Surround-See: Enabling Peripheral Vision on Smartphones during Active Use

Xing-Dong Yang[2], Khalad Hasan[1], Neil Bruce[1], Pourang Irani[1]

[1]University of Manioba,
Winnipeg, MB,Canada, R3T 2N2
{khalad, bruce, irani}@cs.umanitoba.ca

[2]University of Alberta,
Edmonton, AB, Canada, T6G 2E8
xingdong@cs.ualberta.ca

ABSTRACT

Mobile devices are endowed with significant sensing capabilities. However, their ability to 'see' their surroundings, during active use, is limited. We present Surround-See, a self-contained smartphone equipped with an omni-directional camera that enables peripheral vision around the device to augment daily mobile tasks. Surround-See provides mobile devices with a field-of-view collinear to the device screen. This capability facilitates novel mobile tasks such as, pointing at objects in the environment to interact with content, operating the mobile device at a physical distance and allowing the device to detect user activity, even when the user is not holding it. We describe Surround-See's architecture, and demonstrate applications that exploit peripheral 'seeing' capabilities during active use of a mobile device. Users confirm the value of embedding peripheral vision capabilities on mobile devices and offer insights for novel usage methods.

Author Keywords

Peripheral mobile vision, mobile 'seeing', mobile surround vision.

ACM Classification Keywords

H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

INTRODUCTION

Smartphones are equipped with powerful sensors, such as accelerometers, GPS and cameras that facilitate a variety of daily tasks. On commercial and research platforms, such sensors have been utilized in numerous contexts such as for distinguishing user activity [26], for sensing on, behind and around a mobile device [5, 41], for context awareness [46] and for interactive document exploration [13]. The integration of an ever expanding suite of embedded sensors is a key driver in making mobile devices smarter [26]. However, current capabilities are mostly focused on *sensing*. We distinguish 'sensing' from 'seeing' in that the latter facilitates some higher level of recognition or interpretation of objects, people and places in the mobile device's surroundings. What new applications might be possible if mobile devices had advanced *seeing* abilities?

UIST '13, October 8–11, 2013, St. Andrews, UK.

We explore the above theme of empowering mobile devices with enhanced peripheral vision capabilities. Our prototype, Surround-See, consists of a smartphone fitted with an omnidirectional lens that gives the device peripheral vision, of its surroundings (Figure 1). During active use, Surround-See effectively extends the smartphone's limited field-of-sight provided by its front- and back-facing cameras. With an ability to 'see' the rich context of the region around the device, smartphones can trigger environment specific reminders and can respond to peripheral interactions, such as pointing at a smart-appliance for efficient access to its control panel on the mobile device.

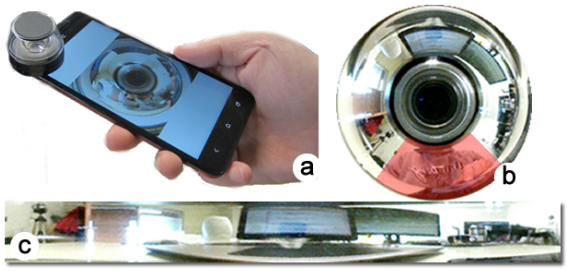

Figure 1 - (a) Surround-See enables peripheral 'sight' on smartphones by means of an omni-directional mirror attached to the mobile device's front facing camera. (b) Surround-See image shows the corresponding scene. (c) The unwrapped image can be used for recognizing the device's peripheral environment (after removing the user's body – shaded in red)

The scenario below captures some of the rich applications Surround-See enables. John, a professional is often using his mobile device. In the morning, he reads the news on Surround-See while his car engine warms-up. Recognizing this, Surround-See triggers a reminder on the danger of eyes-busy mobile use and driving. Later, as he settles into his office while checking email on Surround-See, he points at the speakers in the office, which Surround-See recognizes and provides a control panel to increase the speakers' volume. Laura, a colleague enters his office and asks about his weekend. John picks up his smartphone to show Laura pictures on the phone which Surround-See reorients as the device is positioned closer to Laura. Finally, Laura asks John for directions to the restaurant, which John draws using a stylus and his finger to erase, both of which are recognized as distinct by Surround-See. Shortly after doing some work he decides to step out for only a few

minutes. Surround-See recognizes this activity and asks whether he wishes to take his device. As he is uninterested, he declines by gesturing to the device remotely and Surround See sets the mobile into voicemail mode.

The above scenario captures the various possibilities made available when augmenting a mobile device with peripheral vision. During active use Surround-See can (a) trigger reminders based on an environment it recognizes, (b) it can perceive specific objects in the mobile device's periphery (i.e. speakers), which the user can control by (c) pointing at them. In idle mode, such as when resting on a table, Surround-See can also identify certain activities in the vicinity of the device, such as (d) when the user walks away from it or (e) if the user is remotely waving at it to alter its state, i.e. setting it to voicemail mode.

Our contributions include: (a) the concept of enabling mobile devices to 'see' their surroundings; (b) a demonstration of its value through an implementation of a self-contained proof-of-concept prototype, Surround-See; and (c) a set of applications that demonstrate Surround-See's unique capabilities over current smartphone usages.

RELATED WORK

Mobile 'sensing'

Researchers have investigated the possibility of embedding sensors on mobile devices to facilitate tasks in the background. An extensive review on mobile sensing is available in [36]. We briefly discuss state and context awareness enabled by mobile sensors. Hinckley et al.'s [25] seminal work on sensing techniques for mobile interaction opened a new wave of interfaces that exploit a mobile device's state, such as its orientation. Such capabilities, among others, have enabled a significant suite of interactions including the ability to control virtual objects on a mobile screen [47], for automatically reorienting images [25], to support navigation in applications [15], or to facilitate different gesture based interactions [28]. Many of the sensors proposed in these earlier systems are now common on commercial mobile devices, and have been used to create a wide ecosystem of applications that depend on sensing a device's state.

Researchers have also demonstrated methods to sense a mobile device's environment to alter the state of the device. Schmidt et al. [46] augmented mobile devices with tilt, light and heat sensors to identify if the device is resting on a table, is in a pocket, or is being used outside. Such sensor-based information can be used for automatically changing the device state such as lowering its volume. Context information is also possible with geo-locating sensors that incorporate information about the user's location to provide more relevant and targeted services [15, 34].

Beyond sensing a device state or context, researchers have introduced novel sensor-based interaction techniques to facilitate input outside the device's physical space, in its periphery. Single finger interactions around a device are possible with SideSight [5] and HoverFlow [35] which use IR distance sensors to capture gesture-based interactions above and to the sides of the device. Similarly, Abracadabra [21] facilitates input by activating states of a magnetic sensor and maps these to menu and cursor control. This results in occlusion-free input on small devices such as watches. For the most part, sensor-based interaction metaphors are confined to a specific suite of tasks.

Camera-based mobile interaction

On mobile devices, vision-based systems have advanced to the point of complementing basic sensing mechanisms in real-time. Self-contained mobile-based augmented reality (AR) applications have flourished [50]. Examples include Wagner et al.'s [50] marker-based AR system, Paelke et al.'s [41] hand-held AR soccer game and Hansen et al.'s [20] camera-equipped mobile device to establish a spatial relationship between a virtual environment and the physical space to form a mixed reality space.

Camera based input also been shown to work for 2D [51] and 3D navigation [19], for tracking the user's face for zooming/scrolling documents [48], for detecting where users are standing and reorienting the screen accordingly as the device is placed closer to them [8, 9] or for generating input events (e.g., click, double-click) in real-time [31].

In general, most camera-based interactions require that interactive features be used in exclusion of other tasks, as the user has to explicitly point the front- or back-facing camera at the object of interest. Ideally, vision-based methods on mobiles can include the devices' periphery which contains rich context information. This can open new possibilities for integrating users' interactions naturally into the ecosystem of daily mobile applications facilitating a broader range of implicit interactions that build on Buxton's vision for foreground/background interaction [6].

Peripheral vision

Researchers have explored methods for extending a camera's limited field-of-view without significantly changing its form factor. For example, omni-directional cameras have been applied to robotics problems to give robots 'sight' of their periphery. Applications include estimating a robot's motion [16], revealing a robot's location [10], recognizing its surroundings [45], and navigating around obstacles [33]. Very little work has considered the applications of embedding peripheral vision on smartphones for advanced interactions.

Walking User Interfaces

There is a growing trend toward building interfaces to support walking user interfaces (WUIs) [17], including methods to improve text-entry accuracy to user safety. Surround-See, is particularly of value for WUIs, as it allows users to perform additional actions while using the device.

HARDWARE DESIGN OPTIONS

To enable a smartphone with peripheral vision capabilities during active use led us to explore different hardware

options. We frame our exploration in terms of the level of 'sight' enabled by various sensing technologies, from coarse to fine. Coarse 'seeing' can detect the presence of an object, with limited ability to detect its motion. Finer 'seeing' involves recognizing different objects and the ability to detect changes in the smartphone's surroundings.

Sensor Options

We first considered sensors that can fit on current smartphones. A second requirement was to implement and explore use-cases with a self-contained prototype.

Proximity sensor

Proximity sensors (capacitive, infra-red, or ultrasound) detect the presence and distance of objects away from it (giving them coarse 'sight'). Multiple proximity sensors in a sensor array can detect the motion of an object in a 2D or 3D space [5]. However, such sensors cannot distinguish between different objects or detect the changes in their surrounding environment. The placement of proximity sensors is also challenged by how users hold the device without occluding them.

Magnetic Sensor

Magnetic sensors detect the presence and angular location of a magnet, such as on a ring, in an emitted field (giving them coarse 'sight'). A magnetic sensor array can detect the 2D or near surface 3D motion of an object [38], but like proximity sensors, they do not distinguish specific objects or content changes in the environment.

Image Sensor

Image sensors (CCD or CMOS) are standard on smartphones. They sense an optical image, which can be processed using computer vision techniques to detect the presence of an object in the image, the motion of an object in either 2D or 3D (with a 3D depth sensing camera), or recognize different objects in the surrounding environment.

Sensing Range

Sensors' limited range can be alleviated by forming an array of multiple sensors (e.g. proximity sensor array or camera array). The increased field-of-operation however, comes with a size tradeoff which makes such solutions impractical for mobile devices. Other more practical methods for enlarging the field-of-view consists of using a wide-angle or omni-directional lens, which can capture a good portion of the 360° around a device's periphery.

Sensor Installation and Form Factor Implications

Proximity or magnetic sensor arrays can be placed above or under an open surface on a smartphone [5, 38]. This allows them to sense the entire space around the smartphone without significantly impacting the device's form factor. However, during active use, these become unusable as the user's hand occludes these sensors.

While the built-in smartphone cameras can be used for the purpose of 'seeing', during active use, the front and back cameras face the sky and ground, respectively. The narrow field-of-view of such built-in cameras (43˚-56˚ on the high-definition smartphone camera we used), does not allow these to capture the smartphone's surrounding space (Figure 2). This limitation can be addressed by mounting a wide-angle or omni-directional lens on the built-in cameras.

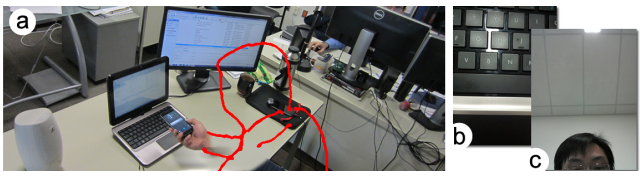

Figure 2 – (a) the environment where Figure 1b was taken from; (b)-(c) images taken from the phone's back and front facing cameras respectively.

SURROUND SEE HARDWARE

To explore the interaction space for peripheral vision on a smartphone we create a self-contained proof-of-concept system, Surround-See. The prototype is a HTC Butterfly smartphone with an omni-directional lens from Kogeto [1] mounted on its front-facing camera (Figure 1a). The front was chosen as placing the lens on the back may make the phone unstable at rest, thus precluding a significant number of interactions.

The smartphone runs the Android operating system on a Quad-core 1.5 GHz Krait CPU with 2GB of RAM. The front camera has a maximum resolution of 2MP. For real-time image processing, we down-sampled the resolution to 960×720. The omni-directional lens has a 360˚ and 56˚ field-of-view in the horizontal and vertical planes, respectively. The final prototype captures a real time RGB image of the 360˚ surrounding view of the device. Figure1b-c shows an omni-directional image captured by the prototype, and its unwrapped counterpart.

SURROUND-SEE CAPABILITIES

We implemented three primary capabilities with Surround-See: 1) to recognize the device's peripheral environment; 2) to recognize objects around the device; and 3) to recognize user activities in vicinity to the device. We implemented these features using OpenCV4Android, a JAVA wrapper that allows the OpenCV library to be used on Android platforms. It is worth mentioning that there are many choices for computer vision algorithms for implementing Surround-See features. We used and present those that have shown effective results in the literature and that can be implemented on a self-contained mobile device prototype.

Recognizing the Device's Peripheral Environment

Recognizing the users' peripheral environment was implemented using Local Binary Patterns (LBP) [40] and a machine learning classifier [7]. Before the recognition process starts, we pre-processed the raw omni-directional image by un-wrapping it to a panoramic image using the method described in [11]. While not technically necessary, unwrapping the image makes the rest of the processing easier. Once unwrapped, the bottom of the panoramic image was cropped so that it did not contain the body of the user (about 1/4 of the panoramic image). The resulting

image from this pre-process contains only a wide view of Surround-See's surrounding environment (Figure 1).

We then used LBP to describe the image using a unique feature vector. LBP detects microstructures inside an image (e.g. lines, edges). The histogram of the microstructures forms a feature vector of the image. LBP is orientation and luminance invariant, making it robust in describing images taken from different angles and different lighting conditions. The algorithm was originally proposed to classify textures (e.g. cloth). It has, however, been shown to be effective in detecting landmarks too [23]. The feature vector was used to train a machine learning classifier or to recognize a peripheral environment. We collected 20 samples for each of the 5 peripheral environments we recognized (lab, office, desk, hallway, and car). We took ten images for each, the morning and afternoon, with the phone in active use position. To train the classifier, we used Chang and Lin's LIBSVM library using the Support Vector Machine (SVM) [7]. We used a RBF Kernel with parameters that gave the highest 5-fold cross-validation scores (e.g. 96%). The trained model was loaded when the system starts. For every 60 seconds, the system sampled an image of the peripheral environment for recognition. The recognition process runs in a background thread, causing no interference to the phone's normal activities.

In contrast to a phone's built-in GPS, Surround-See can detect subtle changes in its location, e.g. movement inside or outside of a room or at a specific location in a room. When combined with GPS data, Surround-See can detect contextual changes happening within the GPS-sensed location, e.g. a crowed street vs. an empty street.

Recognizing Peripheral Items

Our implementation focuses on two general types of peripheral items: surrounding items and the user's hand.

Recognizing an Object in the Smartphone's Environment

An object in the device's environment is recognized using feature point matching using the ORB algorithm [44]. ORB searches each input frame for a desired object using a reference image. The reference image contains only the object to be searched for (see Figure 3a-b). When the system starts, ORB extracts a set of feature points for the reference image. A feature vector (or descriptor) was then generated for every feature point to uniquely describe the characteristics of that feature point. ORB was then used to search for similar feature points in the input frames. If an object in the input frame contained at least a certain number of matching points (e.g. 1/8 of the total matching set), it was recognized as the desired object. We used OpenCV's *FeatureDetector* class for feature point detection, and *DescriptorMatcher* class feature point matching.

ORB is color invariant. It also performs well with objects at different scales. However, we found it to be error prone to changes of the device's orientation. It could fail to identify the object if it appears at a different orientation in the input frame than that in the reference image. This would require the user to hold the device at the same orientation as when the reference image was sampled, which is impractical. We resolved this using the device's built-in compass. When the system starts, Surround-See loads a list of reference images and their corresponding phone orientations when the images were taken. These initial orientations were used to rotate the input frames during the recognition process. In our implementation, we used one sample image for each desired object while multiple samples per object at different orientations is also possible. Our process however does not incur any performance overhead.

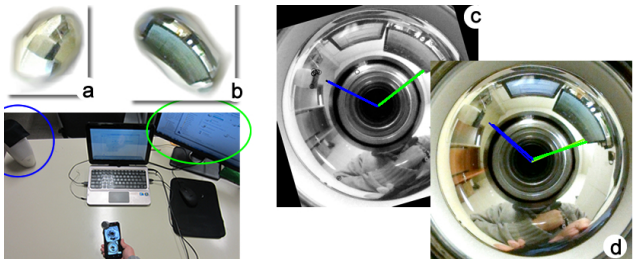

Figure 3 - Reference images: (a) speaker and (b) monitor; (c) rotated input frame for more precise object detection; (d) recognized objects (the blue and green lines indicate the locations of the recognized objects – speaker and monitor).

Recognizing the User's Hand

Explicit mid-air user input is possible, in Surround-See's periphery. Surround-See detects the user's hand using a skin color model in YCbCr color space [30, 42]. A skin color pixel was detected if its Cr and Cb values fall into the ranges [140, 166] and [135, 180] respectively. The resulting input frame formed a black&white binary image with its white region indicating the skin color pixels Figure 4 top).

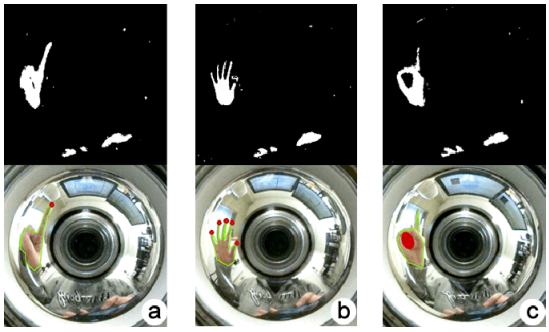

Figure 4– Top: binary image of skin-color pixels; Bottom: (a-b) track fingertip (red dot); (c) detect pinch (red ellipse).

The user's hand was detected by looking for blobs that are larger than a threshold size. We found this method to be effective but also error prone when the background contains colors close to that of the user's skin. We thus dynamically filtered out the background noise by removing the blobs that appeared in the same location for a certain fixed number of frames (e.g. 30 in our application). Finally, the hand contour was obtained by approximating a polygon of the detected hand blob using OpenCV's *approxPolyDP* function (Figure 4 bottom).

Tracking fingertips: Upon extracting the user's hand contour, the user's fingertips were detected by searching through the contour points, and identifying those with a curvature less than a threshold value (e.g. 50˚) [3] (Figure4 a-b) In an omni-directional image, the position of the detected fingertip indicates the position of the real finger around the camera (Figure 5). Surround-See can also detect the finger's up-and-down (vertical) motion. This is achieved by calculating the distance from the detected fingertip to the center of the omni-directional image, where an increase in the distance indicates that the finger is moving upward, and a decrease means the opposite.

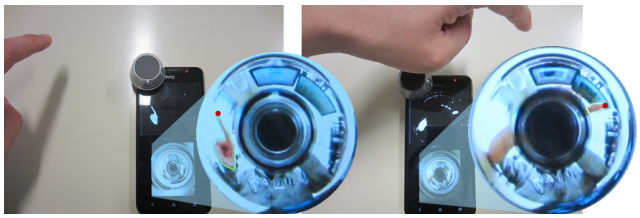

Figure 5- tracking finger position in the device's periphery.

Recognizing hand postures: A hand posture is recognized by counting the number of detected fingertips. This method is simple and fast. It allows 6 different hand postures in total – fist and 1 to 5 fingers. Other methods are also possible in cases where more hand postures are needed [12, 32, 49, 52]. In our implementation, the '1' posture, with the index finger is reserved for pointing.

Detecting 'pinch'-ing: Pinch is detected using Wilson's method [53], where a pinch is recognized when there is a connected blob inside a hand contour (Figure 4c) Pinch can be used as a 'mouse click' to confirm an action or to trigger a command. Once a pinch is detected, the hand posture recognition method is disabled until the pinch is released.

Detecting User Activities in the Periphery

We implemented three different peripheral activity detection capabilities: whether the user is moving the device away from them, whether the user is stepping away from the device and remotely gesturing at the device.

Proximity to User

During active use, Surround-See uses the user's upper body as a reference point to determine its proximity to the user. We demarcated a rectangular region of interest containing only the user's body to detect the smartphone's perpendicular motion relative to the user (Figure 6 left). Motion detection was implemented using optical flow [14], where the spreading of the motion vectors indicated that Surround-See was being placed closer (Figure 6b) to the user and the gathering of the motion vectors indicated that Surround-See was being moved away from the user (Figure 6a). We found this method reliable especially in differentiating between the perpendicular motion and the other motions such as moving left or right. Unlike previously proposed methods (e.g. [4]), Surround-See is self-contained. It does not need overhead cameras or require users to wear sensors on their body. This is extremely important for mobility.

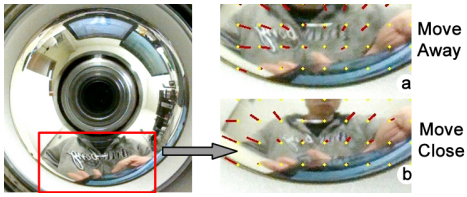

Figure 6 – Left: ROI contains user's body; Right: (a)-(b) optical flow vectors showing the phone is moving away and closer respectively.

It is worth noting that sensing user proximity cannot occur by simply using the smartphone's built-in sensors (e.g. accelerometer or built-in cameras). For example, the accelerometer can tell the direction in which the smartphone accelerates but cannot tell whether or not the user is moving with the smartphone. Similarly, the smartphone's built-in front or back camera may detect its motion but cannot tell whether or not the user is moving at the same time, e.g. walking when holding the phone, as the user is mostly outside the view of these cameras.

Detecting User Activity within a Region of Interest

In idle mode, i.e. when resting on a horizontal surface, Surround-See can detect the user's activity within a user-defined region of interest (ROI). Surround-See can dynamically track the user-defined ROI using the same method described in Figure 3. Once defined, the user's activities within the ROI were detected using optical flow. In our current implementation, Surround-See detects four user activities, including the user's movement in the horizontal and in the perpendicular directions (Figure 7). Additional activities, such as rapid or groups movements can be detected using sophisticated methods (e.g. [39]).

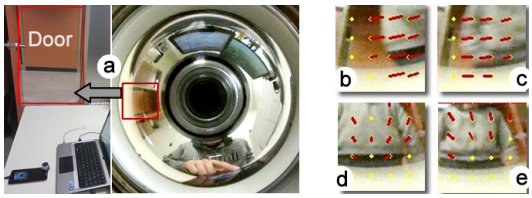

Figure 7 - ROI at the door (a); user moving from left to right (b) and right to left (c); user moving away (d) and closer (e).

Remote Gesturing

In idle mode, Surround-See also allows hand gestures to be carried out when the user is at a short distance away from the phone. This is convenient for situations when the phone is left behind on a desk and the user does not wish to walk up to grab it to invoke a simple command (e.g. turn on the voice mail). Remote gesturing assumes the phone is sitting on a stable platform such as a table and that the view is uncluttered. This allows us to use background subtraction to remove any skin-color noise in the background. This is particularly helpful as a hand blob will appear much smaller in the camera view when the user is not holding the phone, making the blob-size based noise removal error prone.

Surround-See uses a real-time adaptive background subtraction based on a Gaussian Mixture Model [29] (implemented in the *BackgroundSubtractorMOG* class in OpenCV). The algorithm updates the background dynamically such that if a moving object in the foreground stalls for several frames, it will be classified as a part of the background. This creates an effective mechanism for determining the start and end of a hand gesture. For example, when standing still, the user is classified as a part of the background. When the user starts to wave his/her hand, the moving part of the user's body becomes the foreground. This indicates the start of a gesture After the user stops moving the hand, Surround-See observes no moving object and gradually classifies the user's body as background. If the user does not move the hand again, s/he will eventually be classified as the background in several frames. This indicates the end of the gesture.

To track the user's hand trajectory, Surround-See finds the user's hand in the foreground using the same skin-color model described earlier A hand trajectory is composed of the temporal and spatial displacement of the center of the detected hand blob. In our current implementation, Surround-See uses a simple gesture recognition algorithm, which identifies hand gestures based on the hand's moving direction, e.g. moving left, moving right, or waving (move left then right or vice versa). More sophisticated algorithms would increase the remote gesture vocabulary set but could also demand higher processing power [37, 54].

SURROUND-SEE INTERACTIONS

We have implemented a number of interaction techniques to demonstrate several key Surround-See features. Each technique serves as an example of one or more of Surround-See's capabilities. Many of the applications are novel while a few others show how previously proposed techniques can be implemented in a mobile and self-contained prototype.

Pen vs. Touch Input

Capacitive stylus is a valuable addition to the user's finger for handwriting or drawing on smartphones and tablets. However, smartphones' touchscreens cannot distinguish a user's touch from that of a capacitive stylus. This problem can easily be solved with Surround-See as it can recognize objects in the environment. Surround-See can easily distinguish the stylus from the finger when interacting with the touchscreen. We implemented a simple drawing application to demonstrate this unique capability. Users can use a stylus to draw on the touchscreen and use the finger to erase the drawing (Figure 8a). Tracking the pen was implemented using the color model similar to the one used for detecting the user's hand.

Off-screen Pointing

Accessing off-screen objects is often considered a tedious and time consuming task due to the repeated invocation of panning or scrolling operations [27]. Recent research has shown that such a task can be made more efficient by directly pointing in mid-air at the location of the object in the around-device space [22]. Limited work exists on identifying the most appropriate sensing methods to facilitate around-device pointing. We implemented a restaurant search application to demonstrate Surround-See off-screen pointing potential (*Tracking User Finger*). When a restaurant of interest is located outside the map view, users can acquire information about it by directly pointing at its off-screen location indicated by an arrow shown on the screen (Figure 8b). We developed two selection mechanisms, dwell and back-tap (tapping on the back of the phone, sensed by the built-in accelerometers). The user can then select the restaurant to trigger an action, e.g. to retrieve a discount coupon. The user can toggle between off-screen objects in the general direction pointed at by the user by moving the finger up or down vertically (another dimension). In comparison to techniques using infrared proximity sensors [5], Surround-See is capable of tracking continuous finger movement at the corners of the smartphone, which is difficult for an array of range sensors. Furthermore, unlike sensors placed on the side to achieve this task [5], Surround-See's range is not occluded by the user's grip.

Remote Operation

Current smartphones can only be used when the user is directly interacting and in contact with the phone (e.g. by touching the phone's touchscreen). It is, however, quite often that the user may want to operate the phone, even briefly, from a short distance. For example, in a meeting with clients, the user may leave the phone on the meeting table when giving a presentation at the podium. If the phone rings during the presentation, the user may want to be able to mute the phone without having to leave the podium. Surround-See allows the phone to be operated remotely (*Remote Gesturing*). The user can simply wave at the phone to mute it. This operation cannot be carried out with smartphones' existing front or back camera when the phone is in a natural idle position. In our implementation, we map the user's hand gestures to common functions, e.g. wave right to mute the phone, wave right-then-left to unmute it, and wave left to turn on the voice mail (Figure 8c).

Controlling Remote Objects (Physical Shortcut)

Objects recognized in Surround-See's periphery can be used to carry out contextual actions. We created a remote control application, which uses physical objects (e.g. speaker or monitor) as a handler to trigger their corresponding controller on the user's smartphone. Users can point at a speaker to open a volume controller window on Surround-See to remotely adjust the speaker's volume (Figure 8d) (*Tracking Finger* and *Recognizing Environmental Objects*). Users can also point at a monitor to remotely turn it on or off. Here the surrounding objects serve as 'physical shortcuts' for launching applications on Surround-See. Users can also create paper stickers as disposable shortcuts [55]. The mapping between the commands and the physical objects relies on the semantics of the physical objects (intrinsic mapping [55]). This makes

learning shortcuts easy, which is often time-consuming especially when there is a large number of them [18].

Posture for Speed-dialing

Hand postures can be used as an easy and intuitive method to rapidly trigger a command on the smartphone. In our implementation, we used hand postures to trigger speed-dialing on Surround-See (*Recognizing hand postures*) (Figure 8e). We mapped 5 phone numbers to the 5 hand postures (from 1 to 5). To avoid unintentionally making calls, we allow the users to enable or disable *Posture Speed-dial* based on their needs. Unlike the other applications we describe here, hand posture is not exclusive to Surround-See, and can be carried out using the phone's front or back facing cameras. Surround-See provides an alternative, allowing postures to be used when the hand is already in the peripheral space.

Location-based Messaging

When in active use, Surround-See can perform contextual actions based on its location. We implemented a location-based messaging application, which displays a reminder or warning message on the screen (*Recognizing Peripheral Environment*). For example, when the application first recognizes that Surround-See is by the user's office desk, it asks whether the user wishes to "*Sync your phone?*" as a reminder. When it first recognizes the phone is being used in a shared space, such as a lab, the application asks whether to "*Mute your phone?"*. Finally, it warns the user to stop using the phone by showing "*Don't use your phone when driving*" on the screen when it recognizes the user is behind the wheel (Figure 8f). Such reminders can be included for safe utilization of the phone while walking and texting, for example [24]. Note that location detection based on 'sight' extends previous approaches using a proximity sensor [5], i.e. the system can distinguish car-A from car-B.

Proximity-based Screen Rotation

Showing others the content of the screen of a smartphone can sometimes be cumbersome because the user needs to reorient the phone to fit the viewer's field-of-view. The existing approach reorients the content when the phone is tilted. This method is error prone as it does not distinguish between tilt towards and away from the viewer. It is, however, natural for the user to stretch their arm to place the phone closer to the viewer so that the content on the screen can be clearly visible. Based on this observation, we created an image browsing application, which can automatically rotate the orientation of the image by 180˚ when it detects the phone is being moved away from the user (*Proximity to User*) (Figure 8g). It can also rotate the image back to its initial orientation when it detects the phone is being moved back to the user.

Notify to Take the Phone

Occasionally, users may forget to take their cell phone when leaving their home or office. We created a notification application to notify the user when this happens. When Surround-See is idle, e.g. sits on a desk (*Recognize Peripheral Environment*), the application is on. It monitors users' activities around the door of the user's office (*Detecting User Activity within a User-defined Region of Interest*), by detecting the motion of the moving object within the door region. Upon detecting that the user is moving out of the door (implemented using the same optical flow algorithm as described in *Detecting Proximity to User*), it plays a voice message "*Did you forget your phone?*" to notify the user (Figure 8h). The user may choose to go back to the desk to take the phone or make a hand gesture to turn on the voice mail (*Remote Gesture*).

ELICITING USER APPROVAL

We conducted a user survey as an initial step towards assessing users' approval of Surround-See as a concept that can co-exist with common smartphone usage. Our goal was to examine the value proposition of Surround-See's capabilities, our interaction techniques and users' privacy

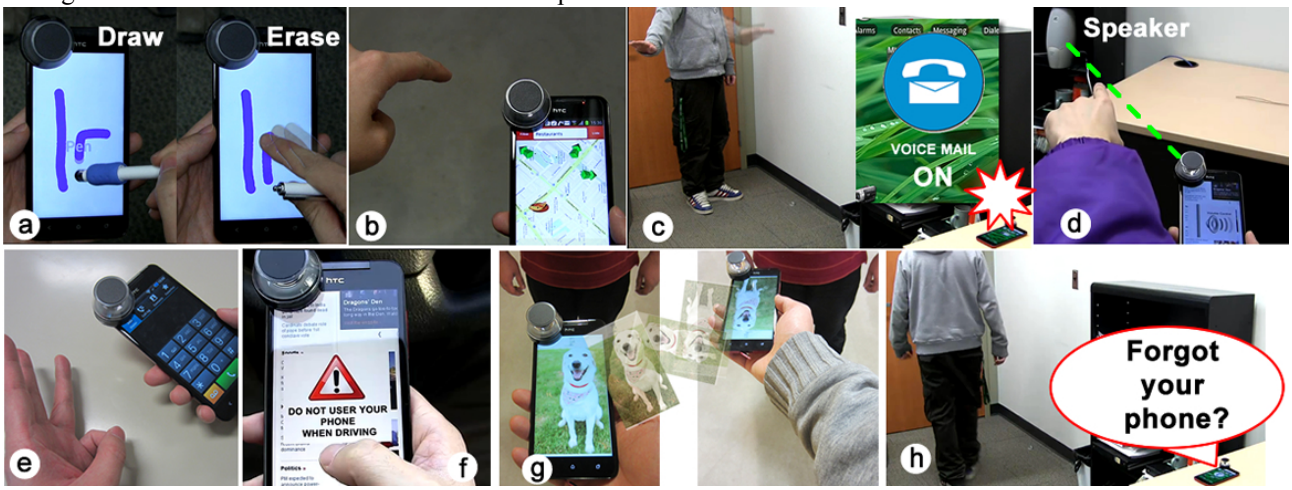

Figure 8 – (a) Left: use pen to draw; Right: use finger to erase; (b) Picking restaurant stored in the off-screen space; (c) Remote gesture to turn on voice mail; (d) Point at a speaker to open a volume controller window; (e) Hand posture for speed-dial; (f) Show a warning message when the user uses the phone behind the wheel; (g) Auto screen rotation based on the proximity to user; (h) Remind the user to take the phone when detects that the user is moving out of the door.

concerns. We adapted the feedback method introduced in [43] and participants made judgments by watching a video showing the Surround-See prototype (same video as the one included in this submission).

Participants

Seventeen computer science students (15 male, ages between 21 and 35) participated in our survey. All of them were smartphone users. Seven participants have used a smartphone for more than 3 years.

Procedure

The participants were shown a video presenting the prototype and all of Surround-See's capabilities. They were also shown the interaction techniques one at a time. For each interaction technique, they filled out a 7-point Likert scale questionnaire (1: strongly dislike and 7: strongly like), and gave reasons to justify their answers. After ranking the interaction techniques, the participants were asked to rank overall how useful they think the techniques are. Finally, they ranked their level of comfort about smartphones that had 'seeing' abilities and held by others, such as family, friends or strangers.

Overall, participants welcomed the idea of making the smartphone more 'sight' enabled during active use. They mostly like the intelligent features (*Notify to Take the Phone* and *Location-based Messaging*) that could help them with common daily slips such as forgetting to take the phone, to mute it in a classroom and features to support remote operations (*Controlling Remote Objects and Remote Operation*). These 4 features were ranked amongst the highest with an average score higher than 6. Three participants commented that they always forgot to mute their phone and another commented that *Location-based Messaging* is a useful feature because "*it could take control, when you forget to do something*". People like the convenient features that allow them to control objects at a distance and indicated that these should become standard on smartphones. A participant commented that *Remote Operation* is "*good because most of the time I leave the phone away and need to return briefly only to set it*". User reports suggest that even when users are not holding their smartphones they still wish to maintain a link with their devices, even at a distance.

Three features (*Proximity-based Screen Rotation*, *Posture for Speed-dialing*, and *Pen vs. Touch Input*) received weaker approval scores between 5 and 6. Most participants agreed that these are handy features to have on top of the phone's existing functions but they also felt these features are limited to a small set of applications. For example, one participant said *Pen vs. Touch Input* is "*useful for drawing apps on my phone. I'd like to see how else it could be used though*". Finally, participants gave a neutral score (4.7, s.e. 0.37) to *Off-screen Pointing*. Most participants did not see high value for this feature in their daily smartphone usage.

Overall, participants did not complain about privacy issues when other people use Surround-See. They felt most comfortable when Surround-See is used by people they know. The user's level of comfort decreases when Surround-See is used by people they know less. They feel neutral (4.3, s.e. 0.24; with 7 being strongly comfortable) when Surround-See is used by a stranger but also expressed a demand for feedback to show that Surround-See is turned on (5.3, s.e. 0.52) (Figure 9). Interestingly, participants wished to also receive feedback if family members had devices with peripheral vision (4.18, s.e. 0.5). This needs to be considered in the design of such devices.

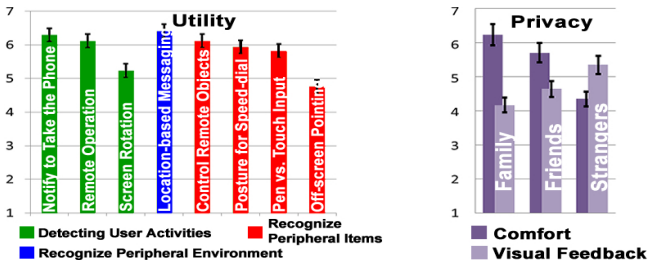

Figure 9 – Left: average user rating for Surround-See's interaction techniques; Right: average user ratings for potential privacy concerns.

DISCUSSION AND LIMITATIONS

In this section, we discuss the lessons we learned and insights we gained from our experience. We also present limitations of our approach.

Omni-directional lens: the omni-directional lens we used provides a 360˚ view of the peripheral space but presents pixel loss. First, the image from an omni-directional lens is distorted, especially towards the center of the concentric circles. During our implementation, we did not observe major issues caused by this distortion. However, the degree of distortion may vary from lens to lens. Calibration may be considered (e.g. checkerboard calibration) when implementing with different lenses. Second, the object seen from the omni-directional lens is smaller than what can be seen with a normal lens. Smaller objects have fewer pixels to describe their characteristics. This has made object recognition harder. These issues might be resolved by using wider angle omni-directional lenses.

Field-of-view: we believe Surround-See's capabilities can be further extended if its field-of-view went beyond the phone's peripheral space, ideally covering the entire 360˚ spherical space around the phone. With our current prototype, peripheral objects may not completely fall into the camera's view, an issue that can be addressed with different omni-directional mirror styles and capabilities.

A wider field-of-view allows the system to gain a better 'picture' of its surrounding environment. For example, when the phone is in active use, the user's face is mostly invisible. A complementary top view may allow Surround-See to run face detection on the missing pixels and check if it is the authorized user who is using the phone. This can

also allow the users to use the original function of the front facing camera, which we had to sacrifice in our current prototype. Equally important is the wide-angle view from the back camera. It complements what is seen from the front camera, and completes the knowledge of the phone's surrounding space. With more advanced image sensing technologies, we may see true 360˚ cameras such as [2] that could be mounted on mobile devices.

Depth sensing: Surround-See can also benefit from depth sensing. With knowledge about peripheral objects' distance, Surround-See can alert the user about incoming people or traffic not only in the front [24] but also from the side during eyes-busy interaction. A stereo omni-directional image may be obtained by using 2 sets of cameras and an omni-directional lens. This setup is mainly used on larger platforms, e.g. robotics. Further work is needed to explore this possibility.

User recognition: user recognition could be a useful addition to Surround-See capabilities. Recognizing who is using the phone can be helpful for increasing its security. Knowing who is in the periphery also allows richer interactions to be carried out, e.g. multi-user input. Future work will explore different ways to recognize users. It is worth noting that beyond a certain distance from the phone, complex pattern recognition tasks are challenging due to inadequate pixel resolution. The set of interactions possible will only increase with improved technology.

Computer vision: the performance of Surround-See relies on several factors, including the mobile devices' computing power, the quality of the camera (including lens), and the choice of computer vision algorithms. Given that smartphone cameras don't typically offer nearly the sensor sizes that appear in more traditional vision applications, and also have a small fixed aperture, one would expect that the robustness of most algorithms will suffer somewhat. On the other hand, the limitations of processing capabilities of mobile devices also places limits on the set of vision algorithms that can be used in the proposed applications. However, these issues will become less significant with advances in mobile imaging and processing capabilities.

Battery life: mobile devices' battery life is a concern in our implementation as batteries drain quickly when the camera is active. This issue can also be less problematic with newer ultra-low-power image sensing chips and improvements in battery technology.

Form factor: The form factor of the current prototype can be improved. The 'useful' lens in our off-the-shelf sensor is far smaller than its casing, which can be removed to better integrate the lens in a future device. The lens can be further engineered to hide inside the smartphone when the front or back facing cameras are needed for other tasks.

System evaluation: Surround-See warrants careful investigation of its interaction and recognition techniques. This will be helpful for understanding its practical usability across different environments and scenarios.

CONCLUSION

We introduced the concept of enabling mobile devices to 'see' their surroundings during active use. We created a proof-of-concept system, Surround-See, by mounting an omni-directional lens on the device's front facing camera. We explored Surround-See's capabilities, and implemented a number of interaction techniques to demonstrate its unique features. In an informal setting, users welcomed the idea of having smartphones with advanced 'seeing' abilities. Future work will focus on increasing Surround-See's field-of-view to its entire surroundings and enabling 3D depth sensing. These will include exploring hardware options and software applications that integrate seamlessly with daily mobile tasks.

REFERENCES

1. Kogeto http://www.kogeto.com/say-hello-to-dot
2. Ricoh's 360-degree camera http://www.ricoh.com/
3. Argyros, A. A. & Lourakis, M. I. A. Vision-Based Interpretation of Hand Gestures for Remote Control of a Computer Mouse. *ECCV'06*, 40-51.
4. Ballendat, T., Marquardt, N. &Greenberg, S. Proxemic Interaction: Designing for a Proximity and Orientation-Aware Environment. *ITS'10*. 10 pages.
5. Butler, A., Izadi, S. & Hodges, S. SideSight: multi-touch interaction around small devices. *UIST'08* 201-204.
6. Buxton, W. Chunking & Phrasing and the Design of Human-Computer Dialogues. *IFIP'86*. 475-480.
7. LIBSVM--A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.
8. Cheng, L.-P., Hsiao, F.-I., Liu, Y.-T. & Chen, M. Y. iRotate: automatic screen rotation based on face orientation. *CHI'12*. 2203-2210.
9. Dearman, D., Guy, R. & Truong, K. Determining the orientation of proximate mobile devices using their back facing camera. *CHI'12*. 2231-2234.
10. Drocourt , C., Delahoche, L., Pegard, C. & Clerentin, A. Mobile robot localization based on an omnidirectional stereoscopic vision perception system. *ICRA'99*. 1329-1334.
11. El Kadmiri, O. An omnidirectional image unwrapping approach. *ICMCS'11*, 1 - 4.
12. Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D. & Twombly, X. Vision-based hand pose estimation: A review. *CVIU'07*, 108 (1-2), 52-73.
13. Eslambolchilar, P. & Murray-Smith, R. Tilt-Based automatic zooming and scaling in mobile Devices-A State-Space implementation. *LNCS'04*. 120-131.
14. Farneback, G. Two-frame motion estimation based on polynomial expansion. *SCIA'03*. 363-370.
15. Gasimov, A., Magagna, F. & Sutanto, J. CAMB: context-aware mobile browser. *MUM'10*, 1-5.

16. Gluckman, J. & Nayer, S. K. Ego-motion and omnidirectional camera. *ICCV'98*. 999-1005.

17. Goel, M., Findlater, L. & Wobbrock, J. WalkType: using accelerometer data to accomodate situational impairments in mobile touch screen text entry. *CHI'12*. 2687-2696.

18. Grossman, T., Dragicevic, P. & Balakrishnan, R. Strategies for accelerating on-line learning of hotkeys. *CHI'07*, 1591-1600.

19. Hachet, M., Pouderoux, J. & Guitton, P. A camera-based interface for interaction with mobile handheld computers. *I3D'05*. 65-72.

20. Hansen, T. R., Eriksson, E. & Lykke-Olesen, A. Mixed interaction space: designing for camera based interaction with mobile devices. *CHIEA'05*. 1933-1936.

21. Harrison, C. & Hudson, S. E. Abracadabra: wireless, high-precision, and unpowered finger input for very small mobile devices. *UIST'09*. 121-124.

22. Hasan, K., Ahlström, D. & Irani, P. AD-Binning: Leveraging Around Device Space for Storing, Browsing and Retrieving Mobile Device Content. *CHI'13*. 899-908.

23. Heikkilä, M., Pietikäinen, M. & Schmid, C. Description of interest regions with local binary patterns. *Pattern Recognition*, 42 (3), 425-436.

24. Hincapié, J. D. & Irani, P. CrashAlert: Enhancing Peripheral Alertness for Eyes-Busy Mobile Interaction while Walking. *CHI'13*. 3385-3388.

25. Hinckley, K., Pierce, J., Sinclair, M. & Horvitz, E. Sensing techniques for mobile interaction. *UIST'00*. 91-100.

26. Hinckley, K. & Sinclair, M. Touch-sensing input devices. *CHI'99*. 223-230.

27. Irani, P., Gutwin, C. & Yang., X.-D. Improving selection of off-screen targets with hopping *CHI'06* 299-308.

28. Jones, E., Alexander, J., Andreou, A., Irani, P. & Subramanian, S. GesText: accelerometer-based gestural text-entry systems. *CHI'10*. 2173-2182.

29. KaewTraKulPong, P. & Bowden, R. An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection. *AVBS'01*.

30. Kakumanu, P., Makrogiannis, S. & Bourbakis, N A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40 (3), 1106-1122, 2007.

31. Kang, Y. & Han, S. Improvement of smartphone interface using an AR marker. *VRCAI'12*. 13-16.

32. Khan, R. Z. & Ibraheem, N. A. Survey on Gesture Recognition for Hand Image Postures. *CIS*, 5(3), 110-121.

33. Kim, J. & Suga, Y. An Omnidirectional Vision-Based Moving Obstacle Detection in Mobile Robot. *IJCAS'07*, 5 (6), 663-673.

34. Kjeldskov, J. & Paay, J. Just-for-us: a context-aware mobile information system facilitating sociality. *MobileHCI'05*. 23-30.

35. Kratz, S. & Rohs, M. Hoverflow: exploring around-device interaction with IR distance sensors. *MobileHCI'09*, 1-4.

36. Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T. & Campbell, A. T. A survey of mobile phone sensing. *CM*, 48 (9), 140-150.

37. Li, Y. (2010). Protractor: A Fast and Accurate Gesture Recognizer. *CHI'10*. 2169-2172, 2010.

38. Liang, R.-H., Cheng, K.-Y., Su, C.-H., Weng, C.-T., Chen, B.-Y. & Yang, D.-N.. GaussSense: Attachable Stylus Sensing Using Magnetic Sensor Grid. *UIST'12*. 319--326.

39. Masoud, O. & Papanikolopoulos, N. A method for human action recognition. *IVC'03*, 21 (8), 729-743.

40. Ojala, T., Pietikäinen, M. & Mäenpää, T. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *PAMI'02*, 24 (7), 971-987.

41. Paelke, V., Reimann, C. & Stichling, D. Foot-based mobile interaction with games. *ACE'04*. 321-324.

42. Ramakers, R., Vanacken, D., Luyten, K., Coninx, K. & Schöning, J. Carpus: a non-intrusive user identification technique for interactive surfaces. *UIST'12*, 35-44.

43. Rico, J. & Brewster, S. Usable Gestures for Mobile Interfaces: Evaluating Social Acceptability. *CHI'10*. 887-896.

44. Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. ORB: an efficient alternative to SIFT or SURF. *ICCV'11*. 2564-2571.

45. Runschotem, R. & Krose, B. Robust scene reconstruction from an omnidirectional vision system. *ToRA'03*, 19 (2), 351-357.

46. Schilit, B., Adams, N. & Want, R.. Context-Aware Computing Applications. *MCSA'94*.

47. Seo, B.-K., Choi, J., Han, J.-H., Park, H. & Park, J.-I. (2008). One-handed interaction with augmented virtual objects on mobile devices. *VRCAI'08*. 1-6.

48. Sohn, M. & Lee, G. ISeeU: camera-based user interface for a handheld computer. *MobileHCI'05*, 299-302.

49. Stergiopoulou, E. & Papamarkos, N. Hand gesture recognition using a neural network shape fitting technique. *EAAI'09*, 22 (8), 1141-1158

50. Wagner, D. & Schmalstieg, D. First steps towards handheld augmented realit. *ISWC'03*. 127 - 13.

51. Wang, J. & Canny, J. (2006). TinyMotion: camera phone based interaction methods. *CHIEA* '06, 339-344.

52. Wang, R. Y., Paris, S. & Popovic:, J. (2011). 6D hands: markerless hand-tracking for computer aided design. *UIST'11*. 549-558.

53. Wilson, A. Robust Vision-Based Detection of Pinching for One and Two-Handed Input. *UIST'06*. 255-258.

54. Wobbrock, J., Wilson, A. & Li, Y. Gestures without libraries, toolkits or Training: a $1.00 Recognizer for User Interface Prototypes. *UIST'07*. 159-168.

55. Yang, X.-D., Grossman, T., Wigdor, D. & Fitzmaurice, G. Magic Finger: Always-Available Input through Finger Instrumentation. *UIST'12*. 147–156.