ViRi: View it Right

Pan Hu, Guobin Shen, Liqun Li, Donghuan Lu
Microsoft Research Asia, Beijing, 100080, China
{v-pah, jackysh, liqul, v-donlu}@microsoft.com

ABSTRACT

We present ViRi – an intriguing system that enables a user to enjoy a frontal view experience even when the user is actually at a slanted viewing angle. ViRi tries to restore the front-view effect by enhancing the normal content rendering process with an additional geometry correction stage. The necessary prerequisite is effectively and accurately estimating the actual viewing angle under natural viewing situations and under the constraints of the device's computational power and limited battery deposit.

We tackle the problem with face detection and augment the phone camera with a fisheye lens to expand its field of view so that the device can recognize its user even the phone is placed casually. We propose effective pre-processing techniques to ensure the applicability of face detection tools onto highly distorted fisheye images. To save energy, we leverage information from system states, employ multiple low power sensors to rule out unlikely viewing situations, and aggressively seek additional opportunities to maximally skip the face detection. For situations in which face detection is unavoidable, we design efficient prediction techniques to further speed up the face detection. The effectiveness of the proposed techniques have been confirmed through thorough evaluations. We have also built a straw man application to allow users to experience the intriguing effects of ViRi.

Categories and Subject Descriptors

C.3.3 [**Special-Purpose and Application-based Systems**]: Signal processing systems; I.4.0 [**Computing Methodologies**]: General Image Displays

General Terms

Algorithm, Design, Experimentation, Performance

Keywords

Fisheye camera; fisheye image; viewing angle estimation; perspective distortion correction; face detection; device-user awareness; light sensor; continuous sensing

MobiSys'13, June 25-28, 2013, Taipei, Taiwan

(1) Slanted View (2) ViRi Effect

Figure 1: Our fisheye camera prototype on a Galaxy Nexus smartphone, and a comparison of the various effects at the 60° slanted viewing angle.

1. INTRODUCTION

A human vision system ensures the best visual quality when viewing from a frontal view, i.e., facing squarely to the display content. People make every effort (mostly subconscious) to have a frontal view, for example by adjusting the screen or properly holding the device. In real life, however, there are plenty of situations where we end up viewing a device's screen from a slanted viewing angle. For example, a person may put a mobile phone on the desk next to the keyboard while working on a PC. When there are prompts such as reminders or push notifications (e.g., SMS, emails, etc.) on the phone screen, the user will be viewing them at a slanted angle; a tablet user may not always hold the device for viewing. She may instead rest the device on a table and continue reading in a more relaxed posture. Additionally, the fact that people desire displays with large viewing angles is also evidence of the high likelihood of viewing screen content at a slanted viewing angle.

Visual content is difficult to consume at a slanted viewing angle. Even though the user may actually see the pixels of the content clearly, recognition is difficult due to the perspective distortion and the smaller effective viewing area. In the above example, the user may have to pick up the phone to read the prompt, which leads to an unnecessary

interruption to ongoing work, especially when the prompt is of little or no importance (e.g., spamming SMS/emails, etc); a tablet user may have to accept lower viewing quality or change to a less relaxed posture, or as many others do, purchase a stand for their tablet. The Microsoft Surface actually comes with a kickstand to ensure a better viewing experience.

In this paper, we present *ViRi*, an intriguing system that enables a mobile user to enjoy a frontal view experience in natural viewing situations. With ViRi, from no matter what angle the user is viewing the display, she will always see the content as if she were viewing the content squarely from the front. Figure 1 illustrates such an effect, as well as the original slanted view for comparison. In the above example, a ViRi mobile phone user would not be interrupted as she can clearly see the prompts even at a glance, and a ViRi tablet user can continue reading from a relaxed position without resorting to any stand to hold the device.

ViRi tries to restore the frontal view effect by augmenting the graphics rendering pipeline with a geometry correction module. The module counteracts the negative effects caused by the slanted viewing angle, i.e., the perspective distortion and reduced effective viewing area. Evidently, the necessary prerequisite is an effective and accurate estimate of the actual viewing angle under natural device usage. This is extremely difficult, as the natural usage implies that the device may be held or placed in a nearby stand, and that the user will not make any special signal to facilitate viewing angle detection. In addition, this has to be achieved under the constraints of the device's computational power and limited battery reserve.

Among all the sensors a mobile phone is equipped with, only the camera, with its remote sensing capability, might be feasible for our target scenario, i.e., sensing the user even when the device is placed away from the user. If we can detect the face and eye positions, we can estimate the viewing angle and make corrections accordingly. However, the current phone camera has a very limited field of view (FOV), and is thus not very helpful because when the camera can see the user, she is already at an almost frontal view.

We propose equipping mobile devices with a fisheye camera that features an extremely large FOV. In our prototype (see Figure 1), we augment the existing phone camera by adding a commercial off-the-shelf fisheye lens. However, using a fisheye lens creates new challenges. First of all, the images captured with a fisheye lens are severely distorted, suffering from both radial distortion and perspective distortion. Secondly, a large FOV can easily include light sources, which leads to harsh lighting conditions (e.g., strong backlight and underexposed faces) that typical face detection applications would carefully avoid. Thirdly, our target scenario requires quick adaptation of the display to the actual viewing angle. Hence ViRi needs to quickly detect the viewing angle (mainly the face detection). Unfortunately, face detection is of high computational complexity and usually takes a long time. This implies significant power consumption and would increase the tension caused by a limited battery charge.

We address all these challenges with ViRi. Instead of developing new face detection algorithms specifically for fisheye images, we properly pre-process the fisheye image and leverage existing face detection tools. In particular, we perform an offline camera calibration process to obtain the camera model, with which we can effectively rectify a fisheye image into a planar one to fix the radial distortion. We perform camera reorientation to fix the perspective distortion, and adopt an auto exposure algorithm to handle harsh lighting conditions.

To reduce energy consumption, we identify different viewing situations (Hand-Held and Off-the-Body) and leverage system states (i.e., screen on/off and if there are pending prompts), low power sensors (accelerometer, light sensor, and proximity sensor) to rule out impossible viewing situations and maximally skip face detection in real viewing situations. For situations in which face detection is indeed necessary, we have developed effective prediction methods to reduce the face detection time without compromising detection accuracy. More concretely, in Hand-Held viewing situations, we perform face detection only when the accelerometer indicates a new (quasi-)stationary state after significant device attitude changes. We have designed an angle-based prediction scheme that extracts the Euler angles between the attitudes before and after the change and predicts the most likely face position from that in the previous attitude. In Off-the-Body viewing situations, a large viewing angle change is usually associated with significant posture changes. We leverage the light sensor to detect such situations. Based on the fact that the device does not move while the user's head will move slightly (even if a person may not sense the movement, the device detects it), we have designed a motion-based prediction scheme that takes two pictures at a short interval (say 300 ms) and predicts the face position according to the differences between the two images using an effective integral-image based technique.

We performed thorough evaluations of each component technique. Our proposed preprocessing techniques boost the face detection ratio by 21%, and angle-based prediction and motion-based prediction achieve accuracy of 90% and 85% , respectively, in component evaluations where the test cases are more stressed. In system evaluation of natural phone usage, performance is even better. The average view angle detection error is usually within 10 degrees. We have also built a straw man application in which the user can provide a picture and experience the effect after viewing angle correction at arbitrary slanted viewing angles. A small-scale informal user study suggests that ViRi provides an appealing viewing experience, but also reveals new issues such as smart content selection (due to the reduced effective viewing area). Also, supporting a wide spectrum of applications requires seamless integration with the graphics pipeline of the operating system. We leave these issues for future study.

Contributions: In brief, we have made the following contributions in this paper:

- We identify the slanted viewing phenomena and advocate a consistent front-view experience; We study the feasibility of using a fisheye camera for perpetual user sensing;
- We design effective pre-processing techniques to ensure robust face detection with existing face detection tools, and design effective angle-based and motion-based prediction techniques to speed up the face detection task in natural viewing situations;
- We leverage system events and low power sensors to avoid unlikely viewing situations and further identify opportunities where face detection can be skipped. We conduct an in depth study of the light sensor, which has not been well explored before.

2. MOTIVATION AND THE PROBLEM

2.1 Motivation

When viewing a device from a frontal view, we face the display squarely (or at least the portion of the screen being viewed when the screen is large). This offers the best viewing quality because, humans have a small depression from 2.5 to 3 mm in diameter at the center of the retina known as the yellow spot or macula, which offers the best vision resolving power. In addition, our eyeballs and head naturally rotate to adapt the viewing angle so that the content being viewed will be imaged at the macula area. In fact, people strive (mostly subconsciously) to maintain a frontal view, by for example adjusting the screen or properly holding the device.

However, in real life, we often encounter situations where the front-view experience is hard to maintain all the time. We may be forced to look at the screen from a *slanted viewing angle*, as in the following common scenarios:

- People often put their phone beside the objects (e.g., laptop, book) they are using. When new information or prompts are received, such as an SMS or a pushed email, it is desirable to be able to peek at the content first before deciding to pick up the device and respond, as the action will interrupt their ongoing activity.
- People often read ebooks or watch videos on their devices such as a tablet. It is very hard to keep a single viewing posture for a long time, and thus highly desirable to continue reading comfortably in relaxed postures that may change from time to time.
- Most mobile devices can automatically switch between portrait and landscape display mode using the accelerometer. This works effectively when the user's view orientation is upright, but will mess things up if the user is watching horizontally such as while lying on a bed [7].
- People prefer large displays for higher productivity or better visual effect. However, the larger display is usually adjusted so that the center area corresponds to the eye height of the user. This ensures a perfect front-view for the center area, but leads to possible slanted viewing angles for side areas.
- People usually place the TV opposite to the sofa. But sometimes people may watch TV from a location other than the sofa, say the dinner table.

The fact that people unanimously require displays with large viewing angles also implies a high likelihood of viewing screen content at slanted viewing angles. Modern display technologies typically support a wide viewing angle. For example, IPS panels can offer a viewing angle up to 178 degrees. With a wide viewing angle display, and the proper orientation of the head and eyeballs, we are almost ensured of receiving clearly displayed content, even at very slanted viewing angles. Nevertheless, although it might be clear, the slanted viewing angle results in a distorted image and a reduced effective viewing area. This will cause an uncomfortable viewing experience and people might find the content is hard to interpret.

2.2 Problem Statement

Our goal is to improve the viewing experience at slanted viewing angles by compensating for distorted images and better utilizing the reduced effective viewing area. This will enable users to consume display content as if they were viewing from the front-view, regardless of a device's resting position. To this end, the necessary prerequisite is an accurate estimation of the user's viewing angle under natural phone usage.

While it is possible to ask the user to manually activate viewing angle detection, it would hurt the user experience, especially when both hands are occupied. Thus, it is more desirable to perform continuous sensing of the user's viewing angles. This is also a first step towards continuous, real-time, and in-situ detection of user attention. With ViRi, several other interesting scenarios can easily be enabled:

- Healthy posture monitoring: it could be desirable to monitor the working posture of users and alert them when they have stayed in the same posture for a long time to avoid problems such as neck pain or myopia. This is generally a difficult task for mobile phones without resorting to wearable peripherals.
- Forgotten device alarm: Alert the user to bring the device when leaving. This can be a highly desired feature, as forgetting a mobile device can cause a great deal of inconvenience and may raise the concern of privacy leaks.

Problem Statement: We hope to design an effective means for a mobile device to perform *continuous*, *real-time* and *in situ* detection if the user is looking at the device, and further estimate the actual viewing angle of the user without changing the user's habits. We focus on solving the problem for mobile devices because the viewing angle detection problem for a fixed display (e.g., a Desktop PC) is a special case. The solution needs to work in a number of real situations such as when the device is being held by the user or is resting nearby. It also needs to respect the constraints of the device's computational power and limited battery reserve.

3. SOLUTION IDEA AND CHALLENGES

3.1 Solution idea

Design Considerations: The *in situ* requirement implies that we cannot depend on an infrastructure aid for the sake of user mobility. The constraint of not altering a user's habits implies that we cannot ask the user to have any special gadgets (such as Google Glasses), nor make any special signal (e.g., manual activation) to facilitate the detection. Thus we have to completely rely on existing phone sensors. Considering the possibility that the device may be away from the body, only the microphone and camera may be viable options. While a microphone can sense the user via voice detection, it is not dependable as a user does not speak all the time, nor will voice detection be able to determine a user's viewing direction.

The only remaining possibility is to use the camera. If we can capture and detect a user's face and eyes, we can estimate the viewing angle. Camera have become pervasive on commodity smartphones. Advances in face detection and increasing availability of public face detection SDKs and web services make it practical to explore many camera-based face-aware applications. However, existing phone cameras usually have a rather limited field of view (FOV). Without careful placement, it is very likely the user's face will fall outside of the FOV. Figure 3.1 shows an image taken by the front camera of a Galaxy Nexus placed near the keyboard

(a) Normal View (b) Fisheye View

Figure 2: Fisheye lens significantly expands the FOV. Images were taken with the front camera on a Galaxy Nexus, with a COTS fisheye lens.

while the user is sitting in front of a PC. Obviously, the user is not captured in the image. The camera cannot be a viable solution unless we can expand its FOV.

Leverage Fisheye Lens: Fisheye lenses can achieve ultra wide FOV of 180-degrees or even wider. There are commodity fisheye lens accessories for mobile phones such as Olloclip [4]. We can concatenate a fisheye lens to an existing phone camera system to expand its FOV.[1] The effect is shown in Figure 3.1. The view covered by the original image is highlighted with the rectangle. As can be seen, the viewing angle is greatly increased and the user is now captured in the image. Quantitatively, the resulting FOV increases from the original 45 and 60 degrees (horizontal and vertical directions) to about 120 and 140 degrees, respectively, after putting on the fisheye lens. The difference in horizontal and vertical FOVs are due to the aspect ratio of the image sensor.

One special merit of fisheye lens is its very large depth of field – the distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image, due to its extremely short focal length. Essentially, the complete scene, near to far, is sharp. Fisheye lens can take sharp images even without focusing. This is a very favorable feature as face detection algorithms desire sharp face images.

3.2 Challenges

A fisheye lens greatly increases the FOV and provides the possibility of achieving our goal, but it also creates new challenges. The adoption of face detection also leads to new issues for resource-constrained mobile devices.

Face Detection in Highly Distorted Images: Fisheye lenses achieve ultra wide angles of view by forgoing straight perspective lines (rectilinear images), opting instead for special mapping (e.g., equisolid angle) to create a wide panoramic or hemispherical image. As a result, they produce highly distorted images with both radial distortion and perspective distortion. A fisheye image is more distorted in the peripheral areas than in the central area due to the view compression. This can be clearly observed from images in Figure 3.1, as well as Figure 3 that shows a set of fisheye

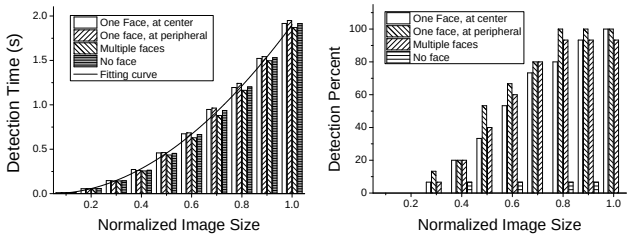
[1]Such a simple concatenation actually sacrifices the FOV of the fisheye lens. We envision future mobile phones may equip a genuine fisheye or ultra-wide FOV camera by design, as there are many other advantages in having such a system to be explored.

Figure 3: Example of fisheye images at different viewing angles and distances. Left to right: viewing angles are 60, 45, 30, 15 and 0 degrees. Top to bottom: viewing distances are 30, 50, and 70 cm.

images taken from different viewing angles and distances in a typical office environment. As a large portion of the scene is imaged in the peripheral areas of the resulting image, this leads to low pixel density for the objects that appear in the peripheral areas (see Figure 3) and becomes more obvious when rectified into planar pictures (see Figure 8). Figure 3 also presents cases in which only a partial face may be captured, especially at large viewing angles.

In actual daily usage, the mobile device usually faces upwards. Due to the extra wide FOV of the fisheye lens, it is very likely to include light sources in the view, especially in indoor environments where the lights are mostly on the ceiling. The phone camera adopts a central weighted average exposure measurement. It can easily be fooled and yields severely underexposed images.

Fast Detection Speed on The Device: Face detection algorithms commonly adopt a binary pattern-classification approach. The content of a given part of an image is first transformed into features by matching against a series of templates. Then a classifier trained on example faces decides whether that particular region of the image is a face or not. This process is repeated at all locations and scales using a sliding window [20]. Thus, face detection time is closely related to image size.



(a) Detection Time (b) Detection Accuracy

Figure 4: Detection time and corresponding accuracy at different image sizes. The size is normalized against the largest input, which is 1280×960.

We measured the face detection time at different input image sizes, where an input image may contain no, one, or

multiple faces and may appear in the center or peripheral areas, on a Galaxy Nexus using Android FaceSDK. Figure 3.2 shows the results, where the x-axis is the image size ratio normalized against the highest resolution 1280×960. From the figure, we can see that the detection time indeed increases with the image size, quadratically to the size ratio and linearly to the area ratio. The detection time seems content independent, as it varied little no matter whether there was one or more faces in the image or no face at all.

However, while leading to faster detection speed, smaller images impair the detection ratio as well. This is confirmed by the detection rate for the same set of images shown in Figure 3.2. There is a tradeoff between face detection accuracy and detection speed, and one cannot achieve a fast detection speed by simply reducing the image size. It is therefore a challenge to achieve both high detection accuracy and fast detection speed.

Energy Consumption: As phones are usually battery operated, we need to reduce energy consumption as much as possible. To understand energy consumption, we measure the energy footprints for different device states that a face detection application may undergo: standby mode (screen off), CPU idle, previewing, image capture. and face detection. Figure 5 shows the measurement results. The image size for capturing and processing is 1280×960. WiFi and Bluetooth are turned off, the screen is lit up but kept at a minimum brightness unless otherwise explicitly stated. The energy consumption for standby mode and for a screen with max brightness is also measured for reference purposes.

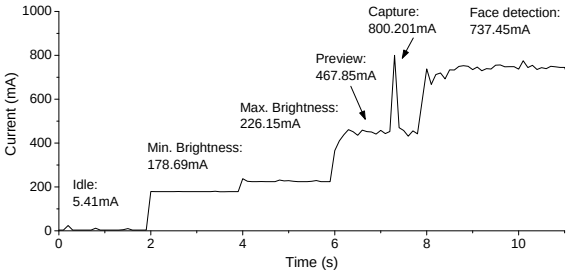


Figure 5: Energy consumption (measured in current strength) at different stages during face detection.

Note that current mobile devices commonly activate a preview state before taking a picture or recording a video. While it is necessary to let the user frame the scene and specify the focus point, such a preview stage can be safely avoided for a fisheye lens because of its extremely large depth of field, and there is no need for framing in our scenario. Therefore, the net energy consumption mainly consists of the power consumed by image capture and face detection, which is around 340 mA and 270 mA (after subtracting the power for preview, which takes about 290mA), respectively, as shown in Figure 5.

4. VIRI SYSTEM DESIGN

4.1 ViRi Overview

ViRi augments existing phone cameras with an external off-the-shelf fisheye lens, and relies on face detection to detect the viewing angle. Instead of developing a new face detection tool specially optimized for fisheye images, we try to leverage existing face detection SDKs [1, 3, 5] by developing effective pre-processing techniques to convert fisheye images into planar ones. In ViRi, a face detection module is treated as a *black box*. As the user's viewing angle on the device may change at any time, ViRi needs to perform continuous monitoring. This can lead to a severe waste of scarce battery resources.

To mitigate the problem, ViRi leverages the free system states and events, and low power sensors (accelerometer and compass, light sensor, proximity sensor) to filter out unlikely viewing situations and aggressively seeks further opportunities to skip face detection in real viewing situations. As a result, viewing angle detection is performed only when there is a significant situation change, such as a device attitude change, the user approaching or leaving, or a significant posture change. To further speed up face detection and reduce energy consumption, ViRi applies various prediction schemes to estimate possible face positions in the newly captured image based on context-aware hints. Intuitively, if a viewing situation change is caused by a device attitude change, then the angle between attitudes (measured using an accelerometer and compass) can be used for prediction. If it is due to a user posture change, we can predict the face area from the differences between neighboring image capture. We then use the small, predicted area of the image for face detection.

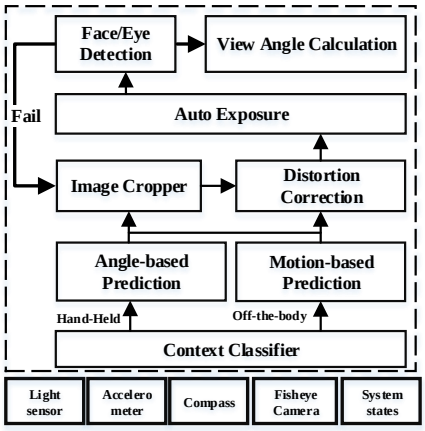


Figure 6: System architecture of ViRi.

4.2 ViRi Architecture and Workflow

ViRi runs as a background service on mobile devices (smartphone and slates), attempting to seamlessly adapt the display to the viewing angle of the user. The architecture of ViRi is shown in Figure 6.

First, ViRi uses a context classifier that takes input from system states, the accelerometer, light, and proximity sensors to identify the interested viewing situations including Hand-Held and Off-the-Body (e.g., on a desk), in which face detection might be conducted. In a Hand-Held viewing situation, when the device's attitude changes from one (quasi-)stationary state to another, we trigger face detection and further leverage the device attitude change angle to make a prediction. In an Off-the-Body viewing situation, face detection is triggered by significant posture changes, which are detected using the light sensor. The motion-based predictor is then called upon. The predicted face area is then cropped

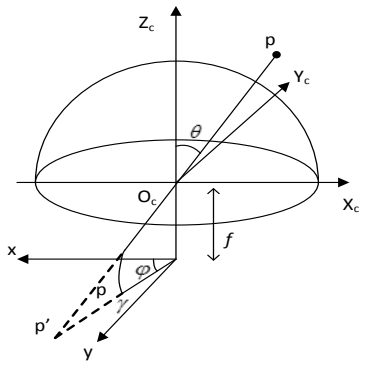


Figure 7: Fisheye lens model.

and fed into the distortion correction module. It further goes through the auto exposure module to equalize the lighting. Finally, face detection is conducted on the resulting image using existing face/eye detection SDK.

In the following sections, we will elaborate on the proposed techniques and present component evaluations.

5. FACE DETECTION AND VIEWING ANGLE ESTIMATION

Due to the view compression characteristic of fisheye lens, a face may be heavily distorted, especially when imaged in the peripheral area. Direct application of face detection tools developed for planar images suffers from low detection accuracy. As aforementioned, we hope to use existing face detection tools through proper pre-processing, including geometry distortion correction and harsh lighting handling.

5.1 Geometry Distortion Correction

Parametric Camera Model with Fisheye Lens: The calibration of a fisheye lens has been well studied. In this paper, we adopt a simplified version from [10]. The fisheye lens model is illustrated in Figure 7. Since the camera is rotationally symmetric, an incoming light ray is uniquely identified by the two angles θ and φ. Let $\mathbf{\Phi} = (\theta, \varphi)^T$. Let $\vec{p} = (p_u, p_v)^T$ and $\vec{m} = (m_x, m_y)^T$ be the pixel coordinates in the image and the ideal Cartesian coordinates in the sensor plane, then the simplified fisheye lens model can be described as

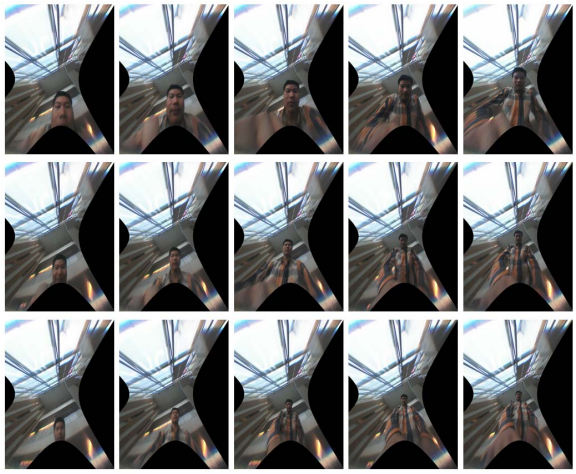
$$\vec{p} = \mathcal{A} \circ \mathcal{F}(\mathbf{\Phi})$$

where $\mathcal{F}(\mathbf{\Phi}) = \mathbf{r}(\theta)(\cos\varphi, \sin\varphi)^{\mathbf{T}}$ is the transform from ray direction $\mathbf{\Phi}$ to the ideal Cartesian coordinates, and $\mathcal{A}$ is the affine transform that forms the final pixel image from the projected image through alignment and scaling, i.e.,

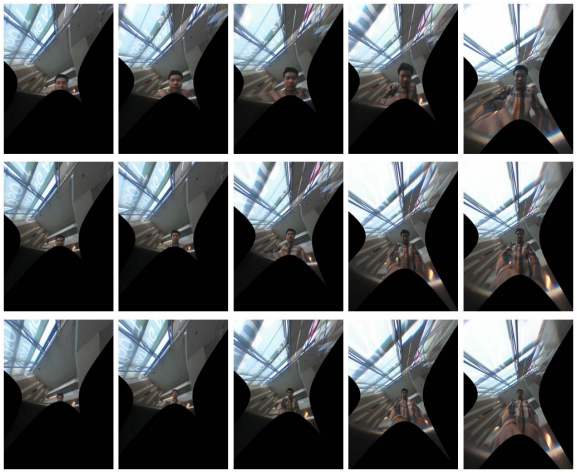
$$\mathcal{A}(\vec{m}) = \mathbf{S} \cdot (\vec{m} - \vec{m}_0)$$

with $\vec{m}_0$ being the possible misalignment between the center of the sensor plane and the light axis of the lens, and $\mathbf{S}$ is the scaling matrix that scales the size of the sensor plane to the actual image dimension in pixels.

Different choices of $r(\theta)$ lead to different camera models. For a fisheye lens model, we use a fourth-order polynomial $r(\theta) = \sum_{i=0}^{4} a_i \theta^i$. The calibration process then identifies all the parameters. We envision the fisheye lens is fixed in the device, so we perform an offline, once-for-all calibration process using a checkerboard and obtain the intrinsic model parameters of the fisheye lens. The procedure is an iterative process that fixes one set of model parameters ($\mathcal{A}$ or $\mathcal{F}$) and optimizes the other, and vice versa.



(a) Rectified Images



(b) Reoriented Images

Figure 8: Fisheye images after geometry distortion correction. Same set of images as shown in Figure3.

Radial Distortion Correction: The first pre-processing step is to rectify a fisheye image into a planar one to correct the radial distortion. Having learnt the parameters of the camera model, we find the mapping between a pixel $\vec{p}$ in a fisheye image and the corresponding pixel $\vec{p}'$ in the rectified image. The mapping is achieved by first resolving the incoming angle for $\vec{p}$ using the fisheye camera model, and then projecting it to another image using the pinhole model $r(\theta) = f \tan\theta$. Because the mapping is fixed for a given pixel position, this process can be effectively implemented via table lookup.

Figure 8-(a) shows the effect of radial distortion correction for the set of images shown in Figure3. From the figures, we can see that all the straight lines that were warped to curves in fisheye images have become straight again.

Perspective Distortion Correction: For many cases in which the face is in a peripheral area of the fisheye image, the perspective distortion stretches the face heavily, which

may render the face detection templates pre-trained from normal face images invalid. To handle this, we reorient the camera, i.e., rotate the fisheye image to center around the face, and achieve the effect as if the image were taken with the camera facing straight to the face. Suppose the face is at the incoming angle $\mathbf{\Phi}$, we simply rotate along the Z-axis by φ first and then rotate the resulting image along the Y-axis by θ. Note that, in real applications, this would create a *paradox*: we hope to rotate the image to center around the to-be-detected face position which is unknown yet. As detailed in Section 7, we will overcome this paradox through prediction: we predict the most likely face location and reorient the camera to center around that.

The effect of camera reorientation is shown in Figure 8-(b). From the figures, we can see that after camera reorientation, the view is centered around the face. Both radial distortion and perspective distortion are fixed and the face looks normal.

5.2 Harsh Lighting Handling

Due to the ultra wide FOV of the fisheye lens, light sources are often captured by the camera. This may lead to strong backlighting and underexposed faces, which usually needs to be intentionally avoided in face detection applications. In our settings, we do not alter a user's usage habits and the device can rest unattended. Hence we cannot manipulate the positioning of the faces nor exclude the light sources. The only possibility is to adjust the exposure and rely on signal processing.

As the face itself can never be a light source, therefore, we first overexpose by one stop (on the basis of the camera's center weighted light measurement results) when capturing images. We further adopt an auto-exposure technique [18] that better equalizes the resulting picture. In brief, it divides the whole image into regions with different exposure zones (borrowed from Ansel Adam's Zone theory), and estimates an optimal zone for every region while considering the details in each zone and the local contrast between neighboring zone regions. It then applies a detail-preserving S-curve based adjustment that fuses the global curve obtained from zone mapping with local contrast control.

5.3 Effectiveness of Pre-processing Techniques

To evaluate the effectiveness of pre-processing techniques, we invited 5 volunteers to help with data collection. For each volunteer, we captured 35 images from various viewing angles (0, 15, 30, ... 60 degrees) and various viewing distances (30, 40, ..., 70 cm). The images are captured in a discussion room with harsh lighting conditions. We plot the face detection rates using different processing techniques in Figure 9, where the X-axis represents different users.

From the figure, we can see the original (i.e., without pre-processing) detection rates vary from 40% to 71%. We applied auto-exposure and distortion correction separately to the raw image to determine their effectiveness in improving face detection rate. We can see that applying a single pre-processing technique already increased the detection rate for most cases. But the improvements were sometimes minor, such as for user 4 and 5. We then combined both techniques and found that there was significant improvement, 21% on average. In our implementation, we treated face detection as a black-box, and are thus unable to tell the exact reason for this observation. To our understanding, applying both techniques highly increases the probability of passing the face detection filters.

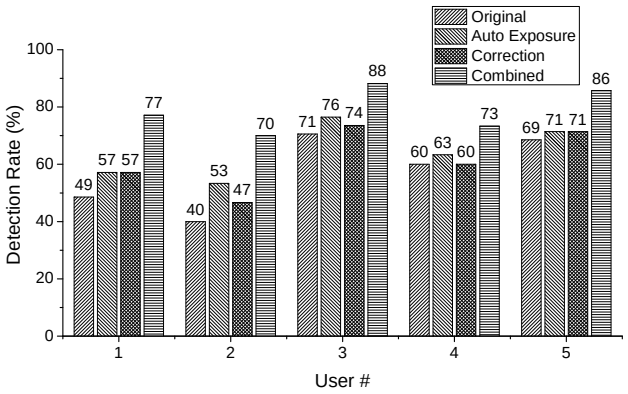


Figure 9: Effectiveness of geometry correction and the harsh lighting condition handling techniques.

5.4 Viewing Angle Estimation

Once we have detected the face, we next find the position of the eyes. Most existing face SDKs provide eye positions as a side product because they are typical feature points in the matching templates in face detection. We then take the middle point of the two eyes and reverse calculate the viewing angle using the camera model. We actually build up a mapping table during the calibration phase and perform simple table lookup when resolving viewing angles.

6. CONTEXT SIFTING

Energy consumption of ViRi mainly consists of two parts: the fixed overhead of image capture and the dynamic consumption of face detection that is dictated by its execution time. Thus, our strategy to save energy is two-fold: to skip unnecessary face detection situations as much as possible, and to reduce the face detection time if we have to.

In this section, we present context sifting that seeks to leverage the system states and low power sensors to rule out all situations of no interest, and to further identify opportunities to safely skip both image capture and face detection. For viewing situations that deserve new face detection, the sifting process also provides hints for face location prediction, which can speed up face detection, as will be elaborated on in the next section.

6.1 Ruling Out Situations of No Interest

Interested Viewing Situations: In typical viewing scenarios, users either hold the device or put the device on a table or stand. We refer to these two situations as *Hand-Held* and *Off-the-Body* viewing situations hereafter for the sake of clarity. In general, in whatever viewing situation, the device is stationary or quasi-stationary and the user always tries to maintain a stable relative position to the device for a better viewing experience. There are cases in which both the user and the device are moving (e.g., viewing while walking or on a moving bus), but these are rare cases and not encouraged as they are harmful to eyesight.

Detection of Interested Viewing Situations: We detect interested viewing situations via simple logical reasoning from several information sources, including system states

Situation	System state	Light sensor	Proximity sensor	Accelerometer	Reasoning logic
No interest	screen off AND no pending prompts	small or zero (<20)	close (<20cm)	motion	OR
Hand-Held	screen on	non-zero (>20)	far (>20cm)	quasi-stationary	AND
Off-the-Body	screen on OR pending prompts	non-zero (>20)	far (>20cm)	stationary	AND

Table 1: Interested normal viewing scenarios and their detections via sensors.

(screen is on, or is off but with pending prompts), IMU sensor (accelerometer and compass, for motion state sensing), light sensor (for environmental lighting), and proximity sensor (closeness to body). All these information sources have very low energy footprints.

The reasoning is presented in Table 1. We first leverage the system states, the proximity sensor and the light sensor to exclude some obvious non-interesting cases in which the user is definitely not viewing the screen. For instance, when the screen is off and there is no content pending to display, or the device is put very close to the face/body (via the proximity sensor), or in a pocket (via the light sensor), or the device itself is in a motion state (via accelerometer) etc., all these are impossible reading situations.

When all the sensors indicate a possible viewing situation, we then leverage the accelerometer to detect if the device is held in a hand or is resting on a table, which corresponds to a quasi-stationary or stationary state, respectively. Using the accelerometer to detect motion states is a well-studied topic [11]. In ViRi, we simply use the variance of the magnitude of acceleration to classify the device's motion states into three categories: motion, quasi-stationary, and stationary.

6.2 Skipping Unnecessary Detections

In real viewing situations, a user subconsciously tries to stabilize the relative position between her face/eyes and the device's screen. Even though the relative position cannot be completely stable and there are subtle variations, such small variations are usually compensated for by the human visual system. Therefore, there is no need to re-estimate the viewing angle in a stationary viewing state. When a large change in the relative position happens, the estimation of the new viewing angle is needed when it enters another stationary viewing situation. Frequent adaptation to small viewing angle changes will lead to a flickering display and actually impair the viewing experience.

A large relative position change is usually associated with a device attitude change or significant head/body movement such as a posture change. The attitude change would only happen in the Hand-Held case. It can be trivially detected using the accelerometer and compass, which may also tell the extent (in Euler angles) of change. For the Off-the-Body situation, posture changes will incur significant change in the lighting condition (seen from the device's view). We can use a light sensor to detect such changes, as elaborated below.

Leverage The Light Sensor: Figure 10 shows a trace of the light sensor for over 8 hours in a working day with normal phone use. Labels 1-4 indicate various zoomed-in situations in the sub-figures below. In particular:

- ◇ When the user moves closer to or away from the device, there is an apparent valley with gradual changes [Sub-figure (1)]. Similar effects are observed for people walking by at a close distance (less than 0.5 meter).

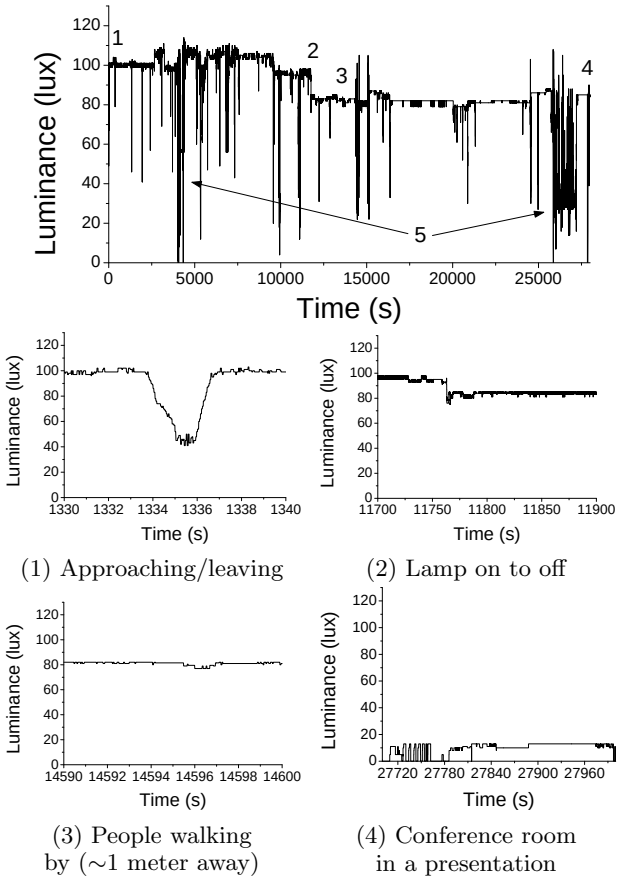


Figure 10: Lighting sensor traces observed by a device for over 8 hours in a typical working day.

- ◇ Turning off the luminance lamp causes a sudden drop in light sensor readings [Sub-figure (2)].
- ◇ When people are walking by at a distance over 1 meter, there is almost no impact on the sensed luminance [Sub-figure (3)].
- ◇ Small fluctuations are observed in a conference room with the projector showing some slides [Sub-figure (4)].

From the figures, we can see that the lighting conditions are mostly stable when there is no significant state change. Label 5 indicates the light sensor readings when the device is put in a pocket. The readings are mostly low but not always zero, because the light sensor resides on one end of the phone, which accidentally captures some light when in the pocket.

Our goal is to detect interested viewing situation changes such as when the user approaches/moves away from the device or changes her viewing posture significantly. Due to the various luminance conditions that the device may undergo, any single threshold-based triggering method, using either

absolute luminance or relative luminance changes, will not work. However, our observations show that a user's posture change, such as approaching/leaving, causes a continuous and monotonous change, in contrast to a sudden change such as turning on/off a lamp. Therefore, in our design, we use the increasing or decreasing trend to trigger face detection. The detection is simple conditioned the continuous increase/drop in light sensor readings for at least 500 ms.

In summary, we exploit the system states and the lower power sensors to filter out unlikely viewing situations, and identify two possible reading situations, Hand-Held and Off-the-Body, using the accelerometer and the light sensor. We perform new face detection only when the viewing situation becomes stationary again after the viewing angle changes.

7. FACE POSITION PREDICTION

The speed of face detection not only affects system energy consumption, but also has an essential impact on the user experience. As aforementioned in Section 3.2, the key is to reduce the size of the image that feeds into the face detection module, which is treated as a black-box in ViRi. This can be achieved via prediction of the potential face area. Due to the very different properties of Hand-Held and Off-the-Body situations, we design different prediction schemes for the two situations.

7.1 Angle-based Prediction: Hand-Held

If the contextual change is caused by the motion states or attitude of the device, we may exploit the relative orientation changes of the device, which can be obtained from the IMU sensor (accelerometer and compass) readings. We always record the orientation of the device when it enters a quasi-stationary state. If the device is manipulated and becomes stationary again, we calculate the relative rotation angles (Euler angles). If there is a face detected in the previous state, we predict where the face is likely to reside using the resulting rotation angle.

We now describe the general prediction process. We use $\vec{p}$ and $\mathbf{\Phi}$ to denote the pixel coordinates and the ray direction in the lens coordinate system C as shown in Figure 7. Then we have $\vec{p} = \mathcal{A} \circ \mathcal{F}(\mathbf{\Phi})$ (refer to Section 5.1). Thus, given the pixel $\vec{p}$ and ϕ (shown in Figure 7), we can calculate θ of the ray direction $\mathbf{\Phi}$. Suppose that the phone coordinate system changes to C' due to pitch, yaw, and roll. Assume that the origin of the coordinate system does not change (rotate around the optic center O_C in Figure 7). The new pixel coordinates of $\mathbf{\Phi}$ (denoted as $\vec{p}'$) are determined by θ' and ϕ', which are the angles between $\mathbf{\Phi}$ and the new $Z_{C'}$ and $X_{C'}$ axes, respectively. To calculate θ' and ϕ', we measure the Euler angles for pitch, yaw, and roll (illustrated in Figure 11) of the relative attitude change using the accelerometer and compass. The details of the calculation are omitted here due to space limitations. They can be found in [2].

In practice, when the user is holding the phone and viewing the display, it is unlikely that the phone is intentionally yawed or rolled (relative to the user) except when changing the device's orientation. The Euler angles of a yaw or roll are thus relatively small. We assume that the origin O_C is fixed, which may not hold in real situations. However, the relative position between the user and the device (held-in-hand) usually does not change significantly in a short time. Therefore, the proposed angle-based prediction can still work in practice.

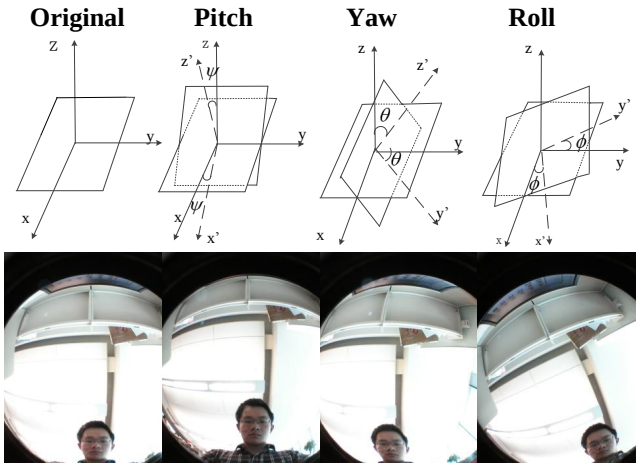


Figure 11: Illustration of Pitch, Yaw, and Roll changes between different stationary attitudes and their effects.

We have described the process of mapping one pixel in the previous image to the new image. We will now discuss how to generate the prediction window based on the face detected in the previous image. The algorithm is illustrated in Figure 12. We use $\vec{p}_l$ and $\vec{p}_r$ to denote the left and right eye in the previous image, respectively. First, $\vec{p}_l$ and $\vec{p}_r$ are mapped to $\vec{p}_l'$ and $\vec{p}_r'$ in a new captured image. Then, the prediction window's size is set heuristically with width $|\vec{p}_l' - \vec{p}_r'|$ and height $\frac{4}{3} \cdot |\vec{p}_l' - \vec{p}_r'|$, where $\frac{4}{3}$ considers the shape of the face.

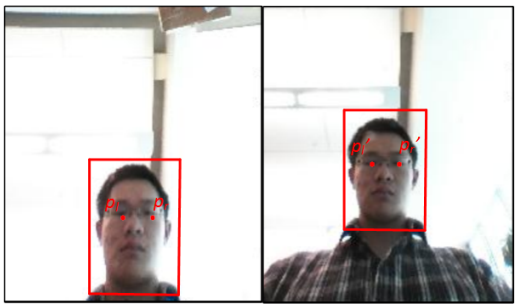


Figure 12: Prediction window generation based on eye feature points mapping.

7.2 Motion-based Prediction: Off-the-Body

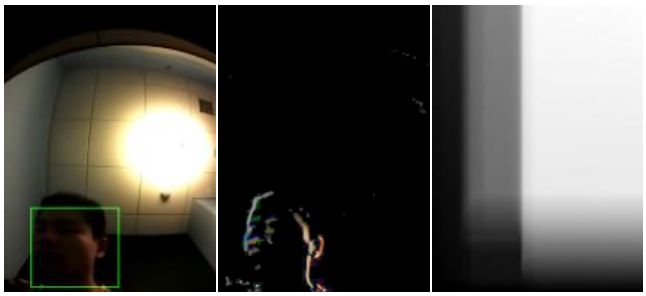
When the phone is put away from the body, e.g., on the table next to the user, the accelerometer will fail to capture the user's movement. Recall that we use the changing trend of the light sensor to detect any situation change. While the omnidirectional light sensor can trigger a state change, it cannot be exploited for prediction purposes. We have designed a motion-based prediction method employing the image sensor.

In such situations, the device is still, the user might be moving, slightly or significantly, and the background generally remains stationary. This is quite common from our user studies involving 10 volunteers. The part of the scene in motion is most likely just the user. Therefore, we have proposed a lightweight, motion-based prediction scheme, which takes two consecutive images captured at short intervals and

performs motion detection to identify potential face areas. One should note that motion-based prediction is inappropriate for when the device is hand-held as the background is also shifting. Applying more advanced motion detection algorithms such as those adopted in video coding systems may solve this problem, but will certainly add greatly to the computational costs. In contrast, we desire lightweight algorithms that better suit mobile devices.

We have designed an *integral-subtraction* image-based prediction scheme. Integral images are widely used to compute rectangle features [20]. The scheme includes three steps. First, we subtract two consecutive images in a pixel-wise fashion and obtain a subtraction image. The parts in motion will have large values in the subtraction image whereas the parts that are still will mostly have zero or small values. Second, an integral image is calculated based on the subtraction image. Third, a sliding window traverses through the integral subtraction image. The sum of the pixel values falling into the window is calculated. The window corresponding to the maximum value is the motion area and used as the prediction results.

Figure 13 shows the intermediate processes of the integral-substraction scheme. Two images are captured when the user was in front of a laptop with the phone beside the keyboard. The interval was 300 ms. The subtraction image is shown in Figure 13-(b). We can see that the subtraction image successfully captures slight posture changes. An integral image (shown in Figure 13-(c)) is then generated based on the subtraction image. The pixel at (x, y) stores the sum of all pixels from the subtraction image whose coordinates (x', y') satisfy $x' \leq x, y' \leq y$. We then traverse the integral subtraction image with a sliding window $[(x_1, y_1), (x_2, y_2)]$, where (x_1, y_1) and (x_2, y_2) are the left top and right bottom corners. The sum of all pixels within the window can be computed with $I(x_2, y_2) - I(x_1, y_2) - I(x_2, y_1) + I(x_1, y_1)$ where $I(x, y)$ represent the element (x, y) in the integral image. The execution complexity of the two-step scheme is proportional to the size of the image. In our implementation, the sliding window size is set to 320×320.



(a) Original (b) Subtraction(300 ms) (c) Integral

Figure 13: Exemplar original, subtraction, and integral of the subtraction images.

7.3 Prediction Effectiveness: Benchmarks

Angle-based Prediction: We collected 80 images from 2 users, each with around 40 images, at random device attitudes. We recorded the accelerometer readings when the images were taken. The maximum angle difference in our experiments reached 60 degrees. We manually labeled the eye positions for each image. We then randomly picked a pair of images (each represents one quasi-stationary state) to test the angle-based prediction scheme. The prediction accuracy is quantified by the pixel distance between the middle point of the predicted and the actual eye locations. We have plotted the cumulative distribution function (CDF) of the prediction errors in Figure 14.

We can see that over 90% of all prediction errors fall below 20 pixels. In practice, the size of the face in the image is typically larger than 60×100 pixels when the user was within a reasonable distance (say less than 1 meter). Thus, an error of 20 pixels is relatively small. Moreover, Figure 14 shows the error between pairs of randomly selected images. The actual rotation angles should be smaller in real viewing situations due to gradual natural transition. Therefore, we plot the prediction errors versus rotation angles in Figure 15. We can see the error is small ($\leq$ 10 pixels) when the rotation angle is below 40 degrees. The error exceeds 20 pixels only when the rotation angle is larger than 50 degrees, which rarely happens in practice.

Motion-based Prediction: We evaluated our motion-based prediction system using traces from 10 users. All traces were collected when users were sitting in their cubicle, using a laptop with their phone on the desk ($\sim$ 40 cm from the camera to the user's face). For each user, we recorded a video clip of 10$\sim$20 seconds and then extracted frames from the clip. The frame rate was 30 FPS. To calculate the subtraction image, we used two consecutive images interleaved by 300 ms (or 10 frames). This interval was sufficient to capture the user's motion while short enough to avoid ambient light change. We collected a total of 1400 test cases. For each prediction, we compared the predicted results with human labeled groundtruth. The successful rates are shown in Figure 16. The overall success rates are high. The worst case is observed for user 3. The reason is that during the video recording, the user's face moved out of the FOV for a while. In cases where the user's face remained in the FOV the whole time, the successful rates were all over 85%.

7.4 Prediction Failure Handling

Angle-based prediction depends on a previously detected face position, while motion-based prediction does not. Therefore, if there is no such face position recorded, angle-based prediction is always skipped. In rare situations that meet the criteria for interested viewing situations, but lack a face in the fisheye images (e.g., the phone is put on the desk and the user is away, with new prompts pending), we will end up with one vain prediction and detection. Prediction may fail. If this indeed happens, we would perform face detection over the entire frame. We notice that this is a limitation because of the black-box approach. Ideally, we would continuously expand the search area from the predicted area, to avoid repeated examination of the predicted area.

8. SYSTEM EVALUATION

8.1 Implementation

We built the ViRi engine on a Samsung Galaxy Nexus phone running Android 4.0. The executable file size was 1.11 MB with a memory footprint of 39 MB. One issue we faced in our implementation was the restriction imposed by the OS in terms of camera usage: we have to turn on the screen and enter the preview mode before we can take a image. That is, we are forbidden to automatically take a

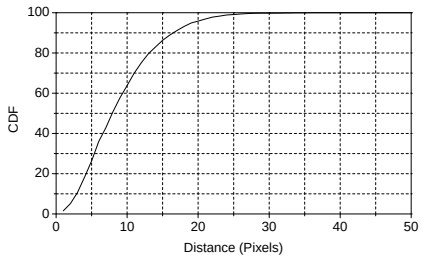


Figure 14: Angle-based prediction accuracy for different users.

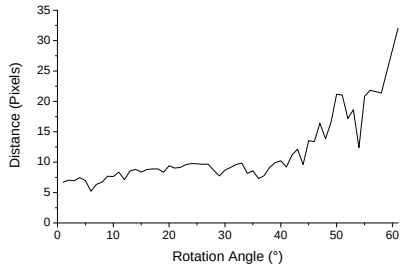


Figure 15: Angle prediction error vs actual rotation angles.

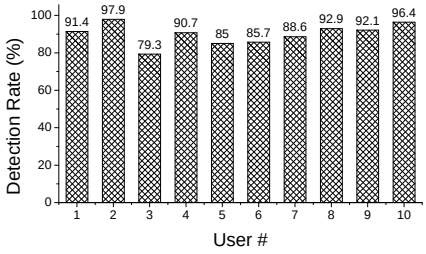


Figure 16: Motion-based prediction rates for 10 users.

photo without activating the screen and previewing. This brings significant overhead in energy consumption.

Perspective Distortion Correction: Due to perspective distortion, the effective display, which is perpendicular to the line connecting the eye and the screen center, has the shape of a *trapezoid*, as depicted in Figure 17, where the phone is laying on a surface and the effective display is shown by in dashed lines. ViRi seeks to properly adjust the

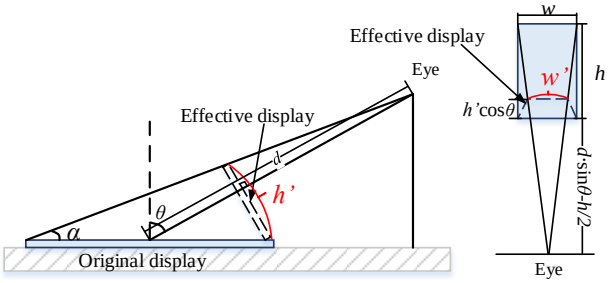


Figure 17: Illustration of the perspective distortion correction process.

to-be-displayed content to generate a normal looking image on the effective display. Obviously, the key task is to find the actual size of the effective display. Assume the distance between the eye and the center of the physical display is d, which is typically set to within the range of 0.3m-1m.

Given the estimated viewing angle θ, we calculate the dimensions of the effective display as follows: let h and w be the height and the width of the physical display. Evidently, the length of the bottom edge of the effective display is w, and we only need to find the length of the top edge (w') and the height (h') of the effective display. According to the Sine Theorem, h' is calculated by

$$h' = h \cdot \frac{\sin\alpha}{\sin(\pi - \theta - \alpha)}$$

where α is the angle shown in Figure 17. Note that α can simply be derived once d and θ are fixed. After obtaining h', w' can be calculated by

$$w' = w \cdot \frac{d \cdot \sin\theta - h/2 + h' \cdot \cos\theta}{d \cdot \sin\theta + h/2}$$

as illustrated in the right part of Figure 17. Once h' and w' are calculated, we can manipulate the to-be-shown content and fit it to the effective display.

8.2 System Evaluation

We have already evaluated component technologies. In this section, we evaluate the overall system performance, including end-to-end face detection accuracy, detection speed, and resource consumption.

Methodology: We separated the two common viewing scenarios and evaluated them separately. We invited different users and recorded their behavior via a camcorder, from which we manually identified the viewing angle changes and compared them with our system detection.

8.2.1 *Effectiveness of Prediction in Face Detection*

Hand-Held Situation: We invited five volunteers to try our prototype phone in typical usage scenarios, e.g., ebook reading, web browsing, and movie playback. The accelerometer triggered image capture when there was a transition between two quasi-stationary states as described in Section 7.1. The accelerometer readings were continuously recorded. The data collection for each volunteer lasted for tens of minutes with 30∼60 images captured. For each state transition, we applied angle-based prediction. The predicted face area was cropped out and fed to the face detection module. If a face was detected, we counted a successful hit. Otherwise, we performed face detection using the whole image. We present the prediction success rates in Figure 18.

We can see the overall prediction success rates varied from 0.78 to 1.00. The right bar in each cluster in Figure 18 shows the detection rate for the whole image without prediction. This accounts for the limitation of the face detection SDK. Note that the actual prediction success rates were higher than those shown in Figure 18, because a prediction should be concluded is a failure only when the face was not detected in the cropped image but successfully detected in the whole image. Thus, taking volunteer 4 as an example, the actual prediction success rate was $78/92 \approx 0.85$. One may notice that the results shown in Figure 18 look much better than those in Figure 9. The reason is that the data collections for Figure 18 were under normal reading conditions, whereas those for Figure 9 were under more stressful conditions, e.g., with harsher lighting.

Recall that the goal of prediction is to reduce the processing time by feeding the face detection black box with a small image. The size of a full image in our system is 768×1024 pixels. We observed the cropped image sizes varied from 0.16∼0.21 of the full size. The processing time decreased quadratically with the image size ratio as shown in Figure 4-(a). Therefore, the face detection after a successful prediction took only 1/25 or even less time than that for a full-sized image. Figure 19 shows the average processing times with prediction, including the handling of prediction failure. The right bar in the figure shows the average processing time for full-sized images. We found that the average

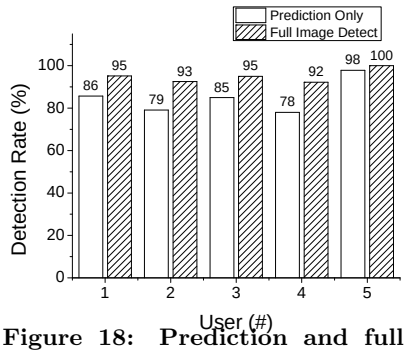


Figure 18: Prediction and full detection rates for 5 volunteers under normal device usage.

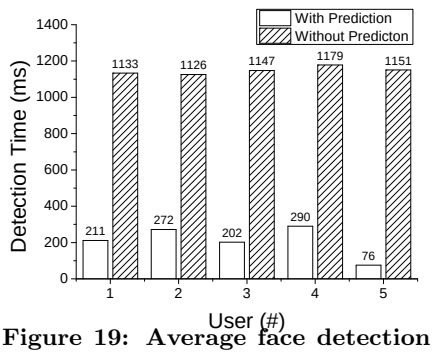


Figure 19: Average face detection times for 5 volunteers, with and without prediction.

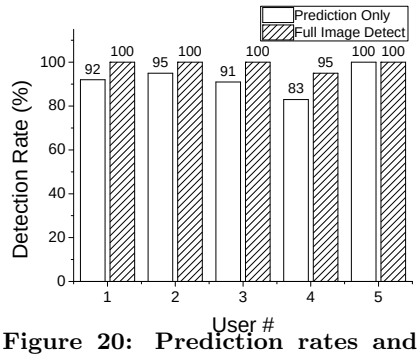


Figure 20: Prediction rates and detection rates for 5 volunteers in Off-the-Body scenarios.

processing time dropped to 76ms when the successful prediction rate was 98%. Though fail predictions led to full image processing, the average processing time with prediction still significantly outperformed those without prediction.

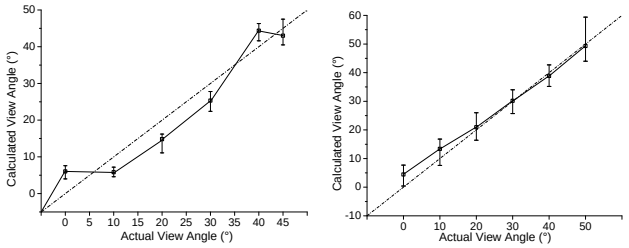
Off-the-Body Situation: We carried out similar experiments when the device was away from the body. We had 5 volunteers put the phone beside their keyboard when they were sitting in their cubicles. The phone invoked a face detection each time the light sensor detected an event such as when the user leaned towards the phone or the phone received notifications. For each volunteer, the light sensor triggered 7∼15 times during half a day usage. We observed the phone successfully captured all intentional events with only a small fraction of false alarms (10% $\sim$ 30% depends on the actual position of the phone).

For each prediction, the predicted area is cropped and fed into the face detection module. If a face was detected, we counted it as a successful prediction. Otherwise, a full scan was carried out to find the face. If a face was detected, we counted it as a successful detection. We show the prediction success rates and detection rates in Figure 20. We can see that the prediction rates varied from 0.83 to 1.00 across different volunteers. The mean processing times with and without prediction showed similar trends with Figure 19, which is therefore omitted due to space limitations. In conclusion, motion-based prediction significantly reduces the execution time of face detection in Off-the-Body scenarios.

8.2.2 *Evaluation of Viewing Angle Detection*

The view direction is determined by two angles θ and ϕ, as defined in Section 5.1. As the two angles are independent, we evaluated them separately for easy acquisition of ground truths. We first evaluated the detection of θ. We obtained 5 groups of images, each of which was from a volunteer. Each group contained images at 6 different viewing angles, i.e., 45°, 40°, 30°, 20°, 10°, and 0°.

We plot the mean estimated viewing angles versus the actual ones in Figure 8.2.2. The dashed line represents the ideal case, where the detected viewing angles equal actual ones, and the error bars show the max and min calculated angles among different image groups. We can see that the calculated results fit well with the ideal line. The errors seem independent from the actual viewing angle. The maximum error across all test cases was less than 8 degrees, which means our viewing angle detection algorithm satisfies moderate application requirements. We also evaluated the detection of ϕ in the same way as that for θ. The results



(a) Pitch θ (b) Roll ϕ

Figure 21: Viewing angle detection accuracy.

are shown in Figure 8.2.2. From the figure, we can see that average estimated angles fit well with the actual ones, and the max-min variation is within 10 degrees.

8.2.3 *Energy Consumption*

In ViRi, we need to monitor the viewing situations continuously, and we have exploited free system states and low power sensors (accelerometer, compass, proximity sensor, and the light sensor) to trigger more energy expensive camera sensors and face detection. The sensors themselves consume negligible energy. For example, the popular LSM330DLC accelerometer consumes about 11uA at its highest rate, and it drops to 6uA in low power mode, and the AK8975c compass consumes about 0.35mA at a high sampling rate.

However, in current smartphones, such background sensing needs to activate the main CPU and thus consumes more energy than necessary. We profiled all the low power sensors used by ViRi, and there are only a few mA differences when using those sensors at the highest sampling rate or not using them at all. Notice that the computational complexity of sensor sampling and processing are very simple. A low power MCU such as MSP430 series can do the job, and consumes only a few milliamperes. Actually, given the high demand of continuous sensing of user contexts by many mobile applications, future mobile phones may have more energy-efficient peripheral sensory boards [13, 14] or may adopt new CPU designs (e.g., [17]) that can dedicate one core for all the low energy sensing tasks.

Besides the energy consumption of the low power sensors, most of the energy consumed by ViRi is spent on image capture and face detection, which would depend on the actual usage pattern of different users. Recall that ViRi is triggered only when there are substantial contextual changes (motion, pending notifications, and lighting). To obtain a

real understanding of the energy consumption of ViRi, we surveyed 10 mobile phone users by checking the number of prompts, including SMS, push emails, reminders, and push notifications from their favorite social network applications (WeChat and Weibo). For push notifications coming in a batch, we treated them as a single event. So did the instant message sessions.

To account for different usage patterns, we conducted conservative estimation by assuming all the predictions would fail, and used the actual energy profiling results of the current implementation. As mentioned in Section 8.1, we have to go through the preview stage in the current implementation and thus consume more energy than necessary. According to the survey results, a user receives on average 30.4 events per day, with the standard deviation being 17.9 events. For each such event, in the worst case where all predictions fail, Viri consumes about 644mA on average for about 3 seconds, including the preview, image capture, prediction, and fail over to full-image face detection. Then the energy consumption for each detection is 0.54mAh. Multiply this by 84, the sum of average and three-times deviation, and the energy consumption is 45.1mAh, which is about 2.5% of a typical 1800mAh battery.

ViRi also executes when the user actually views the screen. In such cases, we perform viewing angle estimation only when the device changes from one (quasi-)stationary attitude to another significantly different one. In addition, the CPU and the screen are already on. Thus the CPU overhead for accelerometer sensing and the energy consumed by lighting up the screen should be excluded. Let's assume a user views the screen for three hours a day and the attitude changes every one minute, and assume a 90% prediction success rate (from Section 8.2.1). Under these settings, ViRi would consume about 10.45mAh, about 0.6% of a 1800mAh battery. Therefore, with a conservative estimate, the energy overhead of ViRi in typical usage is about 3.1% of total battery charge.

8.3 An Strawman App and Early Feedbacks

We have built a straw man mobile application that incorporates all the proposed viewing angle detection techniques. With the application, a user can provide a picture of interest and see the effect of viewing angle correction at arbitrary slanted viewing angles by either holding the phone and changing its orientation or resting the phone on a desk and changing her own viewing posture.

We performed a small-scale, informal user study. All users found that ViRi provides an intriguing viewing experience, especially when the tilting angle is large. Images with a black background showed better results than white ones. We think this is partially due to the OLED screen that remains completely dark for black pixels. The user trial also revealed some issues, some of which were expected. For example, ViRi's perspective correction could lead to either a full but smaller image or partial but larger image, due to the reduced effective viewing area. Figure 22 shows two such examples. The full but smaller one will sacrifice fine details of the original image, whereas the partial but larger one may only show a portion. All users suggested studying smart content selection, i.e., to show full or partial (full or partial what), and to show which portion when showing partial content. Some users were further concerned that if ViRi were applied on a larger display, then the top and bottom pixels would have different viewing angles, which might require luminance compensation. As could be expected, all users want ViRi to be an OS feature and support all applications.

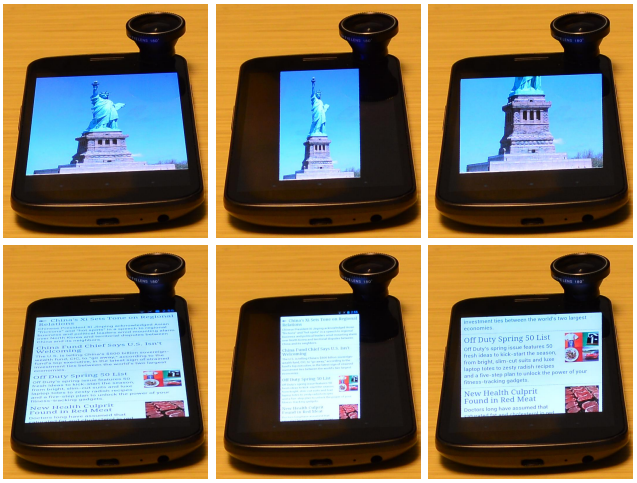


Figure 22: ViRi effects of full frame content. From left to right: original content at slanted view, ViRi with full but smaller image, ViRi with partial but larger image.

9. DISCUSSION

The effective working range of our prototype is only about 2 meters (1 meter to each side of the lens). We conjecture this is partially due to the concatenation of the fisheye lens and the existing phone camera system, which actually reduced the effective FOV of the fisheye lens, and also partially due to the low resolution of the front-camera of the phone we used. We expect if a high resolution, genuine fisheye camera were used, the working range and detection ratio would increase significantly.

The adoption of a fisheye camera in ViRi opens up new opportunities and challenges for many computer vision technologies such as face detection. We have adopted existing face detection tools and focused on various pre-processing techniques in order to use them directly. There are two possible improvements to improve face detection. First, we can apply super-resolution technologies to mitigate the view compression problem of fisheye cameras to increase the face detection rate. Second, we may directly perform face detection on fisheye images.

The high computation complexity of face detection makes Cloud services an appealing solution. We chose not to use the Cloud because it would create dependency on the network infrastructure. Moreover, pictures are usually of large size and wireless communication is also energy hungry, sending pictures to the Cloud may consume more energy than local processing, putting aside the long transmission latency.

The wide FOV of a fisheye lens makes it possible to include multiple faces in certain situations such as in a meeting. This may confuse the motion-based prediction. We have not properly addressed this issue. One strategy is to detect the largest face area and track the user or to simply disable this function if multiple faces are detected. After all, a mobile device is a personal device.

We have mainly focused on perspective correction for on-axial slanted viewing angles, i.e., our sight line is perpendic-

ular to one of the device's boundary frames. In real world situations, we may end up with off-axial slanted viewing angles. In principle, as previously shown, we can detect the actual viewing angles (θ and ϕ), it is more challenging to show perspective corrected content on the screen, as the effective viewing area decreases at the order of $\cos\theta \cdot \cos\phi$.

10. RELATED WORK

Fisheye Image Correction: Several efforts [6, 19, 21] have been made to build a more natural view for a fisheye captured image. In [19], a fisheye photo is divided and projected onto multiple planes, leaving sharp seams between joint scenes. Users can specify where to put the seams in order to make them unnoticeable. In [6], the properties to be preserved are controlled by the user, and the mapping from the view sphere to the image plane is performed via weighted least-squares optimization. These schemes improve distortion correction performance at the cost of human intervention and a high computation cost, which are not desirable for our scenario. Thus, we use a simplified single global mapping to preprocess each image before face detection. The global mapping allows efficient computation via simple table lookup.

Mobile Applications Exploiting Face Detection: It is natural to have face detection on mobile phones for biometric unlocking, self-portrait, and mood sensing [15]. The key constraints for applying face detection on mobile phones are the limited computational resources (memory and CPU) and variable environments [8]. As a result, existing face detection libraries are ported carefully to mobile platforms to minimize memory usage and do hardware specific optimization. Examples [5] include OpenCV4Android, OpenCV iOS, and FaceDetection WinPhone (OpenCV in C#). The latest Android OS already provides native APIs for face detection. In ViRi, we simply leverage existing SDKs instead of developing special face detection algorithms for fisheye images.

Eye Tracking: Eye tracking has recently gained the attention of researchers. In [12], the authors proposed using eye blinks to activate a smartphone app. To this end, they tracked and mapped the user's eye to the smartphone display. Another piece of recent work [9] focused on detecting the blink rate of the user using a Samsung Galaxy Tab. A key challenge in their work was to track the eye in spite of camera motion. They developed an accelerometer-based eye position prediction algorithm that exhibits some similarity to our angle-based prediction scheme. As a salient feature of the Galaxy SIII, Samsung introduced SmartStay, which maintains a bright display as long as the user is viewing the screen [16]. The key differences are that we cover both Hand-Held and Off-the-Body situations and we avoided unnecessary detection using low power sensors to trigger image capture, which significantly reduces power consumption.

11. CONCLUSION

In this paper, we have mainly studied the viewing angle estimation problem of ViRi, which aims to achieve an all-time front-view viewing experience for mobile users in realistic viewing situations. We propose augmenting an existing phone camera system with a fisheye lens and use face detection tools to detect the actual viewing angle. We have come up with effective preprocessing techniques to correct the severe distortion of fisheye images. We have designed and evaluated several techniques for energy savings, including context sifting using low power sensors to maximally skip unnecessary face detection, and efficient prediction techniques to speed up face detection when there is a real need. We have also built a straw man application to allow users to experience the effect of viewing angle correction.

We think ViRi represents an initial step towards a more general direction of mobile sensing: allowing the device to gain perpetual awareness of it user, making mobile devices more intelligent and serving its users better. It is an extremely difficult task under normal usage styles and worth more study. For ViRi, our next step is to further study and integrate the display adjustment to the graphics pipeline of the operating system to support all applications.

12. REFERENCES

[1] Android Face Detector. `http://developer.android.com/reference/android/media/FaceDetector.html`.

[2] Euler Angles. `http://mathworld.wolfram.com/EulerAngles.html`.

[3] Microsoft Face SDK. `http://research.microsoft.com/en-us/projects/facesdk/`.

[4] Olloclip. `http://www.olloclip.com/`.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] R. Carroll, M. Agrawal, and A. Agarwala. Optimizing content-preserving projections for wide-angle images. In *SIGGRAPH '09*, 2009.

[7] L.-P. Cheng, F.-I. Hsiao, Y.-T. Liu, and M. Y. Chen. irotate: Automatic screen rotating based on face orientation. In *CHI 2012*, 2012.

[8] A. Hadid, J. Heikkila, O. Silven, and M. Pietikainen. Face and eye detection for person authentication in mobile phones. In *ICDSC '07*, 2007.

[9] S. Han, S. Yang, J. Kim, and M. Gerla. Eyeguardian: a framework of eye tracking and blink detection for mobile device users. In *HotMobile '12*, 2012.

[10] J. Kannala and S. S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Trans. Pattern Anal. Mach. Intell.*

[11] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *UbiComp'12*.

[12] E. Miluzzo, T. Wang, and A. T. Campbell. Eyephone: activating mobile phones with your eyes. In *MobiHeld '10*.

[13] B. Priyantha, D. Lymberopoulos, and J. Liu. Littlerock: Enabling energy-efficient continuous sensing on mobile phones. *Pervasive Computing, IEEE*, 2011.

[14] M.-R. Ra, B. Priyantha, A. Kansal, and J. Liu. Improving energy efficiency of personal sensing applications with heterogeneous multi-processors. In *UbiComp '12*, 2012.

[15] N. D. L. Robert LiKamWa, Yunxin Liu and L. Zhong. Moodsense: Can your smartphone infer your mood? In *PhoneSense workshop*, 2011.

[16] Samsung. Galaxy S3 Smart Stay. `http://www.samsung.com/global/galaxys3/smartstay.html`.

[17] Texas Instruments Inc. TMS320C6472 Datasheet. `http://www.ti.com/lit/wp/spry130/spry130.pdf`.

[18] L. Yuan and J. Sun. Automatic exposure correction of consumer photographs. In *ECCV'12*, 2012.

[19] L. Zelnik-Manor, G. Peters, and P. Perona. Squaring the circles in panoramas. In *ICCV '05*, 2005.

[20] C. Zhang and Z. Zhang. A Survey of Recent Advances in Face Detection. In *TechReport, MSR-TR-2010-66*, 2010.

[21] D. Zorin and A. H. Barr. Correction of geometric perceptual distortions in pictures. In *SIGGRAPH '95*.