iRotate: Automatic Screen Rotation based on Face Orientation

Lung-Pan Cheng[1], Fang-I Hsiao[1], Yen-Ting Liu[2], Mike Y. Chen[1]
[1]Department of Computer Science and Information Engineering
[2]Department of Electrical Engineering
National Taiwan University
{r98922168,b96902005,b97901121,mikechen}@csie.ntu.edu.tw

ABSTRACT

We present iRotate, an approach to automatically rotate screens on mobile devices to match users' face orientation. Current approaches to automatic screen rotation are based on gravity and device orientation. Our survey of 513 users shows that 42% currently experience auto-rotation that leads to incorrect viewing orientation several times a week or more, and 24% find the problem to be very serious to extremely serious. iRotate augments gravity-based approach, and uses front cameras on mobile devices to detect users' faces and rotates screens accordingly. It requires no explicit user input and supports different user postures and device orientations. We have implemented an iRotate that works in real-time on iPhone and iPad, and we assess the accuracy and limitations of iRotate through a 20-participant feasibility study.

Author Keywords
Device orientation, Face detection

ACM Classification Keywords
H.5.2 [**User Interfaces**]: Interaction styles; Input devices and strategies

INTRODUCTION

Mobile devices, such as iPhone, Android phones, and tablets, support screen rotation to optimize content layout and viewing experience. For example, users may browse the web in portrait mode, and rotate the devices to view photos and videos in wide-screen mode. Current approaches to automatic screen rotation assume that users are using devices upright, and uses input from accelerometers to trigger rotation so that the top of the screen is always pointing up [3,12,18] (i.e. opposite the direction of gravity). This gravity-based rotation leads to incorrect screen orientation when users change their postures, such as when using the devices in bed, and when devices are placed on a flat surface.

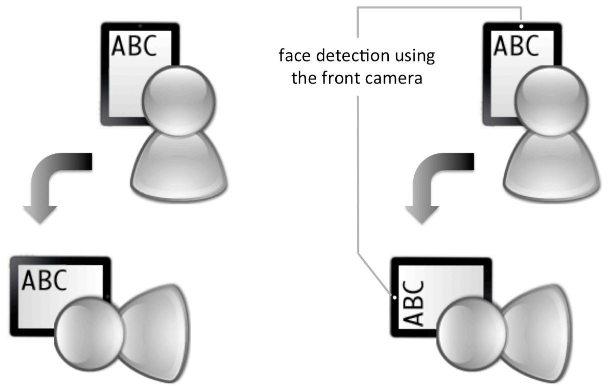

Figure 1. Automatic screen rotation based on: (a) device orientation and gravity (b) iRotate, which automatically detects user's face orientation and rotates screen to match.

Figure 1(a) shows how gravity-based rotation works correctly while user is upright, but is consistently wrong when users' lie down on one side. In order to understand how common and when incorrect rotation happens, we conducted an online survey of 513 smartphone and tablet users. 91% of the users reported that they have experienced auto-rotation that leads to incorrect viewing orientation, and 42% reported that the problem occurs several times a week or more. The most common occurrence of incorrect rotation is when users are lying on one side (57%) followed by when users are putting down the devices (23%).

A common solution to address incorrect auto-rotation is to provide users with the option to manually disable auto-rotation, which locks the device into the current screen orientation. For example, Apple introduced a hardware switch to lock screen rotation in iPad 1, and added software settings to lock screen rotation in iOS 4.0. In addition, a lock icon has been added to the status bar to visually indicate the state of the rotation lock. Most Android devices also provide similar software settings.

This manual approach, however, requires cognitive effort and explicit input to toggle the auto-rotation lock. Even with the visual lock indicator, our survey of 513 users showed that for users who have used rotation lock, 58% have had experience forgetting to turn the lock off. Several gestures have been proposed to temporarily override auto-rotation [11,17], they still require user to learn the gestures, as well as requiring explicit user input.

We present iRotate, which automatically rotates screens on mobile devices based on users' viewing orientation instead of gravity and device orientation. Figure 1(b) shows that it rotates screens correctly when users are upright and when lying down on one side. iRotate uses the front cameras of mobile devices to detect users' faces, and rotates the screens to match users' viewing orientation. It supports usage in various user postures without requiring any explicit user input.

In addition, instead of constantly capturing and analyzing data from the front camera, we use accelerometers to sense changes in device orientation and then trigger face detection. This optimization helps reduce computation and battery usage.

Because iRotate uses computer vision for face detection, it needs to see sufficient portions of users' faces, which may be obscured by fingers and may move outside the cameras' field of view. In order to assess the practicality and limitations of this approach, we conducted a 20-participant study to evaluate the percentage of times that users' face orientations are detectable. We asked the participants to perform reading tasks on smartphones and tablets while they transition between sitting upright and lying down on one side, and also asked them to rotate the devices by 90 degrees in those postures.

We recorded videos using the front cameras and analyzed the video frames to evaluate the feasibility of iRotate and how well it works on current mobile devices. For feasibility, we manually reviewed all video frames and labeled the face orientation when sufficient facial features are present. In addition, we implemented iRotate on iPhone and iPad using the face detection API introduced in iOS 5. Our results show that sufficient facial features are present to correctly detect face orientation within 77.6% of the time. The iOS face detection API can correctly identify face orientation 10.7% of the time.

Our paper makes the following contributions: 1) a new approach to automatic screen rotation based on users' face orientation instead of device orientation, 2) quantified the feasibility of using front-camera based approach.

RELATED WORK

Fitzmaurice et al. described Rotating User Interface which rotates in discrete, 90 degrees steps [6]. Various types of sensors have been used to sense changes in device orientation to automatically rotate screens relative to gravity. Schmidt et al. [18] used two mercury switches to implement auto-rotation so that users can change between portrait and landscape views of documents by simply rotating the device. Also, users can take turns in interacting with the same device by simple handing it over without having to rotate it. Bartlett et al. [3] and Hinckley et al. [12] both used tilt sensors (2-axis accelerometers). The latter group used 45-degree angle threshold to detect orientation changes more quickly. They also used 0.5 second threshold and 5-degree dead band for rotation stability.

Gestures have also been proposed to trigger rotation and override auto-rotation. Apple patented a two-finger rotation gesture to rotate screens [17], and placing both thumbs on the screen as users rotate their devices to trigger rotation without using other sensors [7]. Sensor synaesthesia [11] proposed touching the screen with one finger and holding it down while rotating the device, in order ice and user movement as interaction techniques. For example, VideoMouse [13], which based on the design of Rockin'Mouse [1], uses a camera to extend degree of freedom (DOF) for rotation, zooming and translation. TinyMotion [23] senses 6DOF without additional sensors by analyzing image sequences captured by built-in camera. iSeeU [20] used front cameras on mobile devices to detect face movement for scrolling and zooming.

Face detection APIs are increasingly being integrated into mobile platforms, which makes the capability readily available to all applications on those platforms. Native, face detection APIs were introduced in Android 1.5 and iOS 5. Hannuksela et al. [8] presented a feature based head tracking method which can run in real-time on a resource limited mobile device.

Gábor Blaskó et al. [4] use a Haar-feature-based face detector [16] and Continuously Adaptive Mean Shift (CAMShift) tracking algorithm [5] to track users' head roll angle and update the display when the inferred orientation has changed more than 15°.

SURVEY: AUTO-ROTATION IN THE WILD

To help us better understand how well auto-rotation works in the real world, we created an online questionnaire for smartphone and tablet users. The questionnaire focused on: 1) when and how incorrect rotation occurs, and 2) awareness and usability of the rotation lock. To ensure we get responses from users who regularly use devices that have both auto-rotation and rotation lock, we screened for iPhone and iPad users. We received a total of 513 responses from 394 iPhone users and 119 iPad users. Their average experience using the devices was 9.9 months and 4.7 months respectively. The respondents' age ranged from 18 to 48 years old (average 26) and 66% were male.

When and How Incorrect Rotation Occurs

Of the 513 respondents, 91% reported that they have experienced incorrect auto-rotation, 42% reported that the problem occurs several times a week or more, and 24% reporting it to be a serious to very serious problem. When asked about the most likely cause of incorrect rotation, 57% reported lying down on one side, followed by placing the devices on a flat surface at 23%, and lying down and facing up at 19%. These incorrect rotations are caused by the

Question		iPhone	iPad	Overall
How often incorrect rotation occurs	Several times a day	18%	24%	21%
	Once a day	3%	6%	5%
	Several times a week	18%	14%	16%
	Once a week or less	51%	47%	49%
	Never	9%	8%	9%
When incorrect rotation occurs	Lie down on one side	66%	47%	57%
	Place the device on the flat surface	11%	35%	23%
	Lie down and face up	21%	16%	19%
How serious is the problem of incorrect rotation?	Not serious at all	15%	15%	15%
	Not very serious	26%	26%	26%
	Somewhat serious	35%	31%	34%
	Very serious	19%	19%	19%
	Extremely serious	5%	9%	5%

Table 1. Survey results from 513 users on incorrect auto-rotation.

Question		iPhone	iPad	Overall
Aware of rotation lock	All users	70%	84%	77%
	Novice users	49%	83%	66%
	Experienced users	80%	86%	83%
Usage of rotation lock	Users who make use of it	74%	84%	79%
	Forget to unlock when they need auto-rotation	61%	55%	58%

Table 2. Survey results from 513 users on the usage of rotation lock.

rotation algorithm rotating the screens relative to gravity, while users are expecting the screens to match their view orientations. In particular, a significantly higher percentage of iPhone users chose lying down on one side as the most likely cause, at 66% compared to 47% for iPad users. Details of the results for iPhone and iPad are shown in Table 1 and Table 2.

Awareness and Usability of Rotation Lock

70% of iPhone respondents were aware of the rotation lock on their devices. For iPad users, the percentage is higher at 84%. To understand how quickly users learn about rotation lock, we compare novice users who have used their devices for less than four months, to all other users. For novice iPhone users, only 49% of them are aware of the rotation lock, compared to 80% for more experienced users. Novice iPad users become aware of the rotation lock more quickly, with 83% being aware of it compared to 86% for more experienced users. One possible reason for earlier and higher awareness on the iPad is because it has a hardware switch that can be configured by users, via a software setting, to be a rotation lock or a mute switch. The hardware switch defaults to a rotation lock on iPad 1, but defaults to a mute switch on iPad 2.

Of those who are aware of the rotation lock, 74% of iPhone users and 87% iPad users make use of it. Among those, 61% iPhone users and 55% iPad users reported that they would forget to unlock when they need auto-rotation. Given that the visual rotation lock status is prominently shown on the status bar of these devices.

DESIGN

Our goal is to automatically rotate screens to match users' viewing orientation. Accelerometers are capable of detecting device movement and device orientation relative to gravity, but cannot detect users' viewing orientation. We use front cameras, which are increasingly common on mobile devices, to capture videos of users and detect their face orientation. A simple approach can be entirely camera-based [4], without using any additional sensors. This approach, however, requires constant face detection which requires significant computation and power. In addition, face detection may be challenging in situations such as in low light and in rapid movement.

By augmenting 3-axis accelerometers with camera-based face detection, we use face orientation when it is available, and fall back to gravity-based rotation otherwise. The accelerometer is used to detect changes in device orientation, which can be caused by device rotation or by changes in users' posture. Once a change in device orientation is detected, iRotate turns on face detection and rotates that screen if users' face orientation, relative to the device, has changed. This approach reduces computation and power consumption by not running face detection constantly. In addition, it further simplifies computation by constraining the possible face orientation, relative to device, from three (current, left, and right) to two (current, plus one of left or right).

Orientation Threshold

We use an orientation threshold to trigger auto-rotation. As shown in Figure 2, we define θ as the angle between device's x-axis and earth's horizontal plane, and φ as the angle between device's y-axis. We experimentally measured the orientation threshold used by iPhone and iPad, by monitoring the accelerometer readings and rotating the devices as slowly as possible until the screen rotated. We found that the threshold is θ - φ =30, with 2 degrees of dead band, for both iPhone and iPad. We use the same threshold to trigger face detection for iRotate.

The detailed orientation threshold is defined as follows

if $\theta - \varphi > 30$, rotate to landscape

if $\varphi > 30$ and $\theta \leq \varphi$, rotate to portrait.

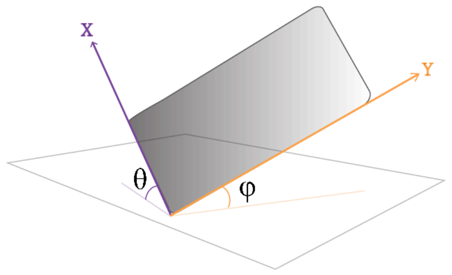

Figure 2. Orientation threshold angles.

FEASIBILITY STUDY

Our user study has the following two goals: 1) assess feasibility of face orientation detection using front cameras, and 2) evaluate face orientation detection performance on current mobile devices.

Device

We measured the front camera's field of view (FOV) on three popular smartphones and three tablets, shown in Table 3 and Table 4. We found that the smartphones had nearly identical FOV and had the same VGA resolution (640x480). iPad 2 and Motorola Xoom had similar FOV, and HTC Flyer tablet having narrower FOV. Their resolution ranged from VGA to 2 mega-pixels. We selected the iPhone 4 and iPad 2 for our user study, as they have similar field of view to other devices, and are popular on the market.

In terms of processors, iPhone 4 uses Apple's A4 CPU, which has a single ARM Cortex-A8 core running at up to 1GHz. iPad 2 uses Apple's A5 CPU, which has dual ARM Cortex-A9 cores running at 1GHz. Both of these processors are comparable in performance to other high-end device.

Smart Phones	iPhone 4	HTC Desire S	Google Nexus S
Portrait	44	43	44
Landscape	55	56	57

Table 3. Front camera's field of view for smartphones

Tablets	iPad 2	HTC Flyer	Motorola Xoom
Portrait	44	33	44
Landscape	57	51	59

Table 4. Front camera's field of view for tablets

Experiment

Participants were asked to perform a reading task, and scrolling as necessary, to simulate typical usage of smartphones and tablets. The experiments started with participants sitting on the edge of a sofa bed and holding a mobile device in portrait mode. The participants followed on-screen prompt messages, and went through the following four tasks while reading: 1) lie down on one side, 2) rotate the device 90 degrees, 3) return to the sitting position, 4) rotate the device 90 degrees. Each posture lasted 30 seconds. Each participant performed one trial using an iPhone 4 and one trial using an iPad 2. The order of the devices was counter balanced across participants.

We recruited 20 participants, 10 female, from our university population. The participants' age ranged from 21 to 36.

Implementation

We developed a custom iOS app for the experiment. It was a universal app optimized for both iPhone and iPad's screens. The app displayed reading material and prompted users to perform the next task every 30 seconds. It recorded video at 30 frames/second using the front cameras during the entire experiment for our offline, "wizard of oz" analysis. In addition, it recorded 3-axis accelerometer readings at 10Hz.

To evaluate how well face detection performs on current mobile devices, we used the face detection API (CIDetector) introduced in iOS 5. Given an image and its orientation, the detection API returns face features, specifically the locations of eyes and mouth, if they are present in the image. Up to four possible image orientations (portrait, landscape left, landscape right, portrait upside down) need to be tested, in order to determine if a face is present and what its orientation is.

While recording the video throughout the experiment, we streamed those video frames to the face detection API in real time. Using the readings from the accelerometer, we were able to reduce the set of possible orientations from four to two (i.e. the current orientation, plus left or right). On iOS 5 beta 7, the average face detection speed we observed in all experiments was 2.6 images/second on iPhone 4 and 7.8 images/second on iPad 2. Because we must test two possible orientations for each image, the effective face orientation detection rate was 1.3Hz for iPhone 4 and 3.9Hz for iPad 2.

Our functional prototype automatically rotates screens to the orientation detected by the face detection API. It counts the number of frames with detected face orientation within a 0.5-second window, and rotates to the most frequently detected orientation. The 0.5-second threshold is the average rotation delay for iPhone and iPad, and is also the value proposed by Hinckley et al. [12]. If the frequency of multiple possible orientations turned out to be the same, or there was no face detected from those frame, then it rotates screen according to the 3-axis accelerometer.

Results

For face orientation detection to be feasible, sufficient facial features must be within the front camera's field of view. Figure 3 shows examples of partial faces that contain sufficient information to identify their orientation. Figure 4 shows examples that we were unable to identify the face orientation. Common causes include incorrect exposure, fingers obscuring the cameras, and camera shake.

Detection Performance

To assess the feasibility of our camera-based approach, we used the "wizard of oz" method and classified each captured video frame manually. Specifically, we looked at the one-second window starting from moment the auto-rotation threshold angle was reached ($T_{threshold}$), as detected by the accelerometers. In total, we labeled 4800 frames (30 frames/second x 1 second x 20 participants x 2 devices x 4 tasks), and we were able to identify users' face orientation in 77.6% of the frames.

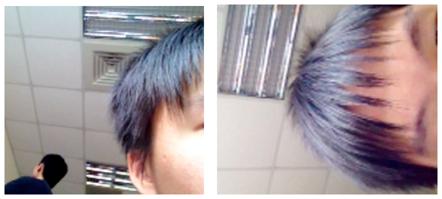
Figure 3. Features of face. Edge of face

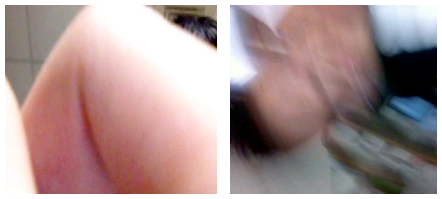
Figure 4. Finger block the camera

To assess the performance of current devices, we compare the detection results returned by the iOS face detection API to the correct orientation. A frame is considered to be correctly detected if and only if a face has been detected and the orientation is correct. A total of 618 detection were performed, and the API was able to identify users' face orientation correctly 10.7% of the time. This detection performance is significantly lower than what is feasible, which is up to 77.6%. This is likely because the front camera usually captures images that only have part of users faces, but the iOS algorithms require that user's eyes and mouth be present. Algorithms that only require partial face features are likely to improve the success rate.

Figure 5 and Figure 6 shows the detailed success rate for each of the four tasks which, from left to right, are: 1) user, holding the device in portrait, changed posture from sitting to lying down, 2) user rotated device from portrait to landscape while lying down, 3) user changed posture from lying down to sitting, and 4) user rotated device from landscape to portrait while sitting.

The front camera's position on the devices has strong effects on detection rate, as they are easy to be obscured in landscape mode. The first task had the highest success rate compared to the rest, because the front cameras are near the top of the devices when held in portrait orientation, and the cameras had a clear view of the user. For iPhone, the lowest success rate is when users were holding it in landscape mode and transitioning from lying down to sitting. Because the iPhone's front camera, in landscape mode, is under the users' thumbs, the cameras were obscured a significant portion of times.

The weight of the iPad made rotation more difficult while lying down compared to sitting upright and compare. The instability caused a large number of blurry images as well as many images without users' faces, causing that iPad task to have the lowest success rate across both devices and across all tasks.

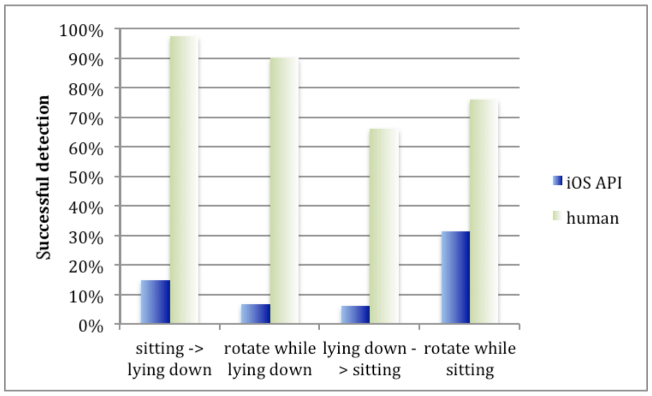

Figure 5. Successful face orientation detection rate for iPhone

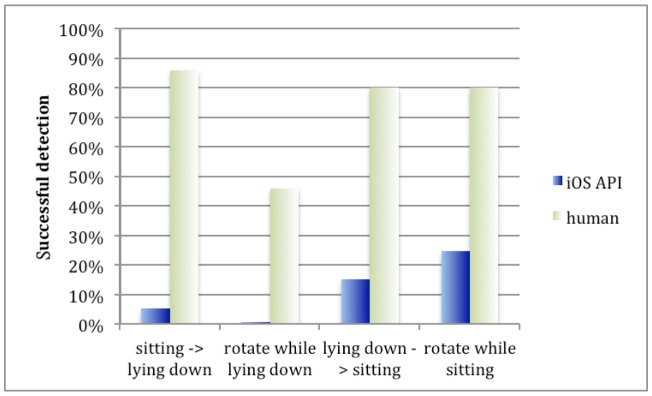

Figure 6. Successful face orientation detection rate for iPad

Feasibility Analysis

We present a feasibility analysis of iRotate across all users trials. In a trial, as soon as a frame has a detectable face orientation, iRotate can correctly rotate the screen accordingly. That trial is considered to be successful, even if all subsequent frames do not have detectable face orientation.

Figure 7 shows the cumulative success rate across all trials, for the four orientations. The x-axis shows the time elapsed after $T_{threshold}$, which is the moment that the orientation threshold angle has been exceeded, as detected by the accelerometers. The four figures correspond to the four tasks and orientation changes. The vertical dotted line, drawn at 0.5 seconds, shows the stability threshold proposed by Hinckley et al. [12].

For example, although the successful detection rate for rotating iPad while lying down is only 45.7%, the cumulative success rate starts at 60% at 0.1 seconds, improves to 65% at 0.5 seconds, and 80% at 1.0 second. Overall, iRotate's front camera-based approach has success rate ranging between 65-100% at 0.5 seconds, and between 75-100% for the four tasks on iPhone and iPad.

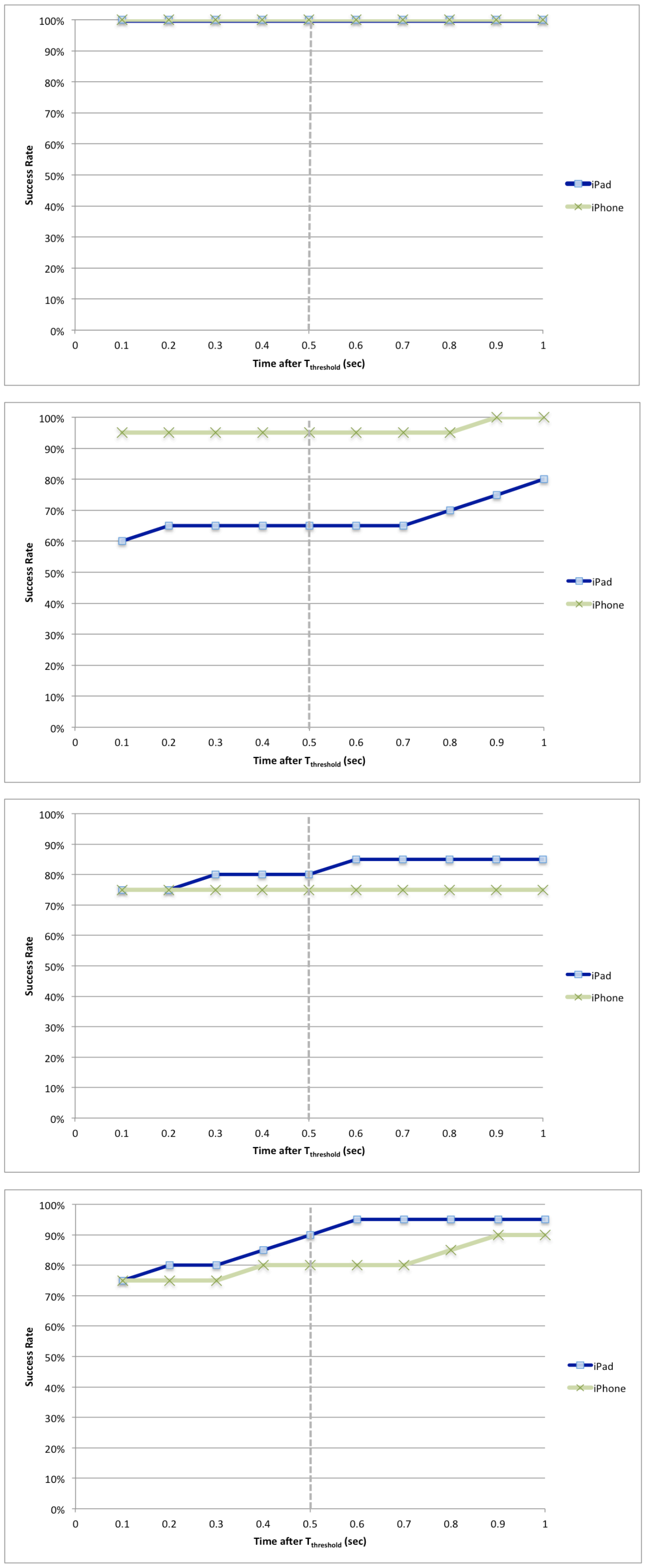

Figure 7. Cumulative success rate of human face orientation detection across the four tasks: 1) user, holding the device in portrait, changed posture from sitting to lying down, 2) user rotated device from portrait to landscape while lying down, 3) user changed posture from lying down to sitting, and 4) user rotated device from landscape to portrait while sitting.

DISCUSSION

Orientation Detection based on Partial Faces

iOS 5's face detection API requires two eyes and a mouth to be present in order to be detected. However, the front cameras frequently only have partial faces in their field of view. This mismatch contributes to the poor detection performance we observed in our user study. Partial face detection and face tracking techniques may help to solve the problem. Camera with even wider field of view should also help capture more portions of a face, improve the detection performance.

Another cause for poor detection performance is that users' often obscure the camera while hold the devices in landscape mode. Adding another camera on the border of the device, or placing one behind the display [15] should help improve the devices' view of users' faces.

Janicek, M. [15] filed a patent that integrate a camera in the center of the display may be a good solution to this problem.

Limitations

Vision-based techniques are sensitive to lighting conditions. We observed images from the user study for which strong backlight triggered the auto brightness adjustment of the camera. It caused users' faces to under expose and became un-recognizable. Also, face detection performance decreases as light level decreases, which would cause iRotate to fall back to using gravity-based rotation in extremely low-light conditions. Adding active infrared emitter with IR capable cameras to the mobile devices is one possible improvement, enabling face orientation detection to function even in complete darkness.

CONCLUSIONS AND FUTURE WORK

We have presented iRotate, a new, front camera-based approach to automatically rotate screens to match users' face orientation instead of gravity. It augments gravity-based rotation techniques by rotating to match users' face orientation, before falling back to gravity-based rotation. The approach can rotate screens correctly in different user postures and device orientation, and does not require explicit user input.

Our other contributions include a survey quantifying the frequency and the most likely cause of incorrect auto-rotation, as well as the awareness and usability of rotation lock. Our survey results show strong user needs for better auto-rotation. We also completed a 20-person study to assess the feasibility of our front camera-based approach, as well as evaluated the performance of face detection algorithms running on current generation of mobile devices.

Our study results show that current face detection algorithms perform poorly both in terms of speed and detection correctness. This is likely because the front camera typical captures images with part of users' faces instead of the entire face. We plan to collaborate with computer vision researchers and machine learning researcher to develop algorithms that can detect face orientation based on partial face features, such as hair and two eyebrows. In addition, we plan to address the second most common cause of incorrect auto-rotation, which is when devices are being used on flat surfaces. We plan to explore using gyroscope readings to trigger face detection using the front camera, and assess its feasibility.

REFERENCE

1. Balakrishnan, R., Baudel, T., Kurtenbach, G., and Fitzmaurice, G. The Rockin'Mouse: integral 3D manipulation on a plane. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1997), 311–318.

2. Ballagas, R., Rohs, M., and Sheridan, J.G. Sweep and point and shoot: phonecam-based interactions for large public displays. *CHI'05 extended abstracts on Human factors in computing systems*, ACM (2005), 1200–1203.

3. Bartlett, J.F. Rock 'n' Scroll is Here to Stay. *IEEE Computer Graphics and Applications*, May (2000), 40–45.

4. Blaskó, G., Beaver, W., Kamvar, M., and Feiner, S. Workplane-orientation sensing techniques for tablet PCs. *Proceedings of the 17th Annual ACM symposium on User Interface Software and Technology*, (2004).

5. Bradski, G.R., Clara, S., and Corporation, I. Computer Vision Face Tracking For Use in a Perceptual User Interface. *Intel Technology Journal 2*, 2 (1998), 12–21.

6. Fitzmaurice, G.W., Balakrishnan, R., Kurtenbach, G., and Buxton, B. An exploration into supporting artwork orientation in the user interface. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, ACM (1999), 167–174.

7. Forstall, S. and Blumenberg, C. Portrait-Landscape Rotation Heuristics for a Portable Multifunction Device. *US Patent 7978176, 12 July, 2011*

8. Hannuksela, J., Sangi, P., Turtinen, M., and Heikkilä, J. Face tracking for spatially aware mobile user interfaces. *Image and Signal Processing*, (2008), 405–412.

9. Hannuksela, J., Sangi, P., and Heikkilä, J. Vision-based motion estimation for interaction with mobile devices. *Computer Vision and Image Understanding 108*, 1-2 (2007), 188–195.

10. Hansen, T.R., Eriksson, E., and Lykke-Olesen, A. Mixed interaction space: designing for camera based interaction with mobile devices. *CHI'05 extended abstracts on Human factors in computing systems*, ACM (2005), 1933–1936.

11. Hinckley, K. and Song, H. Sensor synaesthesia: touch in motion, and motion in touch. *Proceedings of the 2011*

annual conference on Human factors in computing systems, ACM (2011), 801–810.

12. Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E. Sensing techniques for mobile interaction. *Proceedings of the 13th annual ACM symposium on User interface software and technology*, ACM (2000), 91–100.

13. Hinckley, K., Sinclair, M., Hanson, E., Szeliski, R., and Conway, M. The VideoMouse: a camera-based multi-degree-of-freedom input device. *Proceedings of the 12th annual ACM symposium on User interface software and technology*, ACM (1999), 103–112.

14. Ip, H.H.S., Hay, Y., and Tang, A.C.C. Body-Brush: a body-driven interface for visual aesthetics. *Proceedings of the tenth ACM international conference on Multimedia*, ACM (2002), 664–665.

15. Janicek, M. CAPTURING AN IMAGE WITH A CAMERA INTEGRATED IN AN ELECTRONIC DISPLAY. *US Patent App. 20,090/009,628*, 2007.

16. Lienhart, R., Kuranov, A., and Pisarevsky, V. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. *The German 25th Pattern Recognition Symposium (DAGM '03)*, (2003), 297–304.

17. Ording, B., Van Os, M., and Chaudhri, I. Screen Rotation Gestures on a Portable Multifunction Device. *US Patent 7978182, 12 July, 2011*

18. Schmidt, A., Beigl, M., and Gellersen, H.W. There is more to context than location. *Computers & Graphics 23*, 6 (1999), 893–901.

19. Segen, J. and Kumar, S. Human-Computer Interaction using Gesture Recognition and 3D Hand Tracking. *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, IEEE (1998), 188–192.

20. Sohn, M. and Lee, G. ISeeU: camera-based user interface for a handheld computer. *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, ACM (2005), 299–302.

21. Sparacino, F., Wren, C., Davenport, G., and Pentland, A. Augmented performance in dance and theater. *International Dance and Technology 99*, (1999), 25–28.

22. Sparacino, F. (Some) computer vision based interfaces for interactive art and entertainment installations. *INTER_FACE Body Boundaries, ed. Emanuele Quinz, Anomalie, n.2, Paris, France, Anomos*, Citeseer (2001).

23. Wang, J. and Canny, J. TinyMotion: camera phone based interaction methods. *CHI'06 extended abstracts on Human factors in computing systems*, ACM (2006), 339–344.

24. Zhang, L., Shi, Y., and Fan, M. UCam: direct manipulation using handheld camera for 3d gesture interaction. *Proceeding of the 16th ACM international conference on Multimedia*, ACM (2008), 801–804.