Camera Phone Based Motion Sensing: Interaction Techniques, Applications and Performance Study

Jingtao Wang[♦] *Shumin Zhai*[§] *John Canny*[♦]

[♦]Computer Science Division
UC Berkeley, 387 Soda Hall, Berkeley, CA, U.S.A
{jingtaow, jfc}@cs.berkeley.edu

[§]IBM Almaden Research Center
650 Harry Road, San Jose, CA, U.S.A.
zhai@almaden.ibm.com

ABSTRACT
This paper presents *TinyMotion*, a pure software approach for detecting a mobile phone user's hand movement in real time by analyzing image sequences captured by the built-in camera. We present the design and implementation of *TinyMotion* and several interactive applications based on *TinyMotion*. Through both an informal evaluation and a formal 17-participant user study, we found that 1. *TinyMotion* can detect camera movement reliably under most background and illumination conditions. 2. Target acquisition tasks based on *TinyMotion* follow Fitts' law and Fitts' law parameters can be used for *TinyMotion* based pointing performance measurement. 3. The users can use Vision Tilt-Text, a *TinyMotion* enabled input method, to enter sentences faster than MultiTap with a few minutes of practicing. 4. Using camera phone as a handwriting capture device and performing large vocabulary, multilingual real time handwriting recognition on the cell phone are feasible. 5. *TinyMotion* based gaming is enjoyable and immediately available for the current generation camera phones. We also report user experiences and problems with *TinyMotion* based interaction as resources for future design and development of mobile interfaces.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces; Input devices and strategies, Theory and methods

General terms: Design, Human Factors, Performance

Keywords: Input Techniques and Devices, Mobile Devices, Computer Vision, Mobile Phones, Camera Phones, Motion Estimation, Fitts' Law, Human Performance, Handwriting Recognition, Gesture Recognition

INTRODUCTION

Mobile phones have become an indispensable part of our daily life. Their compact form has the advantage of portability, but also imposes limitations on the interaction methods that can be used. While the computing and display capabilities of mobile phones have increased significantly in recent years, the input methods on phones largely remain button based. For basic voice functions, a button press based keypad is quite adequate. But mobile phones are rapidly moving beyond voice calls into domains such as gaming, web browsing, personal information management, location services, and image and video browsing. Many of these functions can greatly benefit from a usable analog input device. Mobile phones may take on even more computing functions in the future if higher performance user interfaces can be developed. We show in this paper that the built-in camera in mobile phones can be utilized as an input sensor enabling many types of user interactions.

Various technologies have been proposed and tested to improve interaction on mobile devices by enhancing expressiveness [11, 12, 18, 23, 24], or sensing contextual features of the surrounding environment [12]. Accelerometers [12, 18, 23, 24], touch sensors [11, 12] and proximity sensors [12] have been used. While some of these technologies may eventually make their way inside the phone, they are rarely seen in phones today.

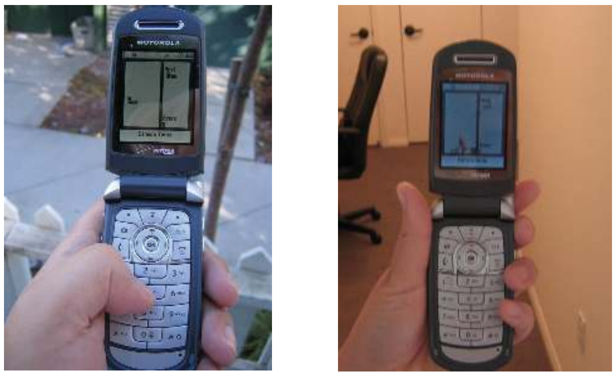
Figure 1: Using *TinyMotion* enabled applications out-doors (left) and in-doors (right)

On the other hand, camera phones are already popular and pervasive. The global cell phone shipment in 2005 was 795 million units, 57% of which (about 455 million units) were camera phones. It is predicted that 85% of the mobile phones will be camera phones by 2008 with a shipment of 800 million units [21].

We have developed a technique called *TinyMotion* (figure 1) for camera phones. *TinyMotion* detects the movements of a cell phone in real time by analyzing image sequences captured by its built-in camera. Typical movements that *TinyMotion* detects include horizontal and vertical movements, rotational movements and tilt movements. In contrast to earlier work, *TinyMotion* does not require additional sensors, special scenes or backgrounds. A key contribution of the paper is an experimental validation of the approach on a wide variety of background scenes, and a quantitative performance study of *TinyMotion* on standard target acquisition tasks and in real world applications.

RELATED WORK

Related work fall into three categories: emerging camera phone applications, new interaction techniques for mobile devices and computer vision based interaction systems.

Emerging Camera Phone Applications

Inspired by the success of CyberCode[19] from SONY, several researchers[20] and companies have designed customized 2D barcodes that can be recognized easily by camera phones. Most of these systems measure the size, position and angle of the barcode relative to the camera's optical axis, which can be used to infer the camera's 3D position relative to the barcode, and provide an alternative spatial input channel. SemaCode (http://semacode.org) is positioned as a general purpose tagging solution, SpotCode(a.k.a. ShotCode) has a feature that maps 2D bar codes to online URLs. On top of their barcode system Visual Codes [20] have also built UI widgets to achieve desktop-level interaction metaphors. Visual Codes is also used to manipulate objects on large public displays interactively [1].

Hansen and colleagues [4] proposed the idea of a "mixed interaction space" to augment camera phone interaction. Their method relies on camera imaging of a light uniform background with a circular marker. This image needs to be laid out on a suitable flat surface. 2D bar codes [1], Orthogonal axis tags [16], high gradient still objects [3] and human faces can all be used as markers to facilitate the tracking task. In contrast, *TinyMotion* provides general motion sensing from whatever scene the camera is pointed at, near or far.

If we make assumptions on the texture/gradient distribution of surrounding environments, other vision based approach such as Projection Shift Analysis [2] or gradient feature matching [9] could also provide real time motion detection on mobile devices. However, these approaches will fail when these strong assumptions do not hold. E.g., image projection based approach won't work on background with repeating patterns [2]. Many everyday backgrounds with less gradient, e.g. floors, carpets and the sky, will make gradient feature detection infeasible.

New Interaction Techniques for Mobile Devices

Previous work has proposed many compelling interaction techniques based on physical manipulation of a small screen device, including contact, pressure, tilt, motion and implicit biometric information. Specifically with regard to navigation, Rekimoto [18] used tilt input for navigating menus, maps, and 3-D scenes, and Harrison et al. [11] and Hinckley et al. [12] have used tilt for scrolling through documents and lists. Peephole Displays [25] explored the pen interactions on spatially aware displays on a PDA. Earlier before the current generation of phones and PDAs, Fitzmaurice et al. [5] explored spatially aware 'Palmtop VR' on a miniature handheld TV monitor.

Computer Vision in Interactive Systems

Considering the amount of information captured by human eyes, using computer vision in interactive systems has long been a popular topic [7]. Much previous research in this category covers multimodal interaction [6], gesture recognition, face tracking, body tracking [17] etc. There are also numerous systems that map certain types of user's movements, for example, body, gesture, finger, face, and mouth movements into computer inputs. Please refer to [6, 17], which include some extensive survey in the related directions, and [7] for some commonly used basic algorithms. However, most of those applications are built on powerful desktop computers in controlled lab environments.

THE TINYMOTION ALGORITHM

Computer vision techniques such as edge detection [9], region detection [7] and optical flow [13] can be used for motion sensing. Ballagas et al [1] had implemented an optical flow based interaction method - "sweep" on camera phones. However, optical flow [13] is a gradient based approach and it uses local gradient information and the assumption that the brightness pattern varies smoothly to detect a dense motion field with vectors at each pixel. Due to the additional assumptions on gradient distribution and the smoothness of illumination, they are usually less robust than direct methods based on correlation or image difference. The latter are used in optical mice, video codecs etc, and we follow suit. *TinyMotion* has used both image differencing and correlation of blocks [8, 14] for motion estimation.

The *TinyMotion* algorithm consists of four major steps: 1. Color space conversion. 2. Grid sampling. 3. Motion estimation. 4. Post processing. All of these steps are realized efficiently by integer only operations.

To use *TinyMotion*, the camera is set in preview mode, capturing color images at a resolution of 176x112 pixels, at a rate of 12 frames/sec[1].

Color Space Conversion

After a captured image arrives, we use a bit shifting method (equation 2, an arithmetic approximation of equation 1) to convert the 24-bit RGB color (20 effective bits in each pixel for our specific camera phone used) to an 8-bit gray scale image.

[1] Without displaying the captured image and additional computation, the camera phones in our experiments can capture images at the maximal rate of 15.2 frames/sec.

$$Y = 0.299 * R + 0.587G + 0.114B \quad (1)$$

$$Y = (R >> 2) + (G >> 1) + (G >> 3) + (B >> 3) \quad (2)$$

Grid Sampling
Grid Sampling, a common multi-resolution sampling technique [14], is then applied on the gray scale image to reduce the computation complexity and memory bandwidth for the follow-up calculations. We use 8x8 sampling window in our current implementation after much experimentation.

Motion Estimation
The motion estimation technique we use is similar to those commonly used by video encoders (MPEG2, MPEG4 etc). We denote the result of grid sampling as a macro-block (MB) and apply Full-search Block Matching algorithm (FBMA)[14, 8] on temporally adjacent fames.

Let I_k represent the current frame and I_{k-1} represent the previous frame. In any frame I, I(x,y) is the pixel value at location (x, y). For FBMA, the MB in current frame I_k is shifted and compared with corresponding pixels in previous frame I_{k-1}. The shifting range is represented as Rx and Ry respectively. In our current implementation, Rx = (-3, 3), Ry = (-3, 3). Common distance measurements include Mean Square Error (MSE, equation 3[14]), Sum of Absolute Difference (SAD) and Cross-Correlation Function (CCF). After block matching the motion vector $\overrightarrow{MV}$ is chose as the corresponding block shifting distance (equation 4).

$$MSE(dx, dy) = \frac{1}{MN} \sum_{m=x}^{x+M-1} \sum_{n=y}^{y+N-1} \left[I_k(m,n) - I_{k-1}(m+dx, n+dy) \right]^2 \quad (3)$$

$$\overrightarrow{MV} = (MV_x, MV_y) = \min_{(dx,dy)\in R^2} MSE(dx, dy) \quad (4)$$

The motion vector $\overrightarrow{MV} = (MV_x, MV_y)$ represents the displacement of the block with the best result for the distance criterion after the search procedure is finished. According to the output of the motion estimation, tilting left is equivalent to moving the phone left, tilting the upper part of the phone towards the user is equivalent to moving the phone upwards, and so on. To detect camera rotation, we split each global MB into 2x2=4 sub MBs and estimate their relative motions respectively.

Post Processing
The relative movements detected in the motion estimation step are distance changes in the x and y directions. These relative changes are also accumulated to provide an absolute measurement from a starting position.

In the current implementation, *TinyMotion* generates 12 movement estimations per second, and takes 19 – 22 ms to process each image frame on a Motorola v710 phone. The memory needed is around 300KB.

IMPLEMENTATION
Our primary implementation platform is the Motorola v710 (a CDMA Phone from Verizon Wireless), a common off-the-shelf camera phone at the time our implementation. The v710 has an ARM9 processor, 4M RAM, 176x220 pixel color display. Our application is written in C++ for BREW (the Binary Runtime Environment for Wireless, http://brew.qualcomm.com) 2.11. We use Realview ARM Compiler 1.2 for BREW to cross-compile the target application. BREW is an efficient binary format that can be downloaded over the air, so there is a rapid distribution path for commercial applications built using *TinyMotion*. We believe *TinyMotion* can also be ported easily to other platforms such as Windows Mobile and Symbian.

To test the efficacy of *TinyMotion* as an input control sensor, we wrote four applications (Motion Menu, Vision TiltText, Image/Map Viewer, Mobile Gesture) and three games (Camera Tetris, Camera Snake and Camera BreakOut). All these prototypes can be operated by moving and tilting the camera phone. Figure 2 shows some screen shots of the *TinyMotion*-enabled prototype applications. Not including the recognizer used in Mobile Gesture, the current *TinyMotion* package includes a total of 23,271 lines of source code in C++. We now discuss the Mobile Gesture and Vision TiltText applications in greater detail.

Mobile Gesture
The Mobile Gesture application was inspired by the idea of using the camera sensor on the cell phone as a stylus for handwriting recognition and gesture based command and control. In the current implementation of Mobile Gesture, the user presses the "OK" button on the phone to trigger the "pen down" operation on the phone. Instead of restricting gesture/handwriting to be single stroke or setting a timeout threshold to start the recognition, the user presses the POUND ("#") key to signal the end of a character.

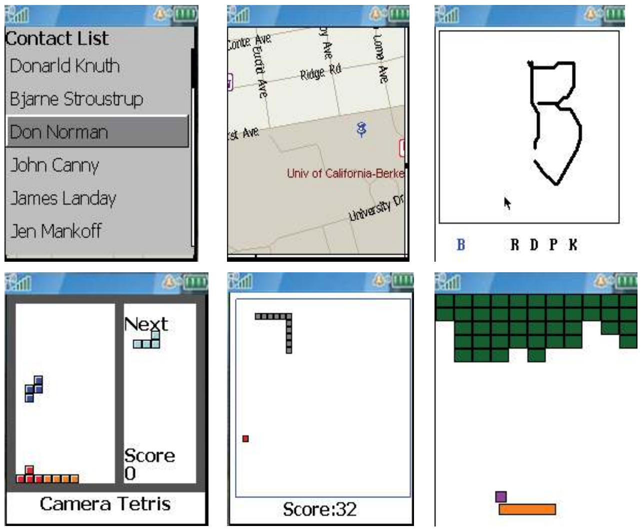

Figure 2: Sample *TinyMotion* applications and games. From left to right, top to bottom – Motion Menu, Image/Map Viewer, Mobile Gesture, Camera Tetris, Camera Snake and Camera BreakOut

The recognizer used by the Mobile Gesture application is a commercial product designed by one of the authors. The original recognizer was designed for handheld devices running Palm Pilot, Windows CE or Linux etc. It supports multilingual input of western character and East Asian double byte characters (e.g. Chinese, Japanese characters). By default, we use a recognition library which supports all the English characters, punctuation symbols and around 8000 Chinese and Japanese characters (6763 simplified Chinese characters defined by the GB2312 national standard and around 1300 Hiragana, Katakana and Kanji characters in Japanese). The size of the recognizer is around 800KB including both the code and the recognition library. If we remove the support for Asian double byte characters, the size of the recognizer and related library can be reduced to around 350kb. On the Motorola v710 phone, it takes 15-20ms to recognize a handwritten Roman character, and 35-40ms to recognize one of the 8000 supported double byte characters. As a reference, it takes the same recognizer around 700ms to recognize the same double byte character on a Palm V PDA with 2 Mb memories, which implies an off-the-shelf cell phone in year 2005 is about 20 times faster than a common PDA in the year 1999 for this specific recognition task. We made one modification on the recognizer after porting it to BREW by adding a four-pixel wide smoothing window filter on the handwriting traces before starting the actual recognition process. This is designed to reduce the hand shaking noise captured by *TinyMotion*. The handwriting traces displayed on the user' screen are not smoothed.

Vision TiltText

We use the following configuration in our *Vision TiltText* text input method, which is a remake of the accelerometer based mobile input method by Wigdor and colleagues [24]. To input character A, a user need to press keypad button "2", hold it, tilt or move the phone to the left, release button "2". To input character B, press and release button "2" without movement, to input character C, press button "2", hold it and tilt or move the phone to the right, then release the button. This definition is based on the convention that the alphabet characters displayed on telephone button "2" is ordered as 'A','B','C' from left to right respectively. To input numeric characters, one presses the corresponding numeric key, move the phone up, then release it. To input the fourth character 'S' or 'Z' on button '7' and '9', the user can press the related button, move down, then release. To avoid noisy movement generated by hand shaking, we set a movement threshold for all the characters that need tilting to enter. When the movement in one direction exceeds the corresponding threshold, the phone will vibrate for 70ms to signal that the input state had changed so the button can be safely released.

FIRST INFORMAL EVALUATION

We evaluated the reliability of *TinyMotion* by two methods. First, we benchmarked the detection rate of camera movements in four typical conditions and four different directions. To measure each direction 50 shift action and 50 tilt actions were performed. In total, 1600 actions were recorded. A shift action required at least a half inch of movement, while a tilt action had to exceed 15 degrees. If the accumulated movements value in a certain direction exceeded the threshold value 5, our system will output that direction as the detected movement direction.

The summarized detection rates in each condition are listed in table 1. Most of the errors in the outdoor direct sunshine condition were caused by unexpected objects (mostly vehicles or people) moving into/out of the camera view during the testing process.

	Left	Right	Up	Down
Outdoor direct sunshine	97%	100%	96%	97%
Outdoor in the shadow	100%	99%	99%	100%
In-door ambient light	100%	100%	100%	100%
In-door fluorescent lamp	100%	100%	99%	100%

Table 1: Movement benchmarking results in four typical environments (shifting and tilting movements in the same direction are not differentiated)

We also conducted an informal usability test by distributing camera phones installed with *TinyMotion*-enabled applications/games to 13 users, most of them students or faculty members in a local university. We asked them to play with the Motion Menu application, the Camera Tetris game and the Camera Snake game (Some applications such as Mobile Gesture and Camera BreakOut were not ready at the time of the informal evaluation) and encouraged them to challenge *TinyMotion* in any background and illumination conditions that they could think of or had access to.

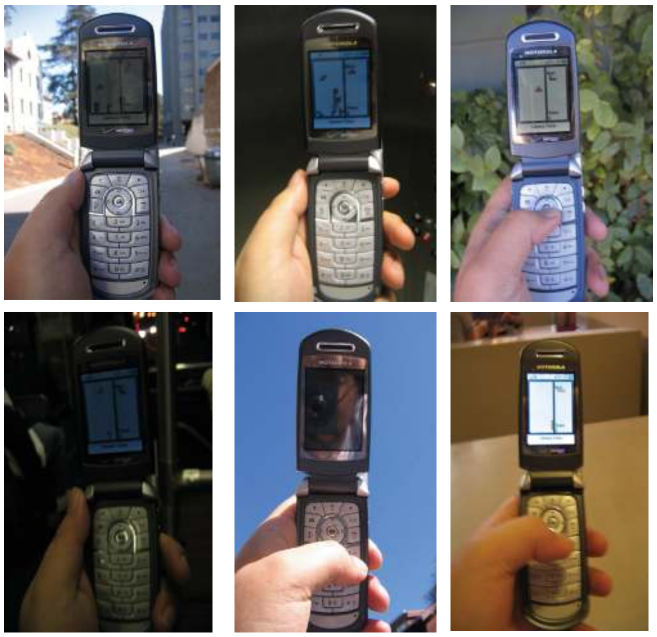

Figure 3: Environment/Backgrounds in which *TinyMotion* work properly

All of the users reported success against backgrounds such as an outdoor building, piles of garbage, different types of floors indoor and outdoor, grass in a garden, cloth, and a bus stop at night, areas with low illumination or colored illumination, different areas in pubs, etc. Most were surprised to learn that the motion sensing was based on camera input. One participant was shocked when he found that *TinyMotion* still worked when he pointed the camera at a blue sky and moved the phone (even motion of smooth gradient images can be detected). Figure 3 shows some difficult situations where traditional edge detection based methods may fail but *TinyMotion* can still work.

The environments in which *TinyMotion* won't work include completely dark rooms, extremely uniform background without any pattern (e.g. the glass surface when an LCD monitor is turned off) and pointing the camera to the outside of a window in a moving vehicle.

The participants in our informal study were clearly amazed with *TinyMotion* and interested in its use. Comments include *"Cool, I didn't expect the tracking can work that well." "Using this (motion menu) makes [operating] cell phones a lot more fun" "it will be an ideal method to play the monkey ball game on a cell phone"*

One user quickly realized that instead of moving the camera phone directly, he can put his other hand in front of the camera lens and control the *TinyMotion* games by moving that hand. Another user initially felt *TinyMotion* was not very sensitive, only to find that his extended index finger covered the camera lens.

FORMAL EVALUATION

Although the results of the first informal user study were very encouraging, a formal study was necessary for understanding the capabilities and limitations of *TinyMotion* as an input sensing mechanism. We had two basic goals for the formal study. One was to quantify human performance using *TinyMotion* as a basic input control sensor. In particular we measured the performance of pointing, menu selection, and text entry by tap-tilt action. The second goal was to evaluate the scope of applications that can be built on the *TinyMotion* sensor, for example using *TinyMotion* to play games and do handwriting / gesture recognition.

Experimental Design

The experiment consisted of six parts:

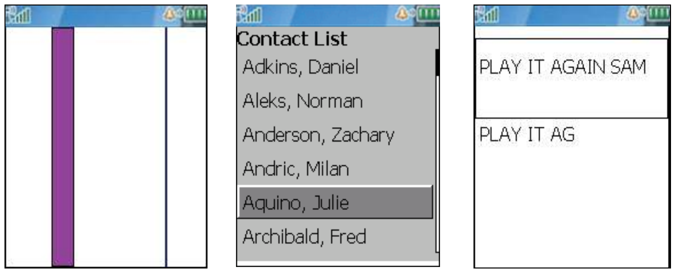

Figure 4: From left to right, screen shots of the target acquisition task, menu selection task and text input task.

Overview. In this session we gave a brief overview of the *TinyMotion* project, and demonstrated some of the *TinyMotion* applications to the participant. We also answered their questions and let them play with *TinyMotion* applications freely.

Target Acquisition/Pointing. This session was designed to quantify the human performance of the *TinyMotion* based pointing tasks. The section started with a warm up practice session to allow the participants to become familiar with the pointing task. Pressing UP button started a trial from an information screen indicating the number of trials left to be completed. Each trial involved moving the cell phone UP, DOWN, LEFT or RIGHT to drive the on screen cursor (a slim line) from an initial position to the target and then pressing OK button (Figure 4, left). If the user hit the target, the target acquisition screen disappeared and the information screen returned. If the user missed the target, the cursor returned to the initial position and the trial repeated until the user successfully hit the target. The users were encouraged to hit the target as fast as possible and as accurately as possible during the target acquisition stage, but could rest as long as needed when the information screen was displayed. We encouraged the users to practice as long as they wanted before the actual test, most of the users practiced for 3 to 5 minutes.

There were 4 different target sizes (20, 30, 40, 50 pixels), 4 different distances (30, 50, 70, 90 pixels) and 4 different movement directions (left, right, up, down) in this task. Each participant completed 160 randomized trials.

Menu Selection. In each trial of this task, a participant was required to select a target name from a contact list. After reading the target name from an information screen, the participant could press the STAR ("*") button to switch to the contact list and start the actual trial. The contact list included 30 alphabetically sorted names and the cell phone could display 6 names per screen. After highlighting the intended name, the user can press the "OK" button to complete the trial and switch back to the information screen (Figure 4, middle).

There were three conditions in this task: cursor button based selection, *TinyMotion* based selection, and *TinyMotion* based selection with tactile feedback. In the tactile feedback condition every time when the highlighted menu item changed as a result of moving the phone, the phone vibrated for around 100ms, providing a non-visual cue to the user about her progress on the menu item movements. We added this condition to check the potential influences of tactile feedback on menu selection.

Each participant was asked to complete 16 selections in each condition. The order of the three conditions was randomized to counter balance the learning effects.

Text Input. In this task, we compared the performance of the most popular mobile text entry method – MultiTap with our Vision TiltText input method. We followed configurations similar to those used in Wigdor et al's original TiltText study [24]. The short phrases of text were selected from MacKenzie's text entry test phrase set (www.yorku.ca/mack/phrases2.txt). The timeout for the MultiTap method was 2 seconds. Due to time constraint of our study, each participant entered only 8 sentences with each input method. Note that in studies like [24], each participant entered at least 320 sentences for each input method tested. As a result, our study was not intended to reveal the learning curve and eventual performance of the input methods tested. Instead, we only measured users' initial performance without much practice.

This task started with a warm up practice session, the users could practice with the two methods tested as long as they wanted. Most of them choose to practice for 2-5 minutes before the actual test. The order of the two methods tested was randomized.

More Complex Applications. After completing the three basic quantitative performance tasks described above, we asked the participants to play with the games we created (Camera Tetris, Camera BreakOut, and Camera Snake) and the handwriting recognition application (Mobile Gesture).

Figure 5: Some sample pictures taken from our user study

After demonstrating the games and the handwriting recognition application to the users, we let the users play with these applications by themselves. They were encouraged to play the games as long as they wanted and enter at least 3 different characters/gestures in our handwriting recognition application.

Collecting qualitative feedback. We conducted a final survey immediately after a user completed all the tasks. In the survey the user completed a questionnaire and commented on the applications they tested and on the idea of *TinyMotion* in general.

To simulate the real world situations of cell phone usage, we did not control the environment used for the study. The participants were encouraged to choose their desired locations to complete the study. Most of the studies were completed in the participants' own chair or at a public discussion area in a lab. Figure 5 shows some of the actual environments used during the study.

Test Participants

17 people participated in our study. 15 of them were undergraduate or graduate students in a university, the other two were staff members of the university. 6 of the participants were female and 11 male. Five of them owned a PDA and all of them owned a cell phone at the time of the study. 12 of the 17 cell phones were camera phones. Four of the participants sent text messages daily, six weekly, three monthly and four never sent text messages. Interestingly, no user in our study use the camera function of their cell phone on a daily basis, three of them use the camera function weekly, four monthly, four yearly and one of them never uses the camera function.

Two participants didn't complete the menu selection task and one participant didn't complete the text input task due to time constraints. One of the studies was interrupted by a false fire alarm for around 25 minutes. All of the participants completed the target acquisition task, played with all the applications we created and completed our survey and questionnaire.

EVALUATION RESULTS

Target Acquisition/Pointing

2842 target acquisition trials were recorded. Despite the low sampling rate of *TinyMotion* and the novel experience of using it, 2720 of the pointing trials were successful, resulting in an error rate of 4.3%, which is common in Fitts' law studies. This means that it is safe to say that it is already possible to use *TinyMotion* as a pointing control sensor.

While there is a vast literature showing hand movements involving various joints and muscle groups follow Fitts' law[4], it is still informative to test whether Fitts' law holds given the particular way a *TinyMotion* instrumented cell phone is held and operated, and the current sampling rate limitation of the cameras in phones.

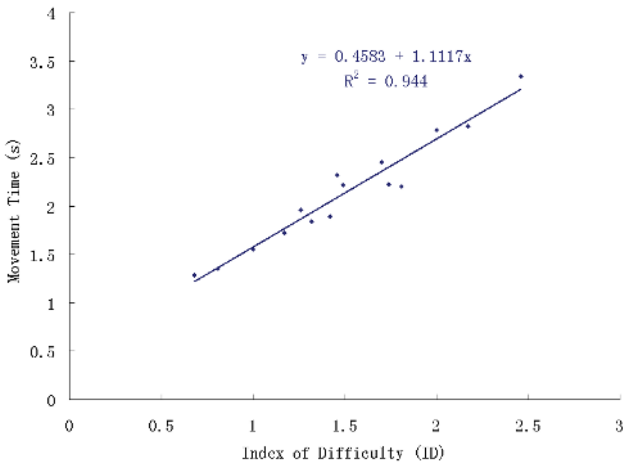

Figure 6: Scatter-plot of the Movement Time (MT) vs. the Fitts' Law Index of Difficulty (ID) for the overall target acquisition task.

Linear regression between movement time (MT) and Fitts' index of difficulty (ID) shows (Figure 6):

$$MT = 0.4583 + 1.1117 \log_2\left(\frac{A}{W} + 1\right) \quad \text{(sec)} \qquad (5)$$

In equation 5, A is the target distance and W is the target size. While the empirical relationship between movement time (*MT*) and index of difficulty (*ID* = log (A/W + 1)) followed Fitts' law quite well (with $r^2 = 0.94$, see Figure 6), both of the two Fitts' law parameters (time constant a = 0.458 sec and information transmission rate 1/b = 1/1.1117 = 0.9 bits/sec) indicated relatively low performance of pointing. This is not surprising given the low sampling rate of the camera (12 frames per second as opposed to 40+ frames per second in a typical computer mouse). However since we now know *TinyMotion* based pointing follows Fitts' law, these parameters can serve as an informative benchmark for future improvement in hardware (e.g. image frame rate, image capturing quality) or the software (detection algorithms and related parameters).

An interesting finding is the difference in manipulation direction of *TinyMotion* instrumented mobile phones. The error rates for four different moving directions (left, right, down, up) were 3.3%, 2.6%, 5.4% and 5.9% respectively. Analysis of variance showed that there was a significant main effect (*F(1, 32)= 4.15, p<0.05*) between horizontal movements and vertical movements in error rate. There was no significant main effect (F(1,32)= 1.15, p=0.29) between horizontal and vertical movements in movement time, although on average it took longer to accomplish vertical target acquisitions than horizontal acquisitions under the same ID value, particularly when ID was high (Figure 7). Participants also subjectively felt that it was more difficult to acquire vertical targets. This result could have implications to *TinyMotion* application design.

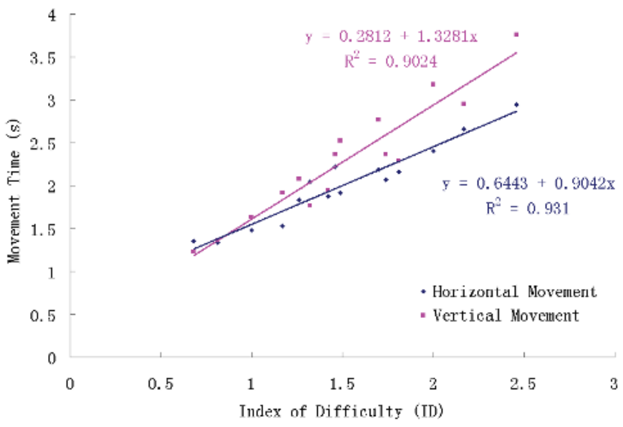

Figure 7: Scatter-plot of the Movement Time (MT) vs. the Fitts' Law Index of Difficulty (ID), separately grouped by horizontal and vertical directions.

Menu Selection

The menu selection results show that it is also possible to use *TinyMotion* as a menu selection mechanism. Of the 755 menu selection actions recorded in our study, 720 of them were successful selections. The overall error rate was 4.6%. The error rates for the three experimental conditions - cursor key selection, *TinyMotion* selection and *TinyMotion* selection with tactile feedback (referred as *TinyForce* later) were 3.2%, 4.8%, and 5.9% respectively. Analysis of variance did not show a significant difference among these error rates. As shown in Figure 8, the average menu selection time was 3.57s, 5.92s, 4.97s for the Cursor Key, *TinyMotion* and *TinyForce* condition respectively. Analysis of variance results showed that there was a significant difference (F(2, 42) = 8.44, p<0.001) in completion time. Pair-wise mean comparison (t-tests) showed that completion time between the cursor key and *TinyMotion* methods, and the cursor key and *TinyForce* methods were significantly different from each other (p<0.01), but not between the *TinyMotion* and *TinyForce* conditions. While on average the tactile feedback did reduce menu selection time, the difference was not significant due to the large performance variance.

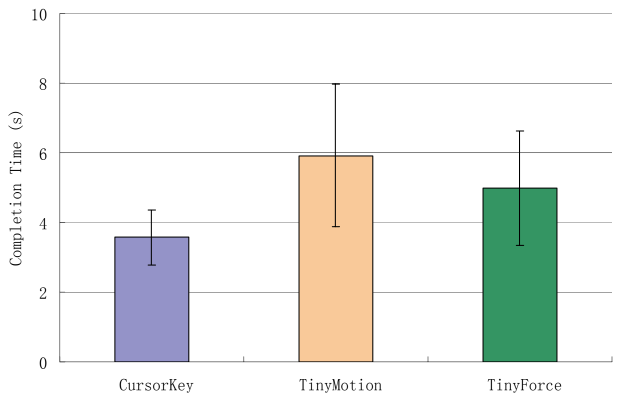

Figure 8: Menu selection time from the experiment

There were several factors that lead to the low observed performance of *TinyMotion*-based menu selection techniques. First, the contact list used in our study was relatively long (30 names or 5 screens). Although the names were sorted alphabetically, it was still hard for the participants to estimate the approximate location of the name in the contact list given that the name distribution was unlikely to be uniform. As a result, if the desired item was not on the first screen, the participants had to scroll down the contact list slowly to locate the name, which proved to be a difficult task based on our observation. Second, our current Motion Menu was based on the menu behavior provided by the BREW platform, when the highlighted item reached the bottom (the sixth) row and a move down command was received, all the currently on-screen items would move up one row and a new item appeared on the sixth row and was highlighted. This feature was not a problem for cursor key based selection, because the user can simply pay only attention to the sixth row, while holding the phone steadily. However, this feature became troublesome when the users use camera movement for menu selection, most users felt it was difficult to keep track of the last row while keep the phone moving.

Text Input

In total 6150 characters were entered (including editing characters) in this experiment.

Consistent with Wigdor et al's finding [24], the overall speed of Vision TiltText was higher than that of MultiTap (Figure 9) and the error rate of Vision TiltText (13.7%) was much higher than for MultiTap (4.0%) (Figure 10). The difference in error rate was statistically significant (F(1, 30) = 44.36, $p < 0.001$), but the difference in input speed was not. Overall, the results of vision based TiltText were similar to or slightly better than the accelerometer based TiltText reported by Wigdor et al [24] at the same (initial) learning stage. This shows that as an input sensor TinyMotion is at least as effective as an accelerometer for tap-tilt action based text input.

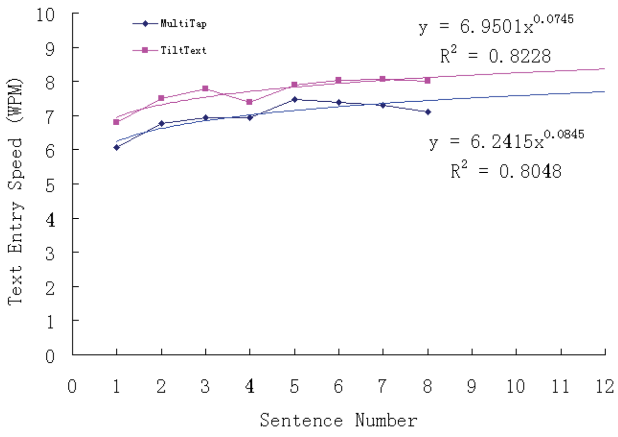

Figure 9: Entry speed (wpm) by technique and sentence for the entire experiment.

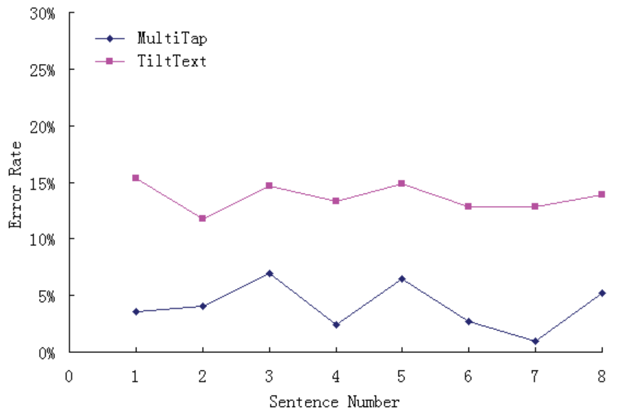

Figure 10: Error rate (%) by technique and sentence for the entire experiment.

Nearly all the users believed that Vision TiltText is an efficient text entry method (average rating 4.2, SD=0.8, 1-5 scale, 5 means most efficient, 3 means neutral, 1 means least efficient, no one rated Vision TiltText "less efficient" or "least efficient") and is easy to learn (average rating 4.3, SD = 0.7, 1-5 scale, 5 means extremely easy to learn, 3 means neutral, 1 means extremely difficult to learn, no one rated Vision TiltText difficult or extremely difficult to learn). 13 users commented explicitly that they would like to use Vision TiltText in their daily life immediately.

Subjectively, participants liked the vision based tilt-text over multi-tap. "[For Vision TiltText,] doing a gesture can really speed things up, it was very intuitive." "[Vision] TiltText [is] faster once you learn it, fewer clicks." "[Vision TiltText based] text entry is truly useful because the multitap for names is annoying, the T9 helps with words, but not names."

More Complex Applications

The participants were excited about their experience of using camera phone movement for gaming. They played the provided games for around 5 – 12 minutes. One user rated use of motion sensing for games as "extremely useful", 10 rated "useful", 6 rated "neutral". No one rated these games "not useful" or "extremely not useful". As a comparison, 9 participants reported that they never played games on their cell phones before, 4 played games yearly and 4 played monthly. In the closing questions, 7 users commented explicitly that these *TinyMotion* games were very fun and they would like to play these games on their own phones frequently.

The user study also revealed several usability problems related with gaming. A lot of participants pointed out that the "conceptual model" or "control" is inconsistent across the current games. e.g. in the Camera Tetris game, when a user moves the cell phone to the left, the block under control will move to the right and vice versa (assuming we are moving the frame, or the background of the game, the block is still). On the other hand, in the Camera BreakOut game, moving the camera phone to the left will move the paddle to the left (assuming we are moving the paddle, the background is still). Around two third of the users believe the "moving background" model is more intuitive while the other one third of users believe the "moving foreground object" model is more intuitive. All of them agree that such game settings should be consistent across all games and it is better to let the user decide which control model to use.

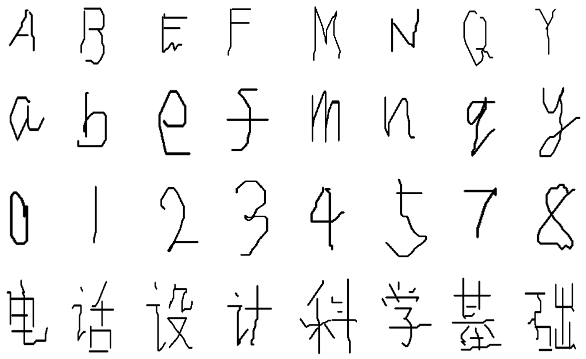

Figure 11: Some handwriting samples collected by mobile gesture that had been successfully recognized. The last row is a list of four Chinese words (with two Chinese characters in each word) meaning telephone, design, science, and foundation respectively. No smoothing operation was applied on any of the handwriting samples.

The users also had diverse opinions on the sensitivity of the game control; three users felt the games control should be more sensitive than the current setting while other two users want the game control to be less sensitive, which seem to suggest games controlled by arm/wrist movements should provide adjustable sensitivity setting for each game (the current game control sensitivity was decided by the authors subjectively).

Most of the users were surprised by the ability of using camera phones to do handwriting and receiving recognition results in real time. After a brief demonstration on how to use the mobile gesture application, all the participants successfully entered some alphabetical characters/numeric characters after 1-2 minutes of practice (some handwriting samples shown in Figure 11). One of the test users, whose native language is Chinese, even tested for more than ten Chinese characters after knowing that the recognizer also supports Chinese and Japanese characters. Based on our observation, it took a participant around 5 – 15 seconds to write an alphabet character and around 15 – 30 seconds to write a Chinese character by using the mobile gesture application. Although from text input speed point of view, Mobile Gesture was obviously slower than most of the keypad input method, most of the users felt really excited when their handwritings (sometimes distorted) got recognized by the cell phone correctly. Indeed most of the users did not believe Mobile Gesture was an efficient text input method (average rating 3.2, SD = 1.3, 1-5 scale, 5 means most efficient, 3 means neutral, 1 means least efficient). But they also thought Mobile Gesture was intuitive to use (average rating 4.0, SD = 1.0). 4 users suggested the ideas of using Mobile Gesture for sentence level input rather than character level input, i.e. predefine some frequent sentences by arm gestures and use Mobile Gesture to trigger those sentences by directly writing the corresponding gesture.

One participant suggested the idea of using Mobile Gesture for authentication tasks. i.e. distinguishing whether a user using the phone is the actual owner by measuring the movement characteristics of predefined trajectories.

DISCUSSIONS AND FUTURE WORK

Battery Life

One major concern related with the popularization of *TinyMotion* could be the battery life. We measured the battery life of *TinyMotion* in two conditions: a power saving situation and an exhaustive usage situation. For the power saving condition, we tested *TinyMotion* by continuously running the Camera Tetris game with the backlight and the vibration function turned off. Our Motorola v710 cell phone ran 8 hours and 7 minutes after a full charge.

For the exhaustive usage situation, we measured battery life while *TinyMotion* was constantly running in the background, with the backlight of the camera phone always on, and the vibration function activated around 40% of the total time, and keypad frequently used. In these conditions the same cell phone lasted around 3 hours and 20 minutes to 3 hours and 50 minutes. In more moderate use (less than 5% continuous use each time) the phone battery should last for several days. On most modern phones, the biggest power drain results from screen backlight, and radio use.

TinyMotion vs. Accelerometer

The most relevant movement sensing technology for mobile devices might be accelerometers. Much mobile device related research [12, 17, 23, 24] uses one or more accelerometers to sense device movements, for fall protection, user context detection [12], UI navigation and text input [17, 23, 24]. Due to its small size and relatively low manufacturing cost, we feel accelerometers also have the potential to become pervasive on mobile devices. Hence we feel a fair comparison of *TinyMotion* and accelerometer will be helpful to mobile interface designers.

Accelerometers do not require the computing power of host mobile devices in the sensing process and do not depend on illumination condition or view background. In contrast, *TinyMotion* requires certain amount of computing power from the device to generate movement estimates and may not work well in extreme illumination and background conditions.

The working mechanisms in accelerometers and *TinyMotion* are very different. The piezoelectric or MEMS sensors in accelerometers are actually sensing *movement accelerations* and the magnitude of *gravitational field*. In contrast, *TinyMotion* is detecting deviation/shifting of backgrounds. Double integral operations are needed to estimate position from the raw output of acceleration sensors, which cause accumulate drift errors and make distance estimation less reliable than acceleration. Similarly, acceleration estimations from *TinyMotion*, derived by differential operations, are less reliable than the original deviation estimations.

Future Work

There are many interesting questions worth exploring in the near future. For one example, we feel it might be important to carefully consider the use a "clutch" that can engage and disengage motion sensing from screen action. For another example, the traditional one dimensional linear menu is obviously not the most effective method for camera movement based navigation. We are exploring the possibility of applying a marking menu [15] approach using gesture angles rather than movement distance for menu selection. We feel that the Vision *TiltText* input method is quite promising and can be further improved, for example, by adding visual feedback to guide the user and speed up the error correction process. Many more applications, such as those involving panning and zooming, should also be explored, particularly in the context of domain specific applications.

CONCLUSION

In this paper, we proposed a method called *TinyMotion* that measures cell phone movements in real time by analyzing images captured by the built-in camera. Through an informal evaluation and a formal 17-participant user study we found that 1. *TinyMotion* can detect camera movement reliably under most background and illumination conditions. 2. Task acquisition tasks based on *TinyMotion* follows Fitts' law and the Fitts' law parameters can be used to benchmark *TinyMotion* based pointing tasks. 3. The users can use Vision TiltText, a *TinyMotion* enabled input method, to input sentences faster than MultiTap with a few minutes of practice. 4. Using camera phone as a handwriting capture device and performing large vocabulary, multilingual real time handwriting recognition on the cell phone are feasible. 5. *TinyMotio*n based gaming is fun and immediately available for the current generation camera phones. Overall, we conclude that by adding software the use of the built-in camera in phones can go beyond taking pictures into the interaction domain. It is already possible to use the motion sensing result for basic input actions such as pointing, menu selection and text input, and the performance of these tasks can be further improved as hardware performance (in particular the camera frame rate) in phones advances. We showed that it is also possible to build higher level interactive applications, such as gaming and gesture recognition, based on our sensing method and we expect broader and more creative use of camera motion in the future.

TinyMotion is a pure software project. We choose not to make any hardware changes to the standard phone so results of our research are immediately available for download and use. *TinyMotion* is open source software released under BSD license. The current implementation can be downloaded freely from *URL http://guir.berkeley.edu/tinymotion*.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Jeff Heer, Jonathan Chung, Ryan Aipperspach, Tye Rattenbury, Ana Chang, Tom Duan, Jono Hey, David Nyguen, Kelly Li, John Suarez, Matthew Kam and Irene Chien for their contributions to this work. We also thank Qualcomm Corp. for their support in funding and equipment.

REFERENCES

1. Ballagas, R., Rohs, M. el al, Sweep and Point & Shoot: Phonecam-Based Interactions for Large Public Displays. In *Ext. Abstracts of CHI '05,* April 2005
2. Drab, S., Artner, N., Motion Detection as Interaction Technique for Games & Applications on Mobile Devices, In *Ext. Abstracts of PERVASIVE '05: (PERMID)*, Munich, Germany.
3. EyeMobile, http://www.eyemobile.com
4. Fitts, P.M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47, 381-391, 1954.
5. Fitzmaurice, G. W, Zhai, S., Chignell, M., Virtual Reality for Palmtop Computers. In *ACM TOIS*, pp. 197–218. July 1993.
6. Forsyth, D., and Ponce, J. Computer Vision: A Modern Approach. Prentice Hall, Upper Saddle River, NJ, 2003.
7. Freeman, W.T., et. al., Computer Vision for Interactive Computer Graphics. In *IEEE Computer Graphics and Applications*, May-June, pp. 42-53, 1998,
8. Furht, B., Greenberg, J., Westwater, R., Motion Estimation Algorithm for Video Compression, *Kluwer Academic Publishers*, Boston/Dordrecht/London, 1997
9. Hannuksela, J., Sangi, P., and Heikkila J., A Vision-Based Approach for Controlling User Interfaces of Mobile Devices, In *Proc. of IEEE Workshop on Vision for Human-Computer Interaction (V4HCI)*, 2005
10. Hansen, T., et al., Mixed Interaction Space - Designing for Camera Based Interaction with Mobile Devices, In *Ext. Abstracts of CHI '05*, April 2005
11. Harrison, B. L., Fishkin, K., A. et al, Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In *Proc. of CHI'98,* pp. 17–24, April 1998.
12. Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E., Sensing Techniques for Mobile Interaction. In *Proc. of UIST 2000,* pp. 91–100.
13. Horn, B., and Schunck, B.. Determining optical flow. AI Memo 572. Massachusetts Institue of Technology, 1980.
14. Kuhn, P., Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation, Kluwer Academic Publishers.
15. Kurtenbach, G., and Buxton, W. User learning and performance with marking menus. In *Proc. of CHI'94*, pp. 258-264, April 1994
16. Moehring, M., Lessig C., and Bimber O. Optical tracking and video see-through AR on consumer cell phones. In *Proc. of Workshop on Virtual and Augmented Reality of the GI-Fachgruppe AR/VR*, pp. 193-204. 2004
17. Moeslund, T.B., and Granum, E. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding* 18, pp. 231–268, 2001.
18. Rekimoto, J., Tilting Operations for Small Screen Interfaces. In *Proceedings of.* UIST 1996, pp. 167–168.
19. Rekimoto, J., Ayatsuka, Y., CyberCode: designing augmented reality environments with visual tags, In *Proc. of DARE 2000*, pp. 1-10.
20. Rohs, M., Zweifel, P., A Conceptual Framework for Camera Phone-based Interaction Techniques, *In Proc. of PERVASIVE 2005*, Munich, Germany, May 2005
21. The Development of Camera Phone Module Industry, 2005-2006, http://www.okokok.com.cn/Abroad/ Abroad_show.asp?ArticleID=1034
22. Wang, L., Hu, W., and Tan, T., Recent developments in human motion analysis, *Pattern. Recognition*, 36 pp. 585-601, 2003
23. Want, R. TiltType: Accelerometer-Supported Text Entry for Very Small Devices, In *Proc. of UIST 2002*.
24. Wigdor, D., Balakrishnan, R., TiltText: Using tilt for text input to mobile phones. In *Proc. of UIST 2003.*
25. Yee, K. P., Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers. In *Proc. of CHI 2003,* pp. 1-8, 2003.