Smartphones

Editor: Roy Want ■ Google ■ roywant@acm.org

SwordFight: Exploring Phone-to-Phone Motion Games

Zengbin Zhang, David Chu, Xiaomeng Chen, and Thomas Moscibroda

EDITOR'S INTRO

Motion games have become a popular development among console gamers; however, they haven't had much impact on the smartphone domain. This piece describes new opportunities for mobile peer-to-peer motion gaming that could transform this area. —*Roy Want*

The landscape of today's mobile games is rich and varied. However, most multiplayer games invariably require players to be physically passive, looking at and interacting with the screen to conduct game action. The success of the Nintendo Wii or Xbox Kinect in console gaming, on the other hand, has demonstrated a demand for much more interactive games.

Here, we report on a novel class of phone-to-phone mobile-motion games that aim to achieve a similar level of physical interactivity. In these games, the phone's position, location, orientation, or movement is an integral part of game play. However, in contrast to the Wii or Kinect, such games don't rely on external infrastructure such as a microphone array or camera—they're played purely phone to phone.

The key underlying technology in mobile-motion games is a new real-time phone-to-phone distance measurement substrate we developed—the Fast, Accurate, and Robust (FAR) localization system.[1] The system enables two potentially fast-moving phones to keep accurate distance estimates. To practically demonstrate the system's ability to enable novel gaming concepts, we developed SwordFight, a prototype mobile-motion game.

SWORDFIGHT

In the SwordFight game, two players wield their phones and try to attack each other. A player can attack the opponent by tapping the screen. If player A attacks and her phone is within 20 cm of the opponent's phone, she wins. However, attacking costs energy, and an attack can only be sustained for four seconds before the energy has completely depleted. Energy can be regained over time when the player remains in nonattack mode.

Although the game is intuitive, it requires the ability to conduct very fast, accurate, and robust distance measurements between the phones so that at any moment during play, both phones have precise distance information. Studies have shown that to sustain high-speed action games, a lag of more than 100 milliseconds will decrease user satisfaction.[2] So, we need to conduct phone-to-phone distance measurements at a rate of at least 10 Hz. The measurements also have to be accurate—with no more than a few centimeters of error—and robust in the face of mobility, noise, and networking issues. In the absence of any external infrastructure, the combination of these three requirements constitutes a significant technical barrier on commodity phones.

We developed the FAR system to systematically improve upon existing phone-to-phone ranging schemes by making them faster and more frequent as well as more robust in the face of mobility—all while maintaining the required degree of accuracy. It's well-known that acoustic sound can be used for distance measurements. Works have demonstrated that under ideal circumstances (such as those with no mobility) and with sufficient computation time, accurate ranging can be achieved even on commodity phones.[3,4] The problem is that in a motion game scenario, the circumstances are far from ideal, and the requirements are substantially more challenging.

The Challenges

First, existing ranging protocols assume that phones are static during the measurement. For a game like SwordFight, this isn't valid, because human hand speed can be up to 2 meters per second.[5] Furthermore, with two phones moving toward or away from each other, we need to consider aspects such as the Doppler effect.

Published by the IEEE CS ■ 1536-1268/12/$31.00 © 2012 IEEE

Second, acoustic ranging requires an expensive cross-correlation algorithm to detect the precise time-of-arrival of the sound signal. The cross-correlation algorithm is computationally intensive and can't run on phones at sufficient speed to enable a SwordFight game.

Third, both computation and communication (acoustic tone exchanges, protocol handshakes, and so on) incur a fundamental and significant delay.

Our Contributions

We systematically address these challenges with three main contributions. First, to enable real-time distance measurement, we replace the standard computationally intensive cross-correlation algorithm with an efficient multistage algorithm. Our algorithm employs autocorrelation to fundamentally reduce computational complexity while preserving accuracy by confining cross-correlation to a very narrow search window.

Second, we employ a new pipelined streaming execution strategy that overlaps protocol communication and algorithm computation. It turns out that both pipelining and streaming are critical in realizing real-time measurements.

Finally, we tackle the practical sources of measurement errors during motion gaming for increased robustness, including environmental issues such as ambient noise and multipath. In addition, we also design methods to address the effects of Doppler shift, which is caused by the rapid movement of phones during the game.

REAL-TIME DISTANCE MEASUREMENTS

To see how our techniques affect lag and frequency, it is useful to consider the execution strategy used in existing schemes,[2,3] in which the essential idea is to have two phones, A and B, play and record known audio tones one after another. Each phone records its own emitted tone and the tone originating from the remote one, and measure the distance by calculating the time difference of the tone's arrival.

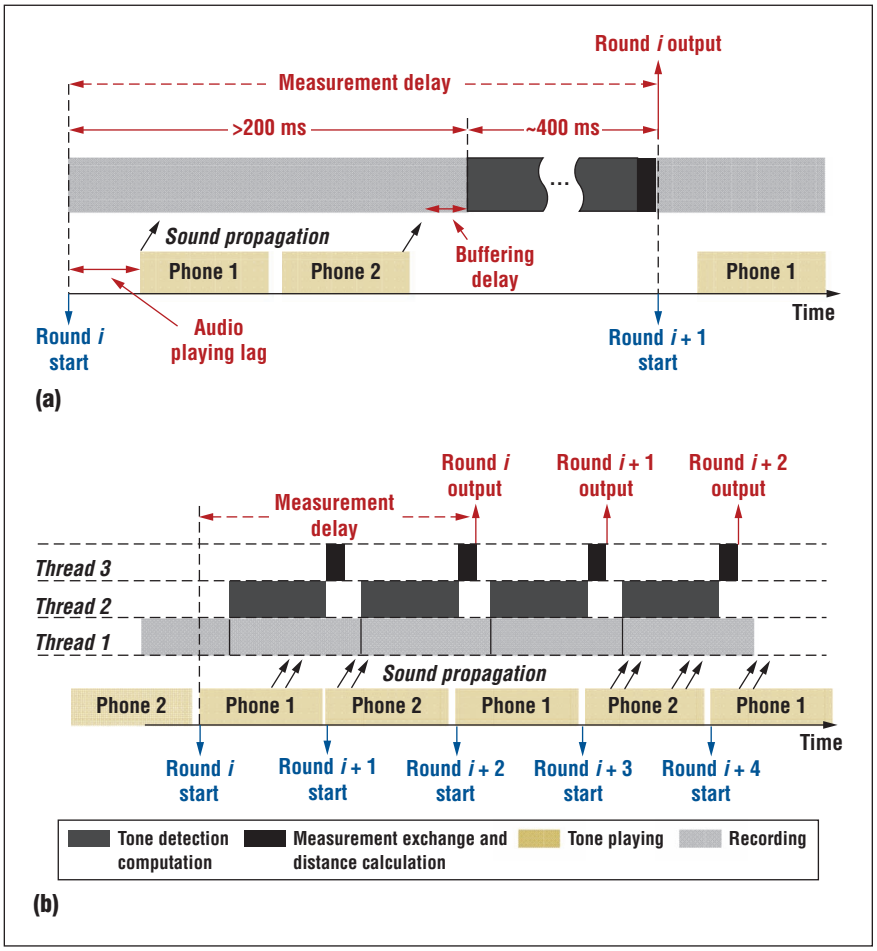


Figure 1. Execution strategies for measuring distance: (a) The traditional strategy first plays and records a tone and then computes and exchanges it, while (b) the FAR system reduces delays using efficient acoustic signal processing algorithms and execution strategies.

As Figure 1a shows, the traditional execution strategy first plays and records the tone and then computes and exchanges it. So, two phones start the recording step at the same time, and then they send out the tones one after another (the tone-playing step). After completing the recording, each phone calculates the exact local timestamp when each tone was received by applying a cross-correlation algorithm to its recorded sound samples (the tone-detection computation). Finally, the computed timestamps are exchanged and the distance between A and B is computed (the measurement exchange and distance calculation).

The problem is that each measurement comprises a number of components, which together result in a prohibitively large measurement delay and thus low measurement frequency. The total delay is not only affected by the playing and recording time, tone detection time, measurement exchange/distance calculation time, and the tone length, but also affected by the speaker's audio playing lag and the microphone's buffering delay. In total, the measurement lag with existing schemes is between 600 milliseconds and 1 second, which is not suitable for fast-speed mobile-motion games.

SMARTPHONES

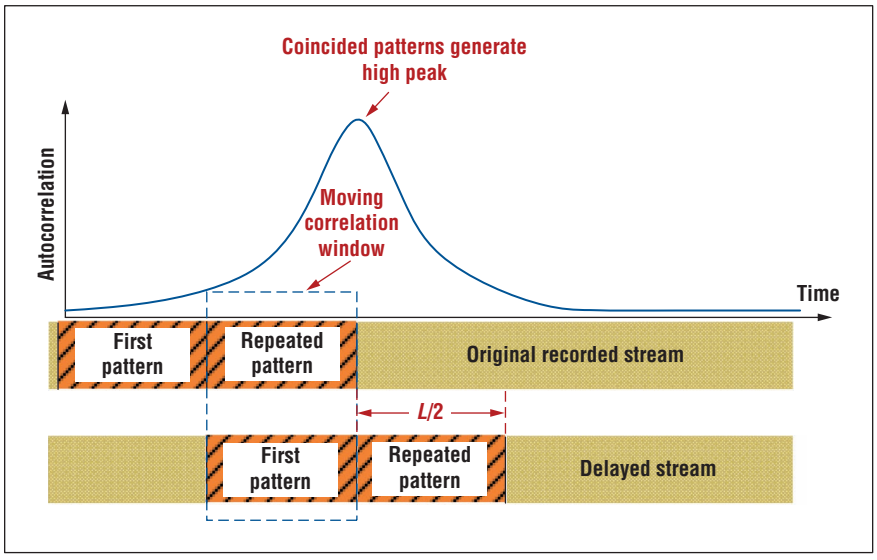


Figure 2. Autocorrelation-based tone detection. When sliding the correlation window at a certain time, the repeated pattern in the original stream will coincide with the first pattern of the delayed stream, which generates a high correlation peak.

If we want to achieve a measurement frequency above the 10 Hz required for mobile-motion games, we must address these delays across the board. The FAR system accomplishes this through an efficient system architecture and set of core acoustic signal processing algorithms. It can take distance measurements at a rate of 12 Hz with a 2 cm median error, and the measurements are based on acoustic signals. It can withstand environmental noise of the same power level as the tones (for example, due to players or spectators talking while playing), multipath (as encountered in small rooms), and the Doppler effect.

The key idea is that instead of running cross-correlation, we use a more computationally efficient autocorrelation primitive to detect the tones. Figure 2 shows the basic autocorrelation methodology we use.

In autocorrelation, we let each phone send out a tone consisting of a pseudorandom sequence followed immediately by an exact copy of this sequence. To detect this tone, the receiver phone records the incoming sound stream $X(t)$ and correlates it with $Y(t)$—which equals $X(t - L/2)$, a delayed version of $X(t)$—in a moving correlation window of width $L/2$ (where L represents the tone length and $L/2$ is the delay). Because each tone has two identical sequences, at a certain time slot, one of the repeated patterns in $X(t)$ will match the first pattern in the delayed stream $Y(t)$, thus generating a high autocorrelation peak.

The benefit of using autocorrelation is that we can compute the correlation for any time slot in $O(1)$ time, which is much more efficient than a cross-correlation algorithm, with a running time of $O(L)$. This lets us detect the tones in real time as they arrive. However, autocorrelation does not give us an exact measurement of the tone's arrival time because of its much flatter peak.

Therefore, after applying autocorrelation, we employ a small-scale cross correlation in a narrow search window, centered on the smoothed autocorrelation peak to detect the precise tone location. We select an empirical size of the search window to minimize the cross-correlation overhead while keeping the tone misdetection ratio low. The computation overhead of the combined algorithm is a little bit higher than that of the (inaccurate) pure autocorrelation algorithm, but it's still much smaller than the original cross-correlation algorithm.

A PIPELINED STREAMING EXECUTION STRATEGY

To further boost the measurement frequency and reduce lag, we introduce two additional improvements in the execution strategy.

We use separate threads to handle playing, recording, tone-detection computation, and measurement exchange/distance calculation (see Figure 1b). In the recording thread, the audio recorder receives the tone signals and periodically fills in a predefined audio buffer. Once the buffer is filled, the computation thread processes the buffered sound samples, while the recording thread continues filling in the next buffer. Once the computation is finished, a separate thread exchanges the tone arrival times with the other phone using Wi-Fi and outputs the final distance result. Because all the procedures are pipelined, the FAR system's measurement frequency is significantly improved.

Also, instead of blocking on computation until completion, as the traditional strategy does, the FAR system implements a streaming mode to continuously send and record tones. The **Play()** API of the audio driver is called at a frequency similar to the tone length, so one tone will be sent right after another. This helps in eliminating the audio-playing lag from the overall measurement time.

Similarly, we also implement the audio recorder to work in a streaming fashion, continuously recording sound samples for computation. As a result, if there's no lag between tones, measurements can theoretically be conducted at a frequency equal to $1/L$, provided tone detection is fast enough to support such a rate.

Our system also implements several additional optimizations to reduce delay, including minimizing tone length

while still reserving measurement accuracy and selecting the minimum supported buffer size of hardware to reduce buffering delay.

REDUCING MEASUREMENT ERRORS

Another important challenge we tackled is robustness. A real-time motion game is not fun if there are frequent measurement errors.

Mobility Robustness

One unique challenge to distance estimation of motion gaming is the Doppler effect caused by player movement. When there is a relative movement between the two phones (specifically, their sound players and recorders), the Doppler effect happens. Intuitively, when two phones are moving toward each other, the sound wave arrives at the recorder earlier than expected. Because the recorder is recording at a constant rate, the sound wave generates fewer samples than expected, so it seems compressed. Thus, we need to shorten the offset used in the autocorrelation calculation. Similarly, when the phones are moving further away, the tone is diluted so a longer offset is needed.

We define the Doppler offset (D_{offset}) as the offset required for autocorrelation. In a static scenario, D_{offset} is $L/2$, but when the phone moves, D_{offset} won't equal $L/2$ because it will depend on the velocities of the two phones. Assuming the maximum speed of a player's hand is 2 m/sec, the possible range of D_{offset} is $[L/2-3, L/2+3]$. If D_{offset} is not correctly set in autocorrelation, tone misdetection happens, so we have to compute autocorrelation with the appropriate offset.

Given that $D_{offset} \in [L/2-3, L/2+3]$, in the worst case, we can use seven parallel autocorrelators. However, each additional autocorrelator comes with a computation overhead, so we apply a *predictive Doppler estimate procedure*. The idea is to predict the likely Doppler shift based on the recent history of distance measurements and to only execute the autocorrelators likely to match the predicted Doppler shift.

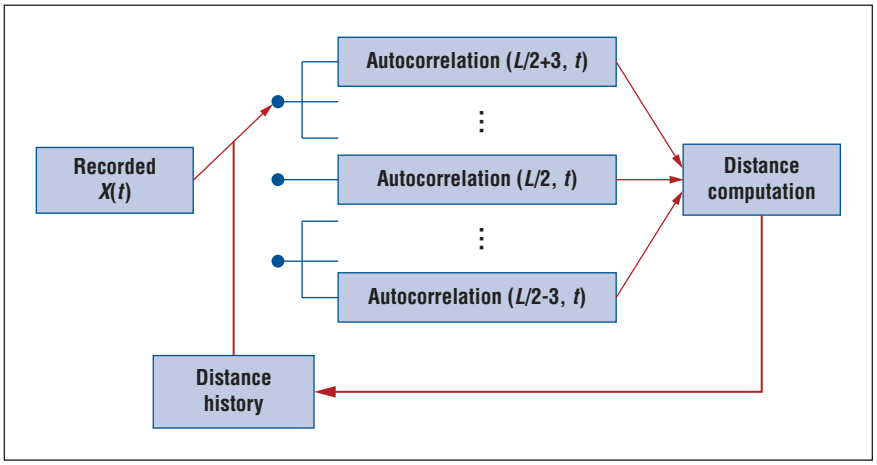


Figure 3. Predictive parallel autocorrelators. The distance history is leveraged to predict whether sound will be diluted or compressed and, subsequently, which group of autocorrelators to use.

As Figure 3 shows, we categorize the Doppler-shifted tones into two groups: compressed and diluted. We use two groups of autocorrelators to deal with them. For compressed cases, we use $D_{offset} = L/2-1$, $L/2-2$, or $L/2-3$; for diluted cases, we use $D_{offset} = L/2+1$, $L/2+2$, or $L/2+3$. In addition, we always run the original autocorrelator with $D_{offset} = L/2$. Using this method, we can reduce the total overhead by nearly half. In fact, our evaluation suggests that almost all of D_{offset} are within $[L/2+2, L/2-2]$, and thus only three parallel autocorrelators are needed to recover from the vast majority of Doppler shifts.

Environmental Robustness

We also address two important environmental factors that can impair distance measurements: ambient noise and multipath effect. We apply a high pass filter (a 9th order Butterworth filter) to filter out signals with frequency lower than 1.5 kHz, which is the typical frequency range of ambient noise. We leverage the fact that the multipath components usually have low power and signal quality after reflection, and empirically set a power threshold to filter out the multipath components experienced when playing in actual scenarios.

The FAR system can also deal with occasional tone losses using a binary coding method by assigning a unique pseudorandom sequence for each phone. If detection is flawless, each phone will see alternating codes. If a phone receives any two consecutive tones whose codes are the same, the phone detects a missed tone condition and accounts for this in the distance calculation.

PERFORMANCE EVALUATION

In the home entertainment domain, nonmobile console gaming systems such as the Wii, PS3 Move controller, and Kinect have embraced player motion-based game play, developing a variety of schemes for player localization. However, these console gaming systems are tethered to a fixed console-based infrastructure. Our FAR system, on the other hand, can compute distances between mobile devices without any infrastructure.

We compare the FAR system with Kinect to check whether it is capable of accurate measurements while players are moving. We obtain Kinect measurements with the Kinect software development kit, which allows programmatic

SMARTPHONES

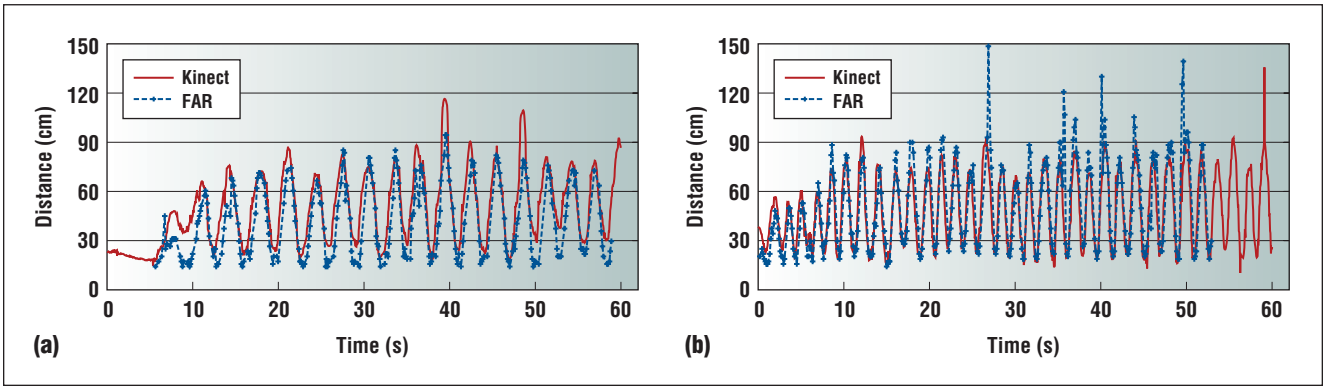


Figure 4. In-situ game-play measurements. Simultaneous distance measurements with FAR and Kinect of players engaged in SwordFight.

collection of the human skeleton coordinates as seen by the Kinect camera.

In our in-situ game-play experiments, we invited six players from the UC Santa Barbara Link Lab to conduct nine in-situ rounds of measurement. We asked players to try out the SwordFight game. During the game, the FAR system provided the in-game distance measurements, while Kinect independently recorded the players' skeletal hand positions. Figure 4 compares the corresponding FAR and Kinect measurements for two representative traces. Despite the occasional mismatches, the FAR system's output is very close to Kinect, which demonstrates the effectiveness of our design.

In some limited cases when opponents are facing each other, FAR can deliver more accurate distance measurements than Kinect. This is because opponents facing each other can obstruct the line-of-sight between the Kinect camera and the phones in their hands. In the in-situ experiments presented in Figure 4, we asked that the players compensate for this Kinect limitation for the sake of experimentation: both players were instructed not to obstruct the line-of-sight between the Kinect and their phones.

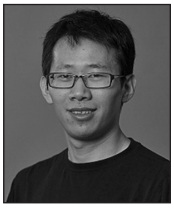
Our work here is merely a starting point—open challenges remain. One issue is that continuous acoustic tones are noticeable, because microphones and speakers on commodity phones only support the audible frequency range. A possible approach is to embed these tones in game music.

A second issue is that line-of-sight blockage between the phones degrades the measurement accuracy, and players can accidentally or purposefully block the microphones and speakers. In future work, we will create a blockage warning or cheat-detection protocol. In addition, we are interested in extending the API to permit simultaneous ranging between more than two phones.

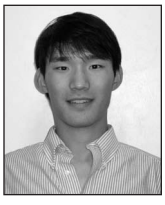
Lastly, we are working on designing and prototyping additional mobile-motion games on top of our phone-to-phone gaming API. P

REFERENCES

1. Z. Zhang et al., "SwordFight: Enabling a New Class of Phone-to-Phone Action Games on Commodity Phones," *Proc. 10th ACM Conf. Mobile Systems, Applications, and Services* (MobiSys 12), ACM, 2012.
2. T. Beigbeder et al., "The Effects of Loss and Latency on User Performance in Unreal Tournament 2003," *Proc. 3rd ACM SIGCOMM Workshop on Network and System Support for Games* (NetGames 04), ACM, 2004, pp. 144–151.
3. C. Peng et al., "Beepbeep: A High Accuracy Acoustic Ranging System Using COTS Mobile Devices," *Proc. 5th Int'l Conf. Embedded Networked Sensor Systems* (SenSys 07), ACM, 2007, pp. 1–14.
4. J. Qiu et al., "On the Feasibility of Real-Time Phone-to-Phone 3D Localization," *Proc. 9th ACM Conf. Embedded Networked Sensor Systems* (SenSys 11), ACM, 2011, pp. 190–203.
5. L. Jones and S. Lederman, *Human Hand Function*, Oxford Univ. Press, 2006.



Zengbin Zhang is a PhD candidate at UC Santa Barbara. Contact him at zengbin@cs.ucsb.edu.



David Chu is a researcher at Microsoft Research Redmond. Contact him at davidchu@microsoft.com.

Xiaomeng Chen is a PhD student at Purdue University. Contact her at chen1058@purdue.edu.

Thomas Moscibroda is a lead researcher at Microsoft Research Asia and a chair professor at Tsinghua University. Contact him at moscitho@microsoft.com.