sensors such as compasses. *Ping-pong* was designed to avoid such tedious procedures. Furthermore, it was designed to be a musical piece in itself: as the music unfolds, the locations of the devices on stage are discovered, using coordinated sounds in a call-and-response manner.

Work previously published by Herrera and Kim [5] dealt mainly with the problem of distance measurement between a pair of devices using pitched sounds, but left aesthetic and musical considerations unattended. This paper addresses these issues, and is structured as follows: section 2 briefly describes the system; section 3 describes how the system can be used musically; a specific implementation of the system is detailed in section 4; finally, in section 5 findings and insights are summarized and possible uses of the position information are suggested.

1.1 Related Work

A sonic location system was presented by Harter, Hopper, Steggles, Ward and Webster [4], but their system tracks a specially developed device using ultrasound and special hardware installed in the space. In *Ping-pong*, on the other hand, only off-the-self devices—smartphones or laptops—are required, and we purposely use sound to carry out the measurement.

Priyantha, Chakraborty and Balakrishnan [7] presented a system that uses dedicated hardware and works on ultrasonic and RF ranges. Despite using a different method, it does share a similarity with *Ping-pong*, as they both work in a completely decentralized way.

More recently, and using specific off-the-self smartphones, Qiu, Chu, Meng and Moscibroda [8] proposed a system to estimate 3D relative positions. While similar to *Ping-pong*, it differs in that it works for a single pair of devices only, and also in that it uses specific features of certain devices (e.g. multiple microphones, measured speaker frequency responses, etc.).

To our knowledge, *Ping-pong* is the first location system that carries out the measurement in a musical way.

2. SYSTEM DESCRIPTION

As previously mentioned, *Ping-pong*'s goal is to discover the 3D positions of devices located in the same physical space. To achieve this, first all pairwise distances are estimated. Once all pairwise distances have been measured, the relative 3D positions of all devices can be found by minimizing an error norm equation. The following subsections briefly describe each step.

2.1 Measuring pairwise distances

Sound can be used to infer the distance between a speaker and a microphone by measuring the time it takes a signal—emitted by the speaker—to reach the microphone. This time is then multiplied by the speed of sound in the propagation medium to estimate the distance. This task is trivial when there is a central clock that can accurately measure this delay, for example, by using correlation methods.

However, in the case where microphone and speaker are controlled by completely separate clocks that can't be accurately synchronized, the problem is much harder. The listening device can't know the precise moment at which the sound was emitted by the other device. Moreover, each device will most-likely have different internal latencies, which also affect the time-delay measurement.

To overcome these problems, *Ping-pong* takes a differ-

[*] Both authors contributed equally

[1] http://www.youtube.com/watch?v=q9srwrBUQ88

NIME'14, June 30 – July 03, 2014, Goldsmiths, University of London, UK.

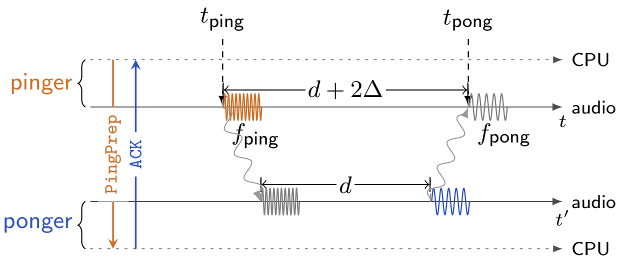

Figure 1: Pairwise distance measurement timeline.

ent approach: the measurement initiator (the *pinger*) emits a pitched sound (*ping*) and the other device (the *ponger*) will reply with a potentially different sound (*pong*). Before carrying out the measurement they must first agree on which pitches will be used and, more importantly, an additional delay d that the *ponger* will wait before replying. If the *ponger* does a good job at ponging exactly d after the *ping*'s onset time (compensating for his own internal latencies), then the *pinger* only needs to measure the time between the *ping* and the onset of the received *pong*. Figure 1 shows the time-line of events that take place when estimating the distance between two devices. By subtracting d and its own internal latencies, the time-delay due to the sound propagation can be computed and the distance between them ($\hat{r}$) can be estimated as $\hat{r} = c\Delta/2$, where c is the speed of sound.

It should be noted that 1 sample at a sampling rate of 22,050 Hz corresponds to approximately 1.5 cm. In other words, an error of 10 samples when estimating the onset (equivalent to 0.45 ms) translates to an error of roughly about 15 cm. Therefore, a very accurate onset estimation algorithm is required.

A simple method to achieve this is described in [5]. Succinctly, an onset detector and a pitched-signal detector run in parallel on the input audio. The onset detection finds possible onsets using a short-over-long term energy ratio. This stage can generate a high rate of false positive candidates. Simultaneously, the pitched-signal detection uses comb filters—tuned to the expected pitch—to detect the *ping* or *pong* signals. By combining the slow acting pitched-signal detection with the fast onset detection, accurate note onsets can be estimated. More details can be found in the cited reference.

2.2 Estimating 3D positions

To estimate the 3D positions, all pairwise distances need to be measured. The IP network is used to coordinate the measurements, schedule notes, and broadcast measured distances to all other devices.

Figure 2 shows, from a high level, the sequence of events that take place to measure distances from all other devices to a single device. The same procedure needs to be repeated for all other devices in the network.

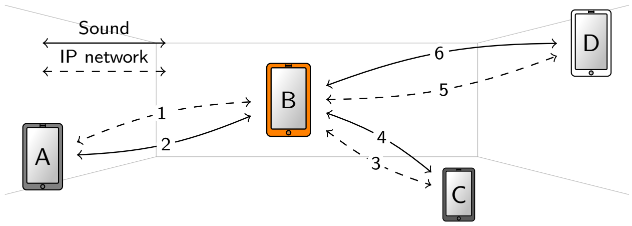

Figure 2: Event sequence to estimate the distances from B to other devices.

Once all pairwise distances have been measured and broadcasted to the other devices in the network, then any device can estimate the locations of the other devices relative to itself. The solution proposed by Wilson, Walter and Abel [10] was used. Given a vector of squared distances from an arbitrary origin $\boldsymbol{r_s}$ (which, without loss of generality could be the location of the devices performing the computations), and a matrix of noisy inter-device squared distances $\boldsymbol{R_s}$

$$\boldsymbol{r_s} = \begin{bmatrix} r_1^2 \\ \vdots \\ r_N^2 \end{bmatrix} \qquad \boldsymbol{R_s} = \begin{bmatrix} r_{1,1}^2 & \cdots & r_{1,N}^2 \\ \vdots & \ddots & \vdots \\ r_{N,1}^2 & \cdots & r_{N,N}^2 \end{bmatrix}$$

where r_j is the distance from the origin to device j and $r_{j,k}$ correspond to the distance between devices j and k. The matrix of positions $\boldsymbol{X}$ can be estimated by minimizing the estimation error $\boldsymbol{\epsilon}$ in

$$2\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{1}\boldsymbol{r_s}^T + \boldsymbol{r_s}\boldsymbol{1}^T - \boldsymbol{R_s} + \boldsymbol{\epsilon}. \tag{1}$$

The interested reader should refer to [10] for details.

It should be noted that at least 3 devices are required to carry out this calculation, which would yield a 2D solution. For a 3D solution, four or more devices are necessary.

3. SONIFICATION OF DISCOVERY

Possible answers to the question of how such a system can be used musically are covered in this section.

The main parameter required by the system to discover the positions is the delay d, which must be agreed between two devices carrying out a distance measurement. Musically speaking, d corresponds to the inter-onset-interval (IOI) between two notes. This delay d is essential to the distance measurement.

Other parameters are also relevant for the measurement, but their effect on the measurement depends on the implementation. As mentioned in section 2 the onset time estimation is extremely important for accuracy. In the system described in [5], sharp attacks are required to accurately estimate the distance while other parameters—such as note envelope and timbre—have little effect on the estimate. It is possible to have a different implementation that uses agreed waveforms (timbre plus envelope) and cross-correlation to estimate onset times, in which case sharp note attacks are not necessarily required.

Other musical parameters that don't affect the discovery process can be controlled or mapped following artistic and aesthetic decisions. For example, scale, note durations (to a certain extent) or other adornments such as pitch-bending can be controlled without affecting the distance measurement estimation.

There are several different strategies to control and make use of these musical parameters. A simple system can define a fixed score, known by all devices in the network, and use the IOIs in the score as the agreed d. In the next section such system is discussed in depth. A more interesting idea is a "form following function" approach, where the score is dynamically generated by mapping parameters of the discovery process to the music itself, either algorithmically or humanly driven. The dynamically generated music can facilitate—or artistically obstruct—the discovery process. For example, a composer or conductor could deliberately control the articulation of the played notes—by using legato notes—to distort distance measurements and generate "geometric tension". This could later be resolved by using staccato notes.

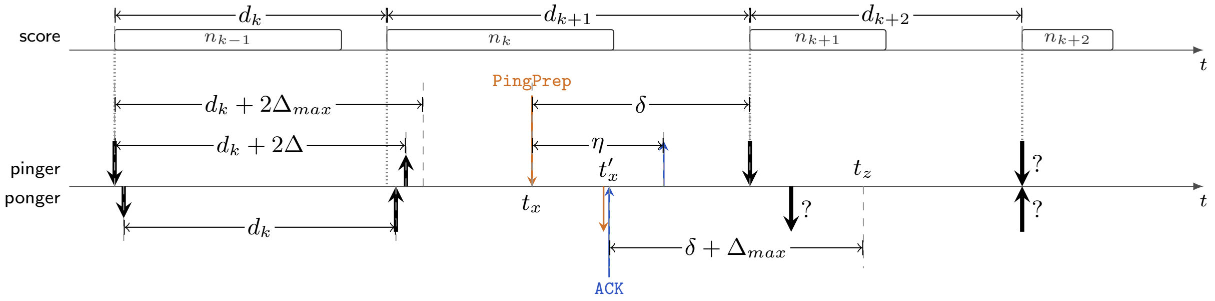

Figure 3: Timeline of events. The time axis t is an arbitrary absolute time. Vertical arrows show when different events happen on the *pinger* (above the axis) and *ponger* (below). Arrows pointing into the time axis are sent and arrows pointing out from it, received. Thick arrows correspond to notes; thin colored arrows represent network messages. Delays due to network and physical distance are depicted simultaneously.

4. PERFORMING A SCORE

An implementation of the simple system mentioned earlier is described here. The system uses a predefined musical score, which is made available to all the devices in the network. Much like in MIDI format, the score is defined as a sequence of indexed notes (n_k for the k-th note), each of which specifies the following data: unique auto-incremental index, onset time (in our case, as a time increment d_k), fundamental frequency, duration, intensity and timbre.

The scheme to play the melody is simple: at any point in time there is a *pinger*, which is the temporary conductor in control of the melody. This device determines which notes it plays and which ones are to be played by other devices in the network. All pairwise measurements are needed, so a "baton-passing" scheme was decided: a device pings all the rest and once all measurements have been carried out, a "baton" is passed to the last *ponger* in the form of a message over the IP network. The new *pinger* repeats the same process. If all devices follow the same sequence of devices (circularly shifted, so that the first *ponger* is the previous *pinger*), then all pairwise distances have been estimated once the initial *pinger* receives the baton back.

Every device must store the index k of the last note it played. This is important to make sure that the baton passing scheme works without noticeably disrupting the timing of the melody. When a pinger receives the baton, it also receives the index of the next note to play. This way, it is possible to keep the timing in a decentralized way, by scheduling the notes with respect to previously played notes.

While holding the baton, a device determines who plays what on a note-by-note basis. For the k-th note (n_k) on the score, the *pinger* decides whether to play it "unpaired" (that is, without expecting an associated *pong*) or with the intent of measuring the distance to another device. In the latter case, implicitly deciding that n_{k+1} will be played by the *ponger*. Prior to pinging, a "handshake" with the *ponger* is required. The handshake is carried over the network. It initiates with a `PingPrep` message identifying the *pinger* and the next note to play. Upon receiving this message, the *ponger* replies with an `ACK` message. The handshake can take random time η to complete, depending on network traffic and other network related delays. To prevent this variable latency from disrupting the timing of the melody, a few decisions have to be made at key times (see subsection 4.2).

Figure 3 shows the different events involved in a *ping-pong* sequence. One interesting point to note is that although there will be a small delay on playing the *pong*—due to the distance between *pinger* and *ponger*—this delay doesn't accumulate, as the next *ping* is scheduled in relation to the previous *ping*. This way, it is possible to keep playing the melody on time.

It is possible for a measurement to fail (see subsection 4.2). In this case, the *pinger* needs to repeat the measurement before proceeding onto the next *ponger*.

Finally, once all pairwise distances have been estimated, there may be notes left to be played in the score. These notes are used to repeat measurements. Measurements to be repeated are chosen by selecting pairs of devices (i, j) with low measurement confidence. A simple metric of the confidence $\mathcal{C}_{ij} = 1/|\hat{r}_{ij} - \hat{r}_{ji}|$, where $\hat{r}_{ij}$ is the estimated distance from device i to device j.

4.1 Considerations

Other melodic parameters such as note pitch, duration, envelope and intensity are derived directly from the score. That said, it is important to point out some limitations of the system.

There is a minimum duration required to detect a *ping* or a *pong*, which in turns imposes a minimum note duration on the notes played to carry out a measurement. This also defines a minimum IOI required to carry out a distance measurement. Other factors to consider are filter response time, maximum distance (see below) and network and audio latency.

The intensity parameter is difficult to control, as it is highly dependent on the device. On one hand, devices may have different characteristics (power output, frequency range, etc.), so a note played at the same "amplitude" in two different devices will most likely have different intensity, and be be perceptually very different. Furthermore, louder sounds mean better SNR, which improves the accuracy of the distance estimation. In fact, below some SNR threshold, the measurement is impossible.

Related to the previous point, beyond some distance, it is hard to estimate r_{ij} due to poor SNR. For this reason, a maximum distance $r_{max} = c\Delta_{max}$ must be specified. As will be discussed in the following section, Δ_{max} helps to recover from failures.

Regarding note envelopes, the measurement looks only at the onset of the note, so there's a lot of flexibility afterwards. That is, as long as the note has a sharp attack, the other parameters of the envelope can be manipulated freely.

Other problems may arise from the interaction of sound and space. Reverberation distorts the signal by adding images due to multiple pathways from source to listener. If there is a direct path, the proposed method for onset detection will detect the corresponding onset, as long there enough energy difference between the short and long term filters. In other words, this also imposes further constraints on the IOI between notes, to allow the previous notes to decay. This also implies that in the absence of a direct path, distance measurements will be noisy and overestimated, as longer paths will be detected. The pitched-signal detection using comb filters shouldn't be affected, as it is insensitive to phase differences.

4.2 Handling potential failures

Unfortunately, measurements do not always work as expected. Under some conditions (e.g. dropped packets), a measurement can fail. In this case, the measurement needs to be repeated. There are different types of failures that can occur, and each type requires different handling, in order to avoid disrupting the music. Table 1 lists the possible failures and how these are handled.

Device	Failure	Handling / Consequence
pinger	Fail to receive `ACK` on time (due to packet loss or because $\eta > \delta$)	*pinger* doesn't know if *ponger* actually received the `PingPrep` message. n_k won't be played, and a new measurement will be attempted using n_{k+1} and n_{k+2}
ponger	Fail to receive `PingPrep`	This implies that the *pinger* will fail to receive `ACK`, which was explained earlier
ponger	Fail to detect *ping* onset	Abort measurement and doesn't play n_{k+1}
pinger	Fail to detect *pong* onset	The measurement is repeated, using n_{k+2} (*ping*) and n_{k+3} (*pong*)
pinger	Erroneous measurement	The measurement is discarded and a new attempt is made

Table 1: Potential failures and corresponding handling when pinging n_k and ponging n_{k+1}.

The basic idea is to keep playing on time, despite possible mistakes. This is possible by leveraging known information about the score and who played what and when.

Instead of playing the right note at the wrong time, it is better to simply skip it. The `PingPrep` message contains the note index k. With this index, the *ponger* can read the necessary delay d_k from the score. In addition, the *pinger* also send δ: the time left to *ping*, from the moment the message was sent. The *ponger* can use this information to determine a deadline t_z, after which the *ping* has already passed (see Figure 3):

$$t_z = t'_x + \delta + \Delta_{max} \quad (2)$$

If possible, plans can be changed if a failure is detected. For example, if the *pinger* realizes the *ponger* won't be able to *pong*, it can decide to play the *pong* note itself.

4.3 Implementation

An iOS version of the system has been developed. It uses OSC protocol (via OSCpack[2]) which allows broadcasting, necessary to discover new devices in the network. Once devices discover each other, they open TCP sockets to handle device-to-device communication. All the DSP was implemented in C++ using the MoMu-STK [1], a port of the Synthesis Toolkit [2] for iOS.

For a detailed description and evaluation of the system, including measurement results and analysis, refer to [5].

[2] `http://www.rossbencina.com/code/oscpack`

5. CONCLUSION

A system capable of discovering the location of networked audio devices in a shared physical space has been described. Possible approaches to sonifying the discovery process have been covered and the musical limitation have been explained.

A specific implementation of such system was described in detail. The system uses coordinated sequencing of spatially distributed sounds to play a melody to carry out the measurements required to estimate the relative locations with sub-metric precision. Potential issues and failures have been identified and solutions proposed.

The idea of not just using sound to carry out the measurement, but also map the discovery process back into the sound itself has been introduced and discussed. In this way, the form of the musical piece could follow the function of the system, which is to discover performers' locations.

Finally, once the spatial position information is estimated, it is possible to use this information to perform spatially-aware musical performances. A simple example of using the information is to pan a sound or melody by emitting it from devices left to right, front to back, or around the audience, depending on the device positions. Another example is to use the position information to sonify the distance relations (e.g. sonify a spanning algorithm, a shortest path problem solution) by passing notes around from device to device.

6. REFERENCES

[1] N. J. Bryan, J. Herrera, J. Oh, and G. Wang. Momu: A mobile music toolkit. In *International Conference on New Interfaces for Musical Expression*, Sydney, Australia, 2010.

[2] P. R. Cook and G. P. Scavone. The synthesis toolkit (STK), 1999.

[3] L. Dahl and G. Wang. Sound Bounce : Physical Metaphors in Designing Mobile Music Performance. In *International Conference on New Interfaces for Musical Expression*, pages 178–181, Sydney, Australia, 2010.

[4] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. In *5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 59–68, New York, NY, USA, 1999.

[5] J. Herrera and H. S. Kim. Ping-pong: Using smartphones to measure distances and relative positions. *Proceedings of Meetings on Acoustics*, 20(1):–, 2014.

[6] J. Oh, J. Herrera, N. Bryan, and G. Wang. Evolving the mobile phone orchestra. In *International Conference on New Interfaces for Musical Expression*, Sydney, Australia, 2010.

[7] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *6th ACM MOBICOM*, Boston, MA, August 2000.

[8] J. Qiu, D. Chu, X. Meng, and T. Moscibroda. On the feasibility of real-time phone-to-phone 3d localization. In *9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 190–203, New York, NY, USA, 2011.

[9] G. Wang, N. Bryan, J. Oh, and R. Hamilton. Stanford laptop orchestra (SLOrk). In *International Computer Music Conference*, Montreal, August 2009.

[10] R. S. Wilson, J. H. Walters, and J. S. Abel. Speaker array calibration using inter-speaker range measurements. In *Audio Engineering Society Convention 116*, Berlin, Germany, May 2004.