TinyMotion : Camera Phone Based Interaction Methods

Jingtao Wang
Computer Science Division
UC Berkeley
Berkeley, CA 94720 USA
jingtaow@cs.berkeley.edu

John Canny
Computer Science Division
UC Berkeley
Berkeley, CA 94720 USA
jfc@berkeley.edu

CHI 2006, April 22–27, 2006, Montréal, Québec, Canada.
ACM 1-59593-298-4/06/0004.

Abstract
This paper presents *TinyMotion*, a pure software approach that detects the movements of cell phones in real time by analyzing image sequences captured by the 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 and can run on today's main-stream camera phones without hardware modification. We describe the design and implementation of *TinyMotion* and analyze the potential interactions that can leverage *TinyMotion*. Three applications and two games were created to test *TinyMotion*. Benchmarking results and usability study show that *TinyMotion* can detect camera movement reliably under most background and illumination conditions.

Keywords
Input Methods, Mobile Devices, Computer Vision, Mobile Phones, Camera Phones, Motion Estimation.

ACM Classification Keywords
H5.2 Information interfaces and presentation: Input devices and strategies, Theory and methods.

Figure 1. Using *TinyMotion* enabled applications out-doors (up) and in-doors (down)

Introduction

Mobile phones have become an indispensable part of our daily life. Their compact form has the advantages of portability, but also limits the interaction methods that can be used. Although the computing and imaging capabilities of cell phones have increased significantly in the past fifteen years, a 12-button keyboard and a discrete four-direction joystick are still the dominant input interface. For basic voice functions, this is not a big hardship. But today's phones are capable of running games, map-based navigation and location services, and image and video browsing, all of which greatly benefit from a usable analog pointing device. We show in this paper that this capability can be added to camera phones with a software application.

Many technologies have been proposed and tested to improve interaction on mobile devices by enhancing expressiveness [4, 5, 9], or sensing context of the surrounding environment [5]. Accelerometers [5, 9], touch sensors [4, 5] and proximity sensors [5] have been used, and may eventually make their way inside the phone. However, today they are an exotic, rarely seen accessory.

On the other hand, camera phones are an extremely popular and pervasive item today. According to In-Stat/MD, the camera phone market is skyrocketing with worldwide annual shipments up more than 200% in 2004. It's expected that the market penetration will almost reach 100% in the near future [2].

In this paper, we present a technique called *TinyMotion* (figure 1) for camera phones. *TinyMotion* detects the movements of cell phones 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 experimental validation of the approach on a wide variety of background scenes, showing that *TinyMotion* gives a usable pointer under almost all conditions..

Related Work

Many compelling interaction techniques on mobile devices have been built by attaching sensors to the device. For navigation, Rekimoto used tilt input for navigating menus, maps, and 3-D scenes, and Harrison [4], Hinckley [5] and Wigdor [9] have used tilt for scrolling through documents, lists and entering texts. Peephole Displays [10] also explored the interaction between spatially aware displays and pen interfaces using external position sensors.

Hansen and colleagues [3] proposed the idea of "mixed interaction space" to augment camera phone interaction. Their method however, 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 for sensing. *TinyMotion* provides general motion sensing from whatever scene the camera is looking at, near or far.

Inspired by the success of CyberCode from SONY, several researchers and companies [7, 8] designed customized 2D barcodes that can be recognized by camera phones. Many interactions can be enabled by associating information in the bar codes with different actions. Most of these recognizers output

transformation parameters that measure the size, position and angle of the barcode relative to the camera's axis. This can be used to infer the camera's 3D position relative to the barcode, and provides an alternative spatial input channel.

The TinyMotion Algorithm

While computer vision techniques such as edge detection, region detection, optical flow, etc., can be used for motion estimation, they are less robust than direct methods based on correlation or image difference. The latter are used in cameras, optical mice, video codecs etc, and we follow suit. *TinyMotion* has used both image differencing and correlation of blocks [6] for motion estimation.

The *TinyMotion* algorithms include four major steps, 1. Color space conversion. 2. Grid sampling. 3. Motion estimation. 4. Post processing. All these tasks can be done efficiently by integer only operations.

At the beginning, we set up the camera in preview mode, capturing color images at the resolution of 176x112 constantly at the speed of 12 frames/sec[1] .

Color Space Conversion

After a captured image arrives, we use a bit shifting method (equation 2, an approximation of equation 1) to convert RGB color to gray scale.

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

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

Grid Sampling

Grid Sampling, a common multiresolution sampling technique [6], 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 experiments.

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 marco-block (MB) and apply Full-search Block Matching algorithm (FBMA) on temporal 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 R_x and R_y respectively. In our current implementation, R_x = (-3, 3), R_y = (-3, 3). Common distance measurements include Mean Square Error (MSE, equation 3[6]), Sum of Absolute Difference (SAD), Cross-Correlation Function (CCF). After block matching the motion vector is chose as the corresponding block shifting distance (equation 4).

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

[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.

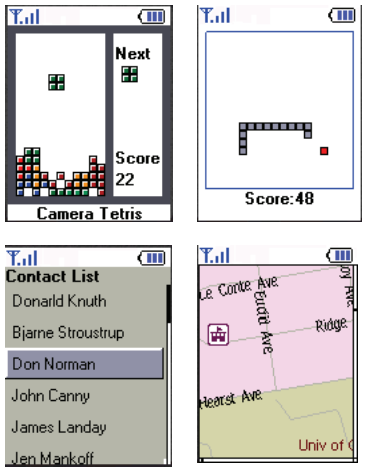

Figure 2. Sample *TinyMotion* Applications.

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

The motion vector $\overrightarrow{MV} = (MV_x, MV_y)$ represents the displacement of the best 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 upper part 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 is also distance changes in x and y direction. These relative changes are also accumulated to provide an absolute measurement to a fixed position.

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

Implementation

The Motorola v710 (a CDMA Phone from Verizon Wireless) is a typical state-of-the-art camera phone. The v710 has an ARM9 processor and 4M RAM, 176x220 pixel color display. Our application is written in C++ for BREW [1] (Binary Runtime Environment for Wireless) 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 diffusion path for commercial applications built using *TinyMotion*. We believe *TinyMotion* can also be ported to other platforms like Windows Mobile and Symbian easily.

We wrote three applications - Motion Menu, Vision TiltText (a remake of [9] by replacing accelerometer with *TinyMotion*) , Image/Map Viewer and two games – Camera Tetris, Camera Snake to test the effectiveness of our algorithm. All these prototypes can be operated by moving and tilting the camera phone. Figure 2 are some screen shots of the *TinyMotion* enabled prototype applications.

Evaluation

We evaluated the reliability of *TinyMotion* by two methods. First, we benchmarked the detection rate of camera movements in four typical conditions. In each condition, we shift/tilt the cell phones in four different directions. We applied 50 shift movements and 50 tilt movements for each direction. 1600 movements were recorded totally. In each movement, we either shift the cell phone on each direction for more than half inch or tilt the cell phone for more than 15 degrees. If the accumulated movements value in a certain direction exceeds threshold value 5, our application will output that direction as the detected movement direction.

	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)

Figure 3. Environments and Backgrounds in which *TinyMotion* can work properly.

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 moving into/out of the camera view during the testing process.

We also conducted an informal usability test by distributing camera phones installed with *TinyMotion* enabled applications/games to 13 users, most of the users currently selected are undergraduate, graduate students and faculty members in a local university. We asked them to play with these applications and encouraged them to challenge *TinyMotion* by any background and illumination conditions that they can think of or have access to.

All of the users reported successful usage in situations like pointing a building outdoor, pointing to piles of garbage, different types of grounds in doors and outdoors, grass in a garden, cloth, bus stop at night, indoor low lamination conditions, indoor/outdoor with color illuminations, different areas in pubs. Most were surprised to learn that motion sensing was based on camera input. One subject 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 detect camera movements.

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

We also collected many interesting feedback from the users from the informal usability tests. Comments from test subject 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.

One user felt *TinyMotion* was not very sensitive at the initial stage, later we found that it was caused by the unusual cell phone grasp that he used – he extended his index finger while holding the camera phone so that it blocked the lens of the camera!

Design Limitations

Although we observed that *TinyMotion* can provide natural and reliable interaction methods to camera phone users, due to the nature of the camera sensor, there are some innate design issues related with *TinyMotion*:

1. *Low refresh rate*. The frame rates for current camera phones are around 15 frames per second. This may limit TinyMotion's usefulness for gestural or other fast UIs. However, frame rates are likely to go up as support for live video (popular in Asia) and video capture on cell phones improves.

2. *Drifting Error*. As with most forms of mouse, drifting errors occur in the current implementation of TinyMotion. If the user moves the device in a loop, the pointer may not return to the same place. On the other hand, the lack of absolute measurement also allows the user to make many "grab and move" motions with a button to achieve larger movements that are possible with a single motion.

Future Work

TinyMotion can be adapted to absolute positioning with a fairly simple modification. By storing all recent images (which are very small after sampling), the current image can be matched with the closest image according to dead reckoning (the accumulation of recent offsets) rather than the last image. This still requires only one image comparison per frame, and avoids the accumulation of errors. This should be a better form of input for drawing applications [10].

Conclusion

In this paper, we proposed a method called *TinyMotion* that can measure cell phone movements in real time by analyzing images captured by the built-in camera. We described the design and implementation of *TinyMotion* and described applications that leverage *TinyMotion*. Benchmarking results and a usability study showed that *TinyMotion* can detect camera movement reliably under most background and illumination conditions.

TinyMotion is open source software released under BSD license. The current implementation can be downloaded freely from *http://guir.berkeley.edu/tinymotion*.

Acknowledgements

We thank Greg Niemeyer, Shumin Zhai, Jono Hey, Tom Duan, John Suarez and Kelly Li for their great suggestions and feedback. We also thank Qualcomm Corp. for their support in funding and equipment.

References

[1] BREW (*Binary Runtime Environment for Wireless*), Qualcomm Corp, http://brew.qualcomm.com

[2] *Camera Phone Market Continues to Boom - 200% Growth in Annual Shipments*, http://www.instat.com/press.asp?ID=1181&sku=IN0401703WH

[3] Hansen, T., Eriksson, E., Lykke-Olesen, *A., Mixed Interaction Space - Designing for Camera Based Interaction with Mobile Devices*, In Extended Proc. CHI 2005

[4] Harrison, B. L., Fishkin, K., A. et al, *Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces*. In Proc. CHI 1998, p. 17–24..

[5] Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E., *Sensing Techniques for Mobile Interaction*. In Proc. UIST 2000, p. 91–100.

[6] Kuhn, P., Algorithms, *Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Kluwer Academic Publishers.

[7] Rohs, M., Zweifel, P., *A Conceptual Framework for Camera Phone-based Interaction Techniques*, In Proc PERVASIVE 2005

[8] SemaCode, http://semacode.org/.

[9] Wigdor, D., Balakrishnan, R., *TiltText: Using tilt for text input to mobile phones*. In Proc UIST 2003.

[10] Yee, K. P., *Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers*. In Proc ACM CHI 2003, pp1-8.