Abstract

A novel system for realtime range and geometry estimation of a group of smartphones colocated in a shared physical space is presented. The system uses off-the-shelf devices and employs audible signals to estimate inter-device (pairwise) distances. Coordinated sound synthesis and processing of a pair of pitched sounds allows to estimate the distance between two devices based on the travel time. To overcome the absence of a centralized clock to coordinate measurements, a synchronous communication channel was used. When four or more devices are present, it is possible to estimate their relative positions in a three dimensional space, by minimizing an equation error norm. The system works both on closed and open spaces. We believe that such system opens the possibility for new ways of interaction that could benefit musical expression, social interaction and gaming.

Ping-Pong: Using Smartphones To Measure Distances And Relative Positions

Jorge Herrera
jorgeh@ccrma.stanford.edu

Hyung Suk Kim
hskim08@ccrma.stanford.edu

Center for Computer Research in Music and Acoustics
Stanford University

December 2, 2013

1 Introduction

There is no doubt that the idea of ubiquitous computing, introduced by Weiser in the early 90s [10], is now a reality. Networks of digital devices are pervasive, and are now part of everyday life. In music, laptops, tablets and smartphones have found their place in ensembles such as laptops and mobile phone orchestras [8, 5]. Such ensembles sometimes use the relative location of performers and/or devices as part of the musical interaction. Dahl's *SoundBounce* [2] is a great example of such situation.

There are different ways of providing the location information in the aforementioned scenarios: one possibility is to use on-board sensors such as gyroscope, compass or GPS, to estimate the location; another option is to manually specify the locations a-priori. While in some situations these mechanisms may work, they present some drawbacks. On-board sensors usually have low accuracy (e.g GPS) or incomplete information (gyroscope and compass). The manual solution can overcome these problems, but is not flexible, doesn't scale well, and can't be used in dynamic scenarios where the location of the devices can change during a performance.

More broadly, networks of devices capable of capturing and reproducing sound can be found in other situations such as gaming, or even casual social interactions in a physical space. We believe that the ability to easily estimate the relative positions of devices could open new possibilities of interaction.

We present a novel system that uses sound to estimate the location of an arbitrary number of co-located devices in a 3D space with sub-metric precision, using only off-the-shelf devices capable of sound-capture, sound-synthesis and network communication. Furthermore, the system uses pitched sounds, and allows for arbitrary temporal organization of them to estimate their relative distances and positions, enabling it as a musical system for position estimation. The proposed system uses network communication to coordinate the measurements, but the measurements themselves are carried out using sound waves propagating through the air.

1.1 Related Work

A good overview on the general problem of location is presented by Lamarca [4]. In the particular case of audio location, many solutions have been proposed. Some use highly specific hardware [3, 6], but this escapes the scope of our work, as we want to focus on a low-cost, off-the-shelf solution. Qiu et.al [7] have proposed a system with high precision using specific off-the-shelf devices. While similar to the method we propose, 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.) Furthermore, making the system musical is a goal that we have, but not a concern for them.

Wilson et.al. [11] have proposed a method to estimate relative positions from noisy pair-wise distance measurements. They propose a closed-form solution that minimizes an equation error norm. Our system implements their solution (see §2.2 for more details).

2 Proposed Method and System

From a high level point of view, our system can be decomposed into two main components: (a) Pair-wise distance estimation; (b) Relative positions (geometry) estimation.

Figure 1 shows a representation of the sequence of events that take place to measure pair-wise distances. Once all pair-wise distances have been estimated, each device is able to compute the relative geometry—*i.e.* 3D locations—of all the devices, generating a 3D map of their locations.

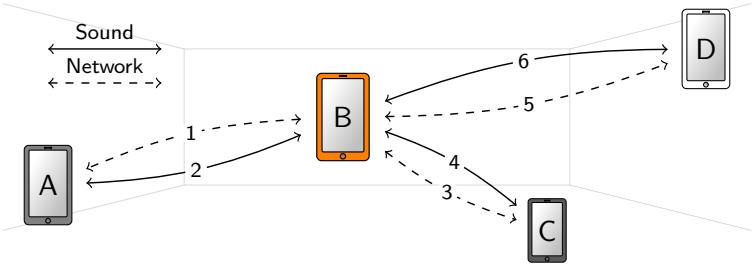

Figure 1: Event sequence to estimate the distances from B to 3 other devices. Every measurement is coordinated via network communication. Then sound is used to measure the round-trip time and estimate the distance. It must be noted that in order to estimate the relative positions, all pair-wise distances are required, so the same process must be repeated for devices A, C and D, and the estimates must be broadcasted to the rest.

2.1 Pair-wise Distance Estimation

In the traditional case—for example, speaker-array calibration—there is usually a central computer that controls the timing of the signals being played and recorded through the different speakers and microphones in the system. Since the central computer controls speakers and microphones, there is a single centralized clock, with respect to which all distance estimations can be computed, for example, by using autocorrelation techniques.

In our system, on the other hand, the scenario is quite different. Instead of a central device coordinating the signals being played and captured, each device acts in an independent fashion. When measuring pair-wise distances, the device starting the measurement has no knowledge of the internal delay of the responding device. To overcome this problem, we propose the following procedure.

Pair-wise distances are estimated by measuring the round trip time of a two note sequence. The first note, the *ping*, is played by the *pinger* (the device that initiates the measurement) at time t_{ping}. The second device, the *ponger*, detects the *ping* and replies with a *pong* note, some time d after the *ping* is detected. The delay d is defined by the *pinger* and communicated to the *ponger* prior to initiate the measurement. Once the *pinger* detects the *pong*, at time t_{pong}, the distance $\hat{r}$ is calculated by the *pinger* as follows:

$$\Delta = t_{pong} - t_{ping} \tag{1}$$

$$\hat{r} = (\Delta - d)c/2 \tag{2}$$

where c is the speed of sound, in m/s. Figure 2 shows an idealized time-line of the events that take place in the distance measurement just described.

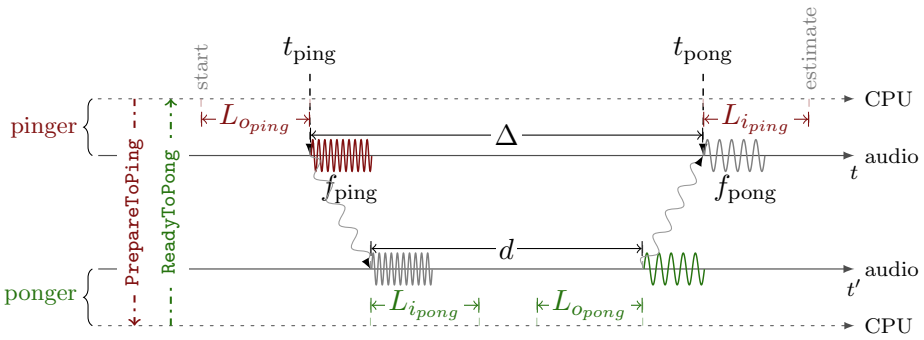

Figure 2: Timeline of the pair-wise distance measurement.

For the *ponger* to respond with a delay of d precisely, it needs to compensate for the audio latency $L_{pong} = L_{i_{pong}} + L_{o_{pong}}$. Similarly, the pinger needs to account for its internal latency $L_{ping} = L_{i_{ping}} + L_{o_{ping}}$ to measure Δ.

Internal latency is often reported by the device, but it can also be measured. A standard way to measure a device's latency is to play a signal, which is then cross-correlated with the input signal. The maximum cross-correlation lag corresponds to the overall system latency. Another option is to estimate the audio latency by listening to itself, using some onset detection such as the one explained in §2.1.1. In our experiments all three methods yield similar results, so we use the latency reported by the operating system.

2.1.1 Signal Processing

The digital signal processing in the system has a dual purpose: first, it must allow a device to detect when the expected note (*ping* or *pong*) is being played at its tuned frequency f_0 (signal detection); secondly, it must accurately determine the onset time of the note (onset estimation).

Signal Detection To detected the presence of the expected signal at frequency f_0, we employ two comb filters in parallel (see Figure 3). One is configured as a "peaking" (IIR) comb filter and the other as a "notching" (FIR) comb filter, both tuned to f_0. The ratio of the energy of both filters is then computed, to provide a measure of the *confidence* that the expected signal is present. The ratio allows to discard wide band signals that could be picked by a single peaking filter.

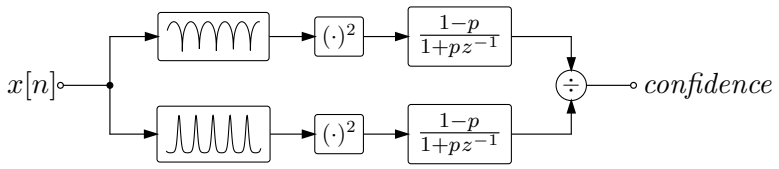

Figure 3: Diagram of the signal detection processing.

To be able to accurately tune the comb filters to any particular frequency, fractional delays are required. A simple first order all-pass fractional delay is not a good choice, as the first order approximates well at low frequencies, but high frequencies are poorly fitted to integer multiples of the fundamental, due to a non-constant group delay of such configuration. Instead, Thiran's method [9] is used to compute a higher order all-pass fractional delay, which ensures maximally flat group delay.

One could consider using the *confidence* signal to estimate the onset of the signal. While in principle this could work, in reality this doesn't work well for our requirements, as the pole p used in the energy envelope follower slows down the response, affecting the accuracy of the onset estimation. Therefore, a fast response processing is needed to estimate the onset of the signal.

Onset estimation To accurately and quickly estimate the onset of the input signal, we computed the short-term over long-term energy ratio of the unprocessed signal.

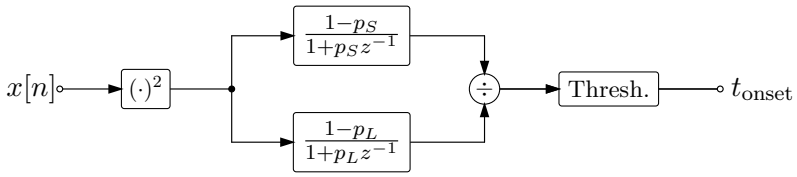

Figure 4: Diagram of the onset detection processing.

The short and long time constants ($\tau_S = 0.5$ms and $\tau_L = 10$ms) are chosen to provide a quick and noticeable spike the energy ratio signal. A simple thresholding mechanism is then used to determine the onset. It should be pointed out that we purposely tuned these values to yield many onsets (high false-positive rate but low false-negative rate). In combination with the signal detection described earlier, our system is capable of discriminate onsets corresponding to the expected signal.

2.1.2 Networking

All measurements are carried out using sound waves propagating through the air, but since there is no central device to coordinate the measurements, we decided to use networking capabilities of the devices to allow for coordination of the measurements. It should be noted that while we could have explored the Network Time Protocol (NTP)[1] to coordinate measurements, this would have required a centralized server, contrary to our idea of a completely decentralized system.

There are two types of communications going on in our system: peer-discovery; and pair-wise measurement coordination.

Peer-discovery After a new device joins the network, it will continuously broadcast a `HelloWorld` message once every few seconds. We use this message pulse to detect when devices join or leave the network. When a `HelloWorld` message is received by a second device (the receiver), it checks if the broadcaster is already in the list of known peers, and adds it to the list if not. Also a device will check the last time the broadcast message was received for each registered device and remove the device if it has not been active for a sufficient duration.

Pair-wise Measurement Coordination When a pair-wise distance measurement is triggered, there needs to be coordination between the pair of devices. The measurement initiator—the *pinger*—first sends a `PrepareToPing` message with the frequency to tune the filters to (f_{ping}), the frequency to reply with (f_{pong}) and the *pong* delay d mentioned in Equation 2. The *ponger*, in turn, acknowledges by changing its state to `ReadyToPong`. Upon receiving the acknowledge message, the *pinger* is in condition to start the measurement as described in Figure 2. Finally, the distance estimate is broadcasted to the other devices in the network, so all of them can independently estimate the geometry.

Given that TCP/IP does not allow broadcasting, we decided to use OSC over UDP for communications. The drawback of this alternative is that the UDP protocol is not reliable, as packages are not guaranteed to reach their destination. While in some audio applications this is acceptable, in our case we had to implement a Retry-Until-Acknowledge protocol on top of OSC to make sure messages would reach the destination.

2.2 Relative Position Estimation

This section outlines the procedure used to convert pair-wise distances into relative positions of the devices. As mentioned before, we implemented the solution proposed by Wilson et.al. [11].

Briefly, given a vector of squared distances from an arbitrary origin $\boldsymbol{r_s}$, and a matrix of noisy inter-device squared distances $\boldsymbol{R_s}$

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

[1] `http://www.ntp.org/`

where r_j is the distance form 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 error in

$$2\boldsymbol{X}\boldsymbol{X}^T = \mathbf{1}\boldsymbol{r_s}^T + \boldsymbol{r_s}\mathbf{1}^T - \boldsymbol{R_s} + \boldsymbol{\epsilon} \tag{3}$$

where $\boldsymbol{\epsilon}$ is the estimation error.

It should be noted that the solution to this equation is rotation invariant, that is, $\boldsymbol{X'} = \boldsymbol{Q}\boldsymbol{X}$ for some rotation matrix $\boldsymbol{Q}$ is also a valid solution.

Finally, noting that $\boldsymbol{X}\boldsymbol{X}^T$ is symmetric, its a singular value decomposition is $\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$. For a three-dimensional approximation, we can zero all but the 3 largest singular values. Let $\boldsymbol{U_3}$ and $\boldsymbol{D_3}$ be the resulting matrices. Then, the estimated positions are given by

$$\hat{\boldsymbol{X}} = \boldsymbol{U_3}\boldsymbol{D_3}^{\frac{1}{2}}\boldsymbol{Q} \tag{4}$$

(see [11] for a detailed derivation).

Given that it is a closed-form solution, involving simple matrix operations, a real-time implementation is perfectly feasible, thus enabling the whole system to work in real-time.

2.3 Implementation

We developed an iOS implementation of the system described above, using the MoMu toolkit [1]. The results presented in §4 were obtained using this app.

3 Musical Considerations

As mentioned earlier, there are parameters in the system that allow users to employ this system musically. It is possible to control and manipulate pitches and timbres used in consecutive measurements. Similarly, it is possible to control the duration of the *ping* and *pong* notes as well as the pong delay d. Therefore, the whole procedure can be carried out by playing melodies, where each note is played by different devices in the network.

Another interesting idea to explore has to do with repetition. Since the distance estimates are noisy, repeated measurements can be used to improve the estimation's accuracy. This idea would allow for larger musical works with motifs that repeat, while improving position estimates.

4 Results

To test the distance estimation system, four devices were placed at the vertices of an imaginary irregular tetrahedron. Distances between pairs of devices were first measured and then estimated using the system described above. Five estimates were made in each direction, giving a total of ten distance estimates per edge. For each pair A and B, five estimations were taken using A as the *pinger* and other five estimates were carried out using B as the *pinger*. The results are summarized in Figure 5. It should be noted that we used the device centers to measure the distances. We will discuss the implications in §4.1.

As can be seen in the Figure 5, the standard deviation ranges from 2 cm to 12 cm, comparable to the device size, which indicates that the system is very precise (estimates are very repeatable). Accuracy—measured as the absolute value of the estimate error—ranges

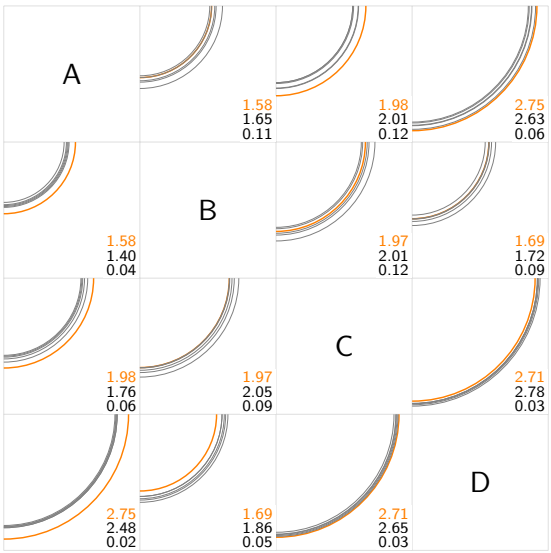

Figure 5: Visualizations of the distance measurements obtained in an experiment using 4 devices: one third generation iPad (A), two fourth generation iPod Touches (B and D) and an iPhone 4 (C). The matrix shows the measured and estimated pair-wise distances (orange and gray, respectively). Rows correspond to *pinger* and columns to *ponger*. For each *pinger-ponger* pair, 5 estimations were carried. The numbers reported are the measurement (top), average estimate (middle) and standard deviation of the 5 estimates (bottom), all in meters.

from 3cm to 27cm. Other tests conducted showed that these values do not depend on the distance between *pinger* and *ponger*. Furthermore, we observed that using larger devices (*e.g.* iPad) increase the error consistently. When measuring distances between smaller devices (*e.g.* iPods, iPhones), the average error decreases to a few centimeters, improving accuracy. If one of the devices is larger, then the error increases by roughly 15-20cm. If both devices are large, then the error increases by about 35cm. Further investigation is needed, but we assume that the larger distance between speaker and microphone in large devices negatively affects the estimates.

4.1 Sources of Error

There are several sources of error that impact the estimation differently. The most obvious one has to do with the fact that in our derivations we implicitly assume that the microphone and speaker in a device are placed in the same position, but in reality this is not the case. This is true particularly in laptops and tablets, were they can be placed up to 0.3 meters apart. As has been observed, this type of error is reduced by using smaller devices such as smart-phones, which naturally have speaker and microphone closer together.

Another source of error is the estimation of the note onset. These errors might have to do with poor SNR in noisy environments. A poor estimation of the time of arrival affects Δ or d in (1) and (2), which directly impacts the distance estimation $\hat{r}$. The absence of high frequency content in the *ping* and/or *pong* signals can also affect the note onset time estimation, as the change in energy is slower. It should be noted that for a sampling frequency of 22.05 kHz (used in the iOS app developed), and considering a speed of sound of 340.29 m/s, an error of 1 sample corresponds to roughly 1.5 centimeters of error.

Another potential source of error is occlusion. When an object is placed between *pinger* and *ponger*, there will be no direct path, so the algorithms will detect reflections, which will clearly overestimate the distance between devices.

Finally, from Equation 2 we can see that the distance estimation is proportional to the speed of sound c. Since c is a function of temperature, this is another minor source of error (c increases by roughly 4% from 5℃ to 25℃).

5 Conclusion

We introduced a system to estimate the spatial position of devices co-located in a shared physical space. The system uses pitched sounds to estimate inter-device distances. With pair-wise distances it is possible to estimate the positions in a 3D space. Network communication was used to overcome the absence of a central clock to coordinate and compute pair-wise distances.

In preliminary tests, the system has shown good precision and accuracy, but both can be negatively affected by different factors such as large distance between device's speaker and microphone or occlusion between devices.

References

[1] Nicholas J Bryan, Jorge Herrera, Jieun Oh, and Ge Wang. Momu: A mobile music toolkit. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), Sydney, Australia*, 2010.

[2] Luke Dahl and Ge Wang. Sound Bounce : Physical Metaphors in Designing Mobile Music Performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 178–181, 2010.

[3] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 59–68, New York, NY, USA, 1999. ACM.

[4] Anthony LaMarca and Eyal De Lara. *Location Systems: An Introduction to the Technology Behind Location Awareness*. Synthesis Lectures on Mobile and Pervasive Computing. Morgan & Claypool Publishers, 2008.

[5] Jieun Oh, Jorge Herrera, Nicholas Bryan, and Ge Wang. Evolving the mobile phone orchestra. In *International Conference on New Interfaces for Musical Expression*, Sydney, Australia, 06/2010 2010. .

[6] Nissanka Bodhi Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket Location-Support System. In *6th ACM MOBICOM*, Boston, MA, August 2000.

[7] Jian Qiu, David Chu, Xiangying Meng, and Thomas Moscibroda. On the feasibility of real-time phone-to-phone 3d localization. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 190–203, New York, NY, USA, 2011. ACM.

[8] Scott Smallwood, Dan Trueman, Perry Cook, and Ge Wang. Composing for laptop orchestra. *Computer Music Journal*, pages 32(1):9–25, 2008.

[9] Vesa Välimäki, Heidi-Maria Lehtonen, and Timo I Laakso. Musical signal analysis using fractional-delay inverse comb filters. In *Proceedings of the International Conference on Digital Audio Effects*, pages 261–268, 2007.

[10] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, July 1999.

[11] R. Scott Wilson, Jeffrey H. Walters, and Jonathan S. Abel. Speaker array calibration using inter-speaker range measurements. In *Audio Engineering Society Convention 116*, 5 2004.