Multiphono: Relative Positioning of Co-located Mobile Devices

Adam O'Sullivan
Department of Computer Science
A.V. William Building
University of Maryland
College Park, MD 20742, USA
adamo@cs.umd.edu

ABSTRACT
Mobile devices continue to become increasingly common in our daily lives, and many individuals carry a mobile device most of the time. This leads to situations where multiple co-located mobile devices are present. We view these situations as an opportunity for new types of interactions with mobile devices. Currently, determining relative positioning information regarding two co-located devices is a difficult task, and many solutions require the use of specialized hardware or existing external infrastructure to function. There are many potential applications that could make use of relative positioning information, if it were easily accessible using hardware already included with mobile devices.

In this paper, we present Multiphono, a mobile application that utilizes front-facing cameras and wireless communication on mobile devices to enable the determination and communication of relative positioning information. The cameras capture images of the ceiling or scene above the devices, after which the images are transferred between the devices and then stitched together. From this process, the relative positions of the two devices can be determined. We developed a proof-of-concept implementation, which demonstrates that this technology works well when the devices are in close proximity, and could be refined to function in other situations.

ACM Classification Keywords
H.5.3 Group and Organization Interfaces: Collaborative Computing

General Terms
Human Factors; Design.

INTRODUCTION
Mobile devices are becoming ubiquitous in our daily lives. It is very common for an individual to carry at least one mobile device, often a mobile phone, much of the time. As such, it is likely that multiple co-located individuals will each have a mobile device. Situations such as these have helped increase the popularity of many mobile applications. For example: photo sharing, contact sharing, music sharing, collaborative gaming, and instant messaging are all useful applications in this context. Given their multi-user nature, many of these applications utilize some form of wireless communication for interaction. Applications of this type are likely to continue to increase in popularity.

While keyboards and keypads (whether implemented with physical buttons or on a touch screen) are still some of the more typical forms of interaction, several other features are becoming more common in commodity mobile devices. A few examples include these additional types of sensors: cameras, gyroscopes, accelerometers, and Global Positioning System (GPS) receivers. These new types of sensors provide opportunities for alternative forms of interaction.

Location information can be extremely important on mobile devices in a variety of situations. The aforementioned GPS receivers can determine a device's approximate absolute position in the world. However, there are many applications where a given device's position relative to other devices is even more useful. A few examples include: gesturing (for a variety of purposes), pairing devices, gaming, and sharing or displaying data objects.

Even with these types of additional sensors, it is non-trivial to determine the relative positioning of two co-located mobile devices. However, through the use of some of these sensors, we can get a much closer fix on the relative positioning.

This raises the primary question addressed in this paper: How can the relative position of co-located devices be determined? We define the problem as follows: two devices located near each other need to know their own location and orientation with respect to the location and orientation of the other device. As previously mentioned, determining the relative positioning of two devices is difficult using only the hardware found in modern mobile devices. Specialized hardware, such as in the ultrasonic

"Bat" location sensor system's [20] use of ceiling mounted receivers in combination with pulse emitting mobile devices, can make this task easier, but often has a high cost and may require an installed external infrastructure to function. Another example of this type of external infrastructure makes use of mounted cameras, computer vision, and geometric world modeling, as described in the EasyLiving project in [20]. Moreover, specialized hardware for the purpose of interacting with these types of systems, is not commonly found in mobile devices today. These factors make using specialized hardware impractical for this task. Additionally, absolute location information provided by common existing systems (e.g. GPS, Wi-Fi) generally lacks the precision necessary to determine the relative positioning of co-located devices.

In this paper, we explore the idea of using hardware commonly found in modern mobile devices to determine relative positioning information between two co-located mobile devices. We examine a proof-of-concept type implementation and view this technology as a building block that can be used to create some unique and interesting applications. We will briefly discuss several possible scenarios / examples that would particularly benefit from this technology:

Information exchange - Transferring an image or some other data object (document, contact information, etc.) from one device to another could be made much simpler, requiring very little configuration. For example: Use a flick gesture on an image on a device (A) to transfer it to another device (B), which is located in the relative direction of the flick. This use is somewhat similar to that described for photo sharing in [16].

Creating a large display from many smaller devices - (similar to the Junkyard Jumbotron [17] and display tiling [16] systems). This is accomplished by placing several mobile devices together on a surface, then displaying an image using the combination of all the devices' displays. This would use the relative position of the devices to determine which portion of the image to display on which device.

Augmented Reality - The mobile device could display a live video stream to the user, with overlays marking the locations of other co-located devices. This information would be determined using the relative positioning information of the devices. This could be used to identify both devices and potentially the users of said devices.

Gaming - Games could utilize the relative location of multiple devices to extend the gaming area. For example, one could create a large pong playing area using multiple devices. Another example would be to create a racing game with a track or map that spans multiple devices. Sifteo cubes [15] can run some games that demonstrate this concept using specialized hardware.

Interacting with a larger display - A user could interact with a larger touch surface / display using his or her own smaller mobile device. The relative position could be used to determine which device touched or gestured toward the larger display. Much research has been done in this area, and relative positioning information could augment the systems described in [1, 3, 5, 9, 10].

Pairing - Being aware of the relative position of devices enables users' devices to know the location of other nearby devices. This can enable an easy means of pairing two devices for a game or other data transfer (e.g. Bluetooth pairing). Users could hold the two devices next to each other for a few seconds to pair. This would be particularly useful, because as Lucero et al. [16] observed, unless users can spontaneously join a group / start interacting with others quickly, they may lose interest. Relative positioning information would make joining a group and connecting with other devices a very quick and simple process.

RELATED WORK

In this section, we examine some significant research that is related to this work in a variety of ways. Some of these projects are similar in their objectives and goals, while others utilize relevant technologies. Many of these systems could be enhanced if the devices involved had access to their relative positioning information.

PhoneTouch [1] is a system that allows for interaction with a touch table surface, using touches from a mobile device, along with traditional finger touches. The mobile devices and touch table surface use wireless communication in combination with accelerometers to determine when a mobile device touches the interactive surface. The devices synchronize their clocks in an initial calibration phase, thereby allowing touch events received by the touch table surface to be corroborated with the mobile device sensor data (e.g. a sharp spike detected by the accelerometer) to determine when a specific mobile device touches the surface (this is used to distinguish between multiple mobile devices). This is an interesting example of how to use accelerometers in combination with wireless communication to facilitate interaction between co-located devices.

Bump [13] is a smart phone application that allows users of two mobile devices to transfer contact information or other files by physically bumping their devices together. Each of the bumped mobile devices record a variety of sensor data (e.g. accelerometer, gyroscope, GPS / location information), which is then sent to the central Bump servers on the Internet. The Bump servers analyze the sensor data and use it to match the two mobile devices, by determining which two devices were involved in the same physical bump. Once the devices have been matched, the information is then transferred from one device to the other via the Internet. This technology is a very effective

example of using sensor data and wireless connectivity to allow devices to communicate and transfer data without going through a complicated configuration or pairing process.

The ConnecTable [2] system enables the coupling of single user displays to form a larger display area and shared workspace. A ConnecTable is an adjustable, pen-operated display, mounted on a base with wheels. ConnecTables can be couple dynamically and on demand. Coupling is accomplished using built-in RFID tags (and corresponding readers) to detect nearby devices. When connected, information objects can be moved from one display to another in the common workspace. These concepts, such as dynamically coupling displays, can be applied to mobile devices without the use of specialized hardware, if the devices are aware of their relative positioning.

Sifteo cubes [15] are an interactive gaming system, comprised of 1.5 inch blocks, each featuring a clickable color screen, mobile CPU, accelerometer, wireless communication, and a near-field object detecting sensor. These blocks interact with each other based on their position relative to other blocks. Users interact with the blocks by clicking the screens, shaking or tilting the blocks, and arranging the blocks next to each other. Sifteo cubes are used for a variety of different types of games, making use of their specialized hardware for this purpose. Some example games involve building out a map of a virtual world using multiple cubes, transferring objects between cubes, using cubes to simulate real objects (e.g. turning a key, opening a treasure chest), and so on. Sifteo cubes demonstrate many interesting applications and features that make use of relative positioning information. However, they are clearly specialized hardware, made specifically for this purpose. With Multiphono, we are trying to accomplish many similar features, using commonly found mobile devices instead.

Junkyard Jumbotron [17] is a system that allows a user to create a large display by combining several smaller displays (such as those found on laptops, smart phones, or tablets). Once the devices / displays have been positioned, each one visits a certain URL, which displays a unique visual code. The user then photographs the displays while they are showing these visual codes, and sends the photograph to a certain email address, where it is processed. The software then determines the relative positioning of each of these devices by looking at where each unique visual code is displayed in the image taken by the user. The displays are then updated with their respective portions of a larger image, thereby creating a larger display from several smaller devices. Similarly, Lucero et al. [16] describe a system of tiling devices to display a composite larger version of an image. This system, however, is based on placing the devices in a pre-determined configuration, with the edges of each device being flush. In both of these systems, multiple devices are combined to form a larger display. If these devices were aware of their relative positioning information in real time, they would be simple to configure, and could be rearranged dynamically without further configuration steps. This could lead to a wide variety of unique applications.

The Swordfish [3] framework allows for user tailored workspaces in Multi Display Environments (MDEs). User defined connections (referred to as "lightweight personal bindings") are made from the edge of one display to another. These personal bindings can be created and modified quickly and easily, thereby allowing users to define workspaces according to their preferences. This system allows multiple users to have individual functional mouse pointers on the same display, simultaneously (with some limitations relating to the operating system). However, if the devices had their relative positioning information, the creation of these personal bindings could be completely automated, and allow users to quickly join and interact with an MDE.

In this work, User-Defined Gestures for Connecting Mobile Phones, Public Displays, and Tabletops [4], the authors investigated whether gesturing with a mobile phone can help users perform complex tasks involving two devices (phone to phone, phone to tabletop, phone to public display). This work focused primarily on the types of gestures performed by participants, along with their feedback. This data is used to discuss which sensors are best used for gesture recognition in a phone. The authors found that for the majority of gestures they observed, relative distance from the mobile phone to the target device changed. Thus, recognizing such gestures requires sensors that can estimate the relative distance between said devices. Additionally, they found that many gestures relied on location changes, which could be detected using accelerometers. Changes in rotation were also common, and can be detected with accelerometers, gyroscopes, and magnetometers. Absolute positioning in space was used in some gestures, and is often difficult to determine precisely in real time. One could use camera based techniques with natural features or special markers in this case. Gesturing could be an even more effective interaction technique if the mobile devices involved (e.g. device gestured with, device gestured toward) were aware of their relative positions

The RELATE [5] model is designed for spontaneous interaction between mobile devices and services available in the user's environment using "spatial references." This is based on the spatial relationship of a user's device to other nearby devices (e.g. those providing services). The authors used specialized hardware for exchanging ultrasound and radio signals between devices (Near Field Communication (NFC), beacons / tags). These signals were used to infer the relative spatial relationship among devices. The signals were then used to display the relative locations on the user's mobile device GUI (e.g. a device in front of a person would appear in some form at the top of the screen on the mobile

device -- with the relative positions on the screen updating as the device is moved). Despite its reliance on specialized hardware, this system demonstrates how useful relative positioning information can be.

BeepBeep [6] is a high accuracy acoustic range sensing system that has two devices each send and receive sounds to infer the distance between them. This is accomplished using only low-cost hardware that is commonly available on mobile devices -- microphones and speakers. BeepBeep can achieve an average accuracy within two centimeters over a range of more than ten meters. Point&Connect [7] is a system based on the work from BeepBeep. It is an intention based device pairing system that is also possible on typical mobile devices without requiring specialized hardware. To pair one's phone with another nearby device in this system, one must perform the gesture of pointing the phone in the direction of the target. The system will receive the gesture, select the correct target, and pair the devices. This target selection is accomplished by measuring the maximum distance change based on acoustic signals. In other words, the device at which the user is pointing should report the largest relative distance change between two emitted beeps (early and late in during pointing gesture). This system makes good use of the existing hardware on commodity devices to determine some positioning information. However, this approach is limited in that while it can determine the distances between devices, it cannot determine their relative positioning.

In the Gesture Connect [8] system, the authors combine the use of NFC tagging with a 3 axis accelerometer for gestures in order to easily connect to and control objects from a user's mobile device. The general idea is simply scan the NFC tag, then use the mobile device to either perform a gesture for common or simple tasks, or use the on screen interface to interact. The goal is to streamline interaction with objects using an individual's mobile device. This work demonstrates the integration of commodity sensor data (NFC and accelerometer) to perform tasks.

The ARC-pad [9] system supports controlling a large screen using a mobile device's touch screen. This is facilitated by simultaneous use of absolute and relative positioning, without any explicit mode switching -- pointing / touching the screen for absolute positioning of the cursor, or sliding / swiping for relative positioning of the cursor. Generally, users will utilize absolute positioning for rapid movement across large distances and relative positioning for more fine position control. This approach could be applied in a variety of scenarios involving co-located devices.

Wallshare [10] is a client server system based on a "shared zone," that is projected on a wall or large screen. Participants can interact with the shared zone using a mobile client application running on a mobile device. Each user gets his or her own pointer on the shared zone. Users can transfer files to and from the shared zone. When a new resource is uploaded, it appears on the shared zone. To download a resource, users simply double click the resource in the shared zone using the client application. Additionally, users can post chat messages or notes to the shared zone. The server piece is made up of two components: resource sharing and zone visualization. This project has uses in many areas, such as entertainment, games, science, and education. The concept of a shared zone is well suited to situations involving co-located users with mobile devices and could be extended to utilize the relative positioning information of users' devices.

The Matrix Desk [11] is a device which is designed to address three problems in a collaborative learning environment: The "screen sharing problem," the "desk configuration problem," and the "input identification problem." The device is an embedded computer with a large desktop surface display, which can receive input from a digital pen. Each desk is equipped with wheels to facilitate connecting and arranging. Students have digital pens with unique electromagnetic identifiers which the displays can identify (solving the input identification problem). There are sensors on the sides of each Matrix Desk, along with RFID to facilitate connecting desks. These are used to determine when desks are attached and how they are connected (e.g. on which sides). The embedded computers communicate with a centralized server for some tasks including organizing the connected desks / analyzing the layout of the shared workspace for the best viewing experience as well as for transferring data objects. Connected desks can take a variety of predetermined shapes, however any two connected sides must be flush. Connected desks can form a square, rectangle, T, L, or U type shapes. This concept can easily be extended to mobile devices that are aware of their relative positioning. In fact, connected mobile devices could be aligned in any arrangement, and would not be limited to predetermined patterns. Also, additional relative positioning information could help address the three problems discussed by the authors in this work.

GOALS

In exploring the concept of determining the relative position of two mobile devices, we set the following goals:

- Be able to determine a device's relative position with respect to another device
- Make use of hardware commonly found in smart phones
- Do not require any specialized hardware or installed infrastructure

TECHNICAL APPROACH

In this section, we will discuss our experiences in investigating, designing, and testing our ideas while trying to achieve our goals.

When we began our investigation, we quickly ruled out the idea of using specialized hardware, as it is impractical, unlikely to be included on commodity devices, oftentimes expensive, and may require an installed external infrastructure to function. Instead, we wanted to develop a system that would make use of hardware that is already commonly found on devices today. We considered the sensors we would likely have access to on a smart phone: microphones, accelerometers, gyroscopes, GPS receivers, cameras, and Wi-Fi.

An initial examination of using absolute location information as provided by common existing sensors (e.g. GPS, Wi-Fi) showed that the positioning information lacked the precision required to determine relative positioning of co-located devices. Next, we considered using accelerometers and gyroscopes to calculate dead reckoning / inertial positioning -- essentially to start with the two devices in a known position (e.g. next to each other), then track their movement in relation to that initial position using accelerometers and gyroscopes (e.g. the device has moved 1 meter in the X direction, and 2 meters in the Y direction). Using that information, we could then determine the relative positioning of the two devices.

We began testing an initial implementation of the dead reckoning / inertial navigation approach in a very simple case: two devices moving in only two dimensions, on a flat surface. Unfortunately, we found that this approach was far too imprecise at the scale in which we were interested. The accelerometers and gyroscopes generated a great deal of noise in the sensor data, which made it difficult to identify small movements. Moreover, significant error accumulated very quickly. This testing indicated that the dead reckoning / inertial navigation approach was unlikely to work for what we were trying to accomplish.

Additionally, we considered using the device's microphone to determine relative location. However, since most devices only have one microphone, we would generally only be able to determine the distance between two devices, not the direction, and thus not their relative positioning.

This lead us to investigate the use of cameras and image processing for determining relative positioning information. We decided to approach the problem using the front-facing cameras on mobile devices to take images of the ceiling or scene directly above each device. The devices would then transmit their respective images to each other, and perform something along the lines of image stitching [18]. This computation is used to understand how the images are positioned in relation to each other. Using this information, we have a good idea of where the cameras, and thus the devices, are positioned relative to each other. Our initial tests were promising, and lead us to improve our approach and implementation.

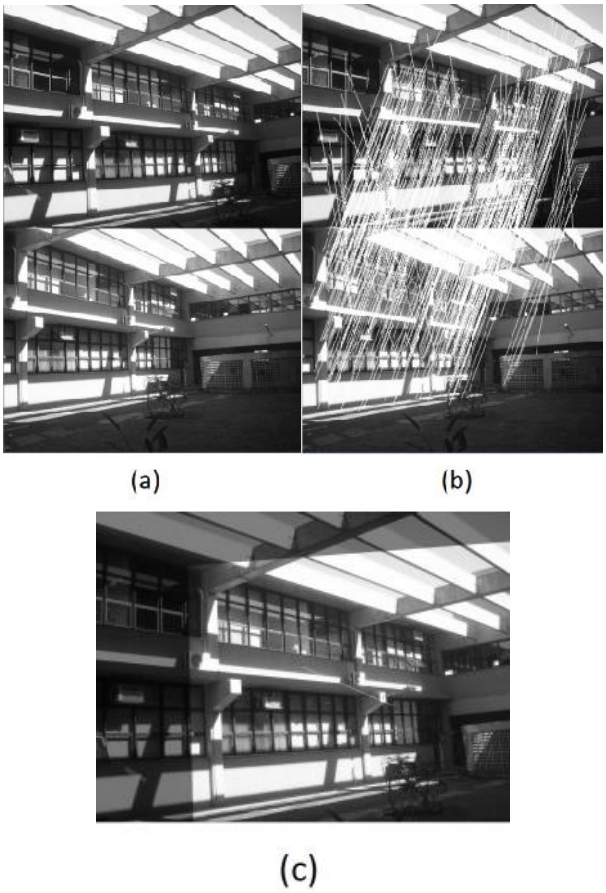

Figure 1. Sample images to demonstrate the image stitching process. (a) The original sample images. (b) The sample images with features having been detected in both images and matched with the their respective counterparts in the other image (matching features are connected between the images using thin white lines.) (c) The final image, where one image has been warped to match the other, and then one image is overlaid on the other. The thin white line connects the center points of both images, prior to any warping. Starting from image 1, following this line takes you in the direction of image 2 in relation to image 1.

IMPLEMENTATION

Multiphono is implemented as a mobile application on the iOS operating system. It utilizes Wi-Fi for communication between devices and the OpenCV [12] library for image processing. We used two Apple iPod Touch (4th generation) devices for our test implementation.

When the Multiphono application is first launched, it uses Apple's Bonjour [19] protocol to attempt to discover and locate another device running Multiphono. Once the other device has been located, a socket connection is established between the applications, making use of the GCDAsyncSocket library [25]. The applications then begin regularly capturing images from their respective devices'

front-facing cameras, by making use of Apple's AVFoundation [22], CoreVideo [23], and CoreMedia [24] frameworks to create an AVCaptureSession and interface directly with the camera. Next, the each application will store its captured image in memory and transmit it across the socket connection to the other device. Each device will then have a copy of both images (this is represented in Figure 1a.). When the application receives an image from the connection, it begins using OpenCV [12] for image processing. It uses the OpenCV implementation of SURF (Speeded Up Robust Features) [14] to calculate and extract feature points (via the cvExtractSURF(...) function) from both the local image and the received image. These feature points are generally specific areas in the image, such as edges or points that make up objects in the image.

The next step is to match the feature points from one image to another. This is accomplished using a simple nearest neighbor search (though other algorithms can be applied here as well) (This is represented in Figure 1b. The matched features are connected using a thin white line. The features are located at the endpoints of each line).

Once the feature points have been matched, we compute a transform (from one image to the other) called a homography. This homography transform maps straight lines to straight lines. The homography is computed using the RANSAC [26] (Random Sample Consensus) algorithm (through the use of the OpenCV cvFindHomography(...) function).

This homography relates the pixel coordinates in image 1 and image 2. When the homography matrix is applied to every pixel to image 2, image 2 becomes warped to fit image 1. This is represented in Figure 1c. Image 2 has been warped to fit image 1, and is then overlaid and blended with image 1. In this example, a white line is drawn from the center of image 1 to the center of image 2 (before it was warped). This points us in the direction of image 2 as it relates to image 1, starting from the center of image 1. We can use this data to determine the relative location of each camera, which in turn tells us the relative location of each device (as the cameras are built in to the devices). This image stitching type process is described in further detail in [18]. A full working example of this type of code is distributed as a sample with OpenCV (in find_obj.cpp).

Currently, the Multiphono user interface simply displays the stitched together images (with the most recent locally captured image at the center, and the received image transformed according to the homography matrix and overlaid with the local image). Additionally, a line is drawn in the computed direction of the device which sent the image (as seen in Figure 1c). This is simply to facilitate the testing and investigation of this technology. Many types of applications can be built on top of this framework.

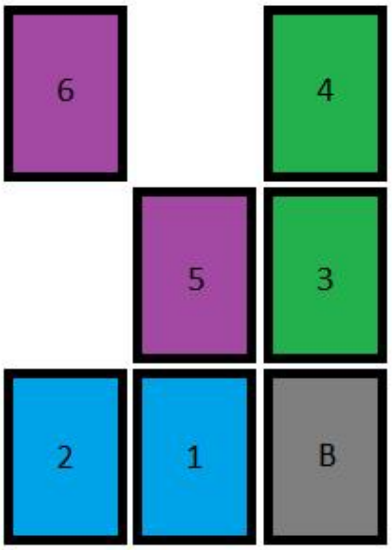

Figure 2. The testing position layout for the two devices. One device was always placed at the base (B) position, while the other was placed in one of the numbered positions.

Figure 3. An image of the ceiling / area above the devices, as was visible during the evaluation. The devices were tested on a table that was partially underneath a glass shelf (pictured from the devices' perspective).

EVALUATION

The primary test involved running Multiphono on two devices, each placed in one of a set number of positions, one device always being placed in the base (B) position, and the other in one of the numbered positions (see Figure 2). The ceiling (pictured in Figure 3) was approximately 1 meter above the table surface on which the devices were placed. For each set of positions, we tested Multiphono for one minute. At each second mark during the testing

minute, we recorded whether the application running on the device in the base (B) position had correctly determined its position relative to the other device. If the application drew a line pointed in the correct direction of the other device (within 30 degrees of the other device's center), we recorded a success, otherwise we recorded a failure for that second. We performed this test at each set of positions three times, and averaged the results. Table 1 shows the results for each set of positions.

Positions	% of Minute Successful
B, 1	87.22%
B, 2	78.33%
B, 3	70.0%
B, 4	38.33%
B, 5	48.33%
B, 6	< 10.0%[1]

Table 1. Shows the average percentage of each minute that the base device successfully determined its position, relative to the other device. ([1] in this particular condition, the software did not perform well, and was difficult to measure consistently)

Examination of the results show a clear trend: The closeness of the two devices is correlated with their success at determining relative positioning. We believe this is because the closer the devices are together, the closer their cameras are, which leads to the images they capture being more similar (in the sense that they capture many of the same parts of the scene from only slightly different perspectives), and thus easier to stitch together.

Additionally, we tested Multiphono in a variety of other ways (though in a less formal manner). Some of the other tests included moving the devices around each other as well as rotating the devices to various angles in real time and verifying that the devices updated correctly with respect to their relative positions. Throughout this testing, we found the previously mentioned trend to be true as well. A device's success at determining relative positioning is correlated with its closeness to the other device in our implementation.

FUTURE WORK

While we have discussed the strengths and limitations of our Multiphono implementation, it is clear that additional work needs to be done before this technology is fully realized. Many of the potential areas for improvement are closely related. One improvement would be to capture higher resolution images from the front-facing cameras on the mobile devices. This would allow for more accurate matching and image stitching. However, processing higher resolution images would require a more powerful CPU. Moreover, a faster CPU would allow developers to build complex applications on top of this technology. In our implementation, the CPU is occupied by the real time image processing that is taking place, leaving very little processing ability for applications running simultaneously. Significantly more powerful mobile processors will likely become a reality, though, as mobile processors continue to improve (both in terms of speed and number of cores).

Another potential area for improvement would be for the application to utilize the GPU for processing the images, as opposed to simply using the main CPU. This would also likely allow for faster image processing, as well as the processing of higher resolution images.

Additionally, it may be possible to incorporate sensor data from sensors other than the front-facing camera. Accelerometer and gyroscope data may be used to enhance the application's model of the situation and enable a more accurate determination of relative positioning. Integration of sensor data could also lead to a variety of forms of interaction as well.

As discussed in the introduction, a wide variety of applications could be made possible by or benefit from the use of this technology.

CONCLUSION

We presented Multiphono, an application which demonstrates that the relative positioning of two devices can be determined using the devices' front-facing cameras. We described our proof-of-concept implementation, and an initial evaluation of the accuracy and effectiveness of this technology. Multiphono meets the three goals we set out to accomplish and can be used as a building block for new types of applications. This implementation indicates that while more work is needed, this technology could work very well for determining the relative positioning of co-located mobile devices in a variety of contexts.

REFERENCES

1. Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. 2010. PhoneTouch: a technique for direct phone interaction on surfaces. In Proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST '10). ACM, New York, NY, USA, 13-16.
2. Peter Tandler, Thorsten Prante, Christian Müller-Tomfelde, Norbert Streitz, and Ralf Steinmetz. 2001. Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST '01). ACM, New York, NY, USA, 11-20.
3. Vicki Ha, Kori Inkpen, Jim Wallace, and Ryder Ziola. 2006. Swordfish: user tailored workspaces in multi-display environments. In CHI '06 extended abstracts on Human factors in computing systems (CHI EA '06). ACM, New York, NY, USA, 1487-1492.
4. Christian Kray, Daniel Nesbitt, John Dawson, and Michael Rohs. 2010. User-defined gestures for connecting mobile phones, public displays, and tabletops. In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10). ACM, New York, NY, USA, 239-248.
5. Hans Gellersen, Carl Fischer, Dominique Guinard, Roswitha Gostner, Gerd Kortuem, Christian Kray, Enrico Rukzio, and Sara Streng. 2009. Supporting device discovery and spontaneous interaction with spatial references. Personal Ubiquitous Comput. 13, 4 (May 2009), 255-264
6. Chunyi Peng, Guobin Shen, Yongguang Zhang, Yanlin Li, and Kun Tan. 2007. BeepBeep: a high accuracy acoustic ranging system using COTS mobile devices. In Proceedings of the 5th international conference on Embedded networked sensor systems (SenSys '07). ACM, New York, NY, USA, 1-14.
7. Chunyi Peng, Guobin Shen, Yongguang Zhang, and Songwu Lu. 2009. Point&Connect: intention-based device pairing for mobile phone users. In Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys '09). ACM, New York, NY, USA, 137-150.
8. Trevor Pering, Yaw Anokwa, and Roy Want. 2007. Gesture connect: facilitating tangible interaction with a flick of the wrist. In Proceedings of the 1st international conference on Tangible and embedded interaction (TEI '07). ACM, New York, NY, USA, 259-262.
9. David C. McCallum and Pourang Irani. 2009. ARC-Pad: absolute+relative cursor positioning for large displays with a mobile touchscreen. In Proceedings of the 22nd annual ACM symposium on User interface software and technology (UIST '09). ACM, New York, NY, USA, 153-156.
10. Pedro Gonzalez Villanueva, Ricardo Tesoriero, and Jose A. Gallud. 2010. Multi-pointer and collaborative system for mobile devices. In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10). ACM, New York, NY, USA, 435-438.
11. Hercy N. H. Cheng, Yi-Chan Deng, Ben Chang, and Tak-Wai Chan. 2005. MatrixDesks: interactive computing desks toward one-on-two educational computing environments. In Proceedings of th 2005 conference on Computer support for collaborative learning: learning 2005: the next 10 years! (CSCL '05). International Society of the Learning Sciences 48-52.
12. OpenCV. http://code.opencv.org/projects/opencv/wiki, April 2013.
13. bump: The easiest way to share. http://bu.mp/company/, April 2013.
14. Herbert Bay, Tinne Tuytelaars, and Luc Van Gool12. SURF: Speeded Up Robust Features. Computer Vision – ECCV 2006, pages 404–417.
15. David Merrill, Emily Sun, and Jeevan Kalanithi. 2012. Sifteo cubes. In CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12). ACM, New York, NY, USA, 1015-1018.
16. Andrés Lucero, Matt Jones, Tero Jokela, and Simon Robinson. 2013. Mobile collocated interactions: taking an offline break together. interactions 20, 2 (March 2013), 26-32.
17. Rick Borovoy, and Brian Knep. 2012. Junkyard Jumbotron. MIT Center for Future Civic Media. Web 25 (2012).
18. Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. International Journal of Computer Vision 74.1 (2007), 59-73.
19. Apple Bonjour. https://developer.apple.com/bonjour/, April 2013.
20. Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward and Andy Hopper. 2001. Implementing a Sentient Computing System. Computer 34, 8 (Aug. 2001), 50-56.
21. Barry Brumitt, John Krumm, Brian Meyers, and Steven Shafer. "Ubiquitous computing and the role of geometry." Personal Communications, IEEE 7.5 (2000), 41-43.
22. Apple AVFoundation Framework Reference. https://developer.apple.com/library/mac/#documentation/AVFoundation/Reference/AVFoundationFramework/_index.html, April 2013.

23. Apple CoreVideo Framework Reference. http://developer.apple.com/library/ios/#documentation/CoreVideo/Reference/CVFrameworkRef/_index.html, April 2013.
24. Apple CoreMedia Frmaework Reference. https://developer.apple.com/library/mac/#documentation/CoreMedia/Reference/CoreMediaFramework/_index.html, April 2013.
25. CocoaAsyncSocket libraries. https://github.com/robbiehanson/CocoaAsyncSocket, April 2013.
26. Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24.6 (1981): 381-395.