Enhancing the Human-Computer Interaction on Camera-Equipped Mobile Phones

Jun Li
Graduate School of Science
and Engineering Department
University Of Toyama
3190 Gofuku, Toyama-shi, Toyama
Japan

Zheng Tang
University Of Toyama
3190 Gofuku, Toyama-shi, Toyama
Japan

Abstract: - Mobile phones interactions are mostly done via the buttons, thumbwheels or touchscreens. Thus, in this s paper we present an approach for enhancing Human-Computer Interaction (HCI) on camera-equipped mobile phones. By using this approach a user is able to interact with an application by moving the mobile phone. The built-in camera is used to capture frame sequences while the movement and rotation are being estimated. The movement and rotation data are used as an input for the application. In our approach, we present some method to overcome the limitation of the mobile phone and further enhance the precision as previously reported by other researchers. The algorithm used in estimating the 3D motion is the neuron network. To conclude, some examples of the implementations using the proposed approach are discussed in this work.

Key-Words: mobile phone, camera, Human-Computer Interaction, neuron network

1 Introduction

Mobile phones have been adopted into our daily life and it plays a more and more important role. In addition to the standard voice function of a telephone, the current mobile phones can support many additional services such as game, Short Message Service(SMS) for text messaging, email, packet switching for accessing the Internet and Multimedia Messaging Service(MMS) for sending and receiving photos and videos. We also depend on it in managing our personal day-to-day activity by using address book and calendar functions available in the mobile phones.

Mobile phones are becoming a personal computer in both functionality and interaction. The most common interaction is through buttons, thumbwheel or touchscreens. In this paper we present an approach for enhancing Human-Computer Interaction (HCI) on camera-equipped mobile phones which currently constitutes 85% of the market.

Our approach is to use the mobile phone's built-in camera as the source of input. The user's physical movement of the device is captured in frame sequence, which is analyzed to determine scroll direction and magnitude. The detected direction is sent to the event handler exactly as a corresponding mouse event while the magnitude is used to specify the current scroll level. In addition, based on the determined 2D motion data, we can also estimate user's movement in 3D. Fig.1 shows the diagram of interaction. For example when a user plays some shooting game, it's almost impossible to click the direction button and the fire button at the same time with one hand. But, by using our approach a user can indicate the direction by moving the mobile phone and at the same time he can shoot by clicking the fire button with one hand. We will introduce a demo of handwriting and a demo of 3D viewport control, which we implemented, too.

Fig. 1 Diagram of interaction

2 Related Work and Present Problem

There have been several studies in the past years that have investigated the HCI of mobile phone. Several projects have experimented using the camera on mobile devices for tracking and augmenting reality [3]. Rohs et al. [1] performed tracking based on dividing incoming camera frames into blocks and determined how the blocks move given a set of discrete possible translations and rotations. On the other hand, Drab, S. [2] uses his Projection Shift Analysis algorithm to do the motion detection.

Although a lot of researches have been done, it is still difficult to implement a practical motion estimation system which can run on a common mobile phone, especially the 3D motion estimation and this is mainly caused by the two biggest problems that will be discussed below.

2.1 The Limitation of Mobile Phone

Actually the hardware has become more powerful than ever. Considering the portability and the battery life, the computation capacity of mobile phone is finite. However the operations of edge detection, 2D motion estimation and 3D motion estimation requires very large computation capacity.

In 3D motion estimation, geometric manipulation such as sin(x) and floating-point is required. But not all of these are currently being supported by most mobile phones.

Due to the mass matrix computation, the problem of memory insufficient is often encountered. So, we have to take a balance approach between the memory usage, speed and precision.

2.2 The Complexity of Reality Environment

Variable factors from the environment contribute towards the noise. The mentioned noise can be a result of many actions, e.g. when a user is moving the device and at the same time the focus object is as well moving. As a result, this can cause a low precision and even the wrong result. Severe lighting differences or the low contrast of the environment can also reduce the precision.

The user's movement is complex and fuzzy. So the system needs more intelligence.

3 Motion Estimation

The objective of the mentioned "motion estimation" is not to estimate the motion of object but the motion of the camera.

2D motions can be estimated by the following two steps. The first step is to select some point in the frame as feature point. Although there are many method available for this selection, but in this proposal we are using the edge detection method. The second step is to estimate the direction and magnitude of movement by comparing the feature point between frame t and frame $t-1$.

3D motion estimation is more difficult than 2D motion estimation. In our approach, first we divide the whole frame into several regions and then estimate the motion of each region.

3.1 Edge Detection

Edge Detection [5] [6] is the process of finding edges in images. The goal of edge detection is to mark the points in a digital image at which the luminous intensity changes sharply. Sharp changes in image properties usually reflect important events and changes in properties of the environment. Several algorithms have been designed to detect edges and the complexity degrees ranges. As for our implementation, we choose the Sobel.

In simple terms, the operator calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point and therefore how likely it is for that part of the image to represents an edge, as well as how that edge is likely to be oriented. In practice, the magnitude calculation is more reliable and easier to interpret than the direction calculation.

Mathematically, the gradient of a two-variable function is at each image point a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each image point, the gradient vector points in the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction. This implies that the result of the Sobel operator at an image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values.

Mathematically, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes and the other for vertical. If we define A as the source image, where as G_x and G_y are two images which at each point contain the

horizontal and vertical derivative approximations. To avoid using floating-point, we use Equation1 to compute the grayscale value of A.

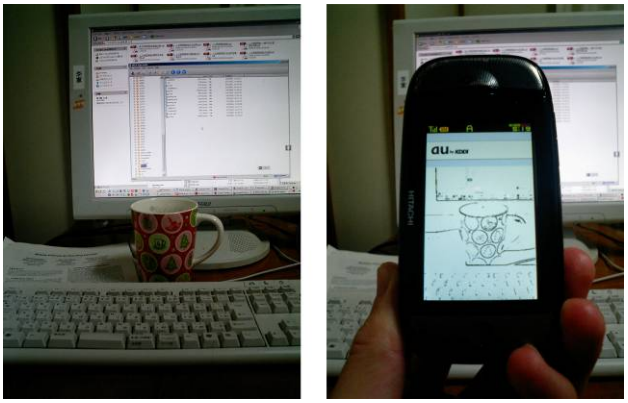
$$A_{gray} = \frac{30A_{red} + 59A_{green} + 11A_{blue}}{100} \quad (1)$$

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \times A_{gray} \quad (2)$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \times A_{gray} \quad (3)$$

$$G_{xy} = \sqrt{G_x^2 + G_y^2} \quad (4)$$

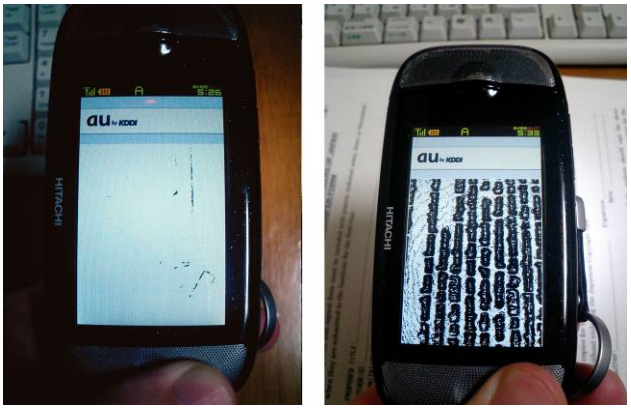
Fig.2 shows the demo of edge detection running on our test device.



(a) **(b)**

Fig. 2 Edge detection

In some special status, the result of default Sobel edge detection may be insufficient (Fig.3a) or overabundant (Fig.3b). As illustrated in Fig.3a, there are too less feature points to complete motion estimation. On the other hand, the condition of in Fig.3b returns too much feature points which will exhaust the computation resources and cause the delay of interaction.



(a) **(b)**

Fig. 3 Limit of edge detection

To solve this problem, a Threshold value T is defined. If G_{xy} is less than T, it will not be exported as feature point F_{xy}.

$$F_{xy} = \begin{cases} 1 & (G_{xy} > T) \\ 0 & (G_{xy} <= T) \end{cases} \quad (5)$$

The Threshold value T is adjusted using Equation 5 in real time. ΔT is the bias, N is the count of feature point, $N_{\min}$ is the minimum of N, and $N_{\max}$ is the maximum of N. Therefor the count of feature point is restricted within $N_{\min} \sim N_{\max}$. By using this method, the precision is improved and the computation resource is saved.

$$T = \begin{cases} T + \Delta T & (N < N_{\min}) \\ T & (N_{\min} \le N \le N_{\max}) \\ T - \Delta T & (N > N_{\max}) \end{cases} \quad (6)$$

3.2 2D Motion Estimation

To estimate the 2D motion, a comparison between frame t and frame $t-1$ is needed. Intend that, the maximum distance can be detected in pixel is i. A $(2*i+1)\times(2*i+1)$ matrix D_{xy} is used to present the direction and the magnitude. Here set $i=4$, so D_{xy} is a 9×9 matrix, however what we actually used in our implement is $i=10$.

In the frame t, a feature point and its 9×9 pixel neighborhood are showed as Table 1. In the frame $t-1$, the feature points in the same regions are showed as Table 2.

				1				

Table 1 frame t

	1							
			1					
		1						

Table 2 frame t-1

$$D_{xy} = \sum_{F_{xy}=1} \sum_{l=-4}^{4} \sum_{m=-4}^{4} F_{xy} \qquad (7)$$

After calculating the Equation 7, find out the maximum $D_{xy\max}$ in D_{xy} . If $D_{xy\max}$ is larger than threshold, then it is the result, and its subscript shows the vector of movement.

3.3 3D Motion Estimation

The usage of geometric method to estimate the 3D motion is possible but taken into considerations the limitations of mobile phone and the complexity of reality environment it is difficult to be used in practical method. Instead of using geometric method, we use the neural network algorithm. The network can be trained offline before using and this reduces the computation requirement. Furthermore, by applying the neural network method, there is no need for geometric function and floating-point which are not supported by most mobile phone. Using the neural network algorithm also improves the fault-tolerance of the system.

The neural network algorithm is only used to estimate the motion pattern, which has 12 kinds of possibilities such as pan left, pan right, zoom in, zoom out and etc.

3.3.1 Region Fragment

For using 2D motion data to estimate the 3D motion, divide the frame into 9 regions (Fig. 4). Estimate the 2D motion of each region, and save the result in R_{xy} which is a 3×3 matrix and look like Table 3.

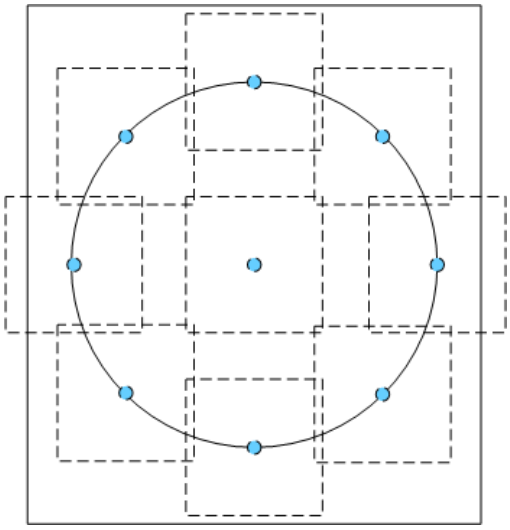


Fig. 4 region fragment

→	→	↗
→	→	→
→	→	↘

Table 3 the vector of R_{xy}

3.3.2 The Backpropagation Algorithm

The backpropagation algorithm [7][8] is a well known learning algorithm for feed-forward neural networks. The backpropagation neural network works in two modes, a supervised training mode and a production mode. The training can be summarized as follows:

Start by initializing the input weights for all neurons to some random numbers between 0 and 1, then:

1. Apply input to the network.
2. Calculate the output.
3. Compare the resulting output with the desired output for the given input. This is called the error.
4. For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output.
5. Repeat the process until error reaches an acceptable value (e.g. error < 1%), which means that the neural network was trained successfully, or if we reach a maximum count of iterations, which means that the neural network training was not successful.

3.3.3 Network Training

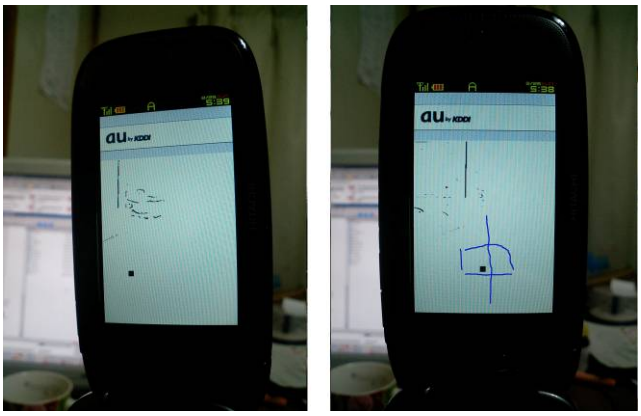
For each of 12 possible motion pattern, record a set of R_{xy} , and then use these data to train the backpropagation neural network. After the training phase, we can test it on the device.

4 Applications

The demo applications were implemented on the Brew 3.1 of Qualcomm and tested on W41h model of AU, which has a 2.1 mega pixels built-in camera. The programming language is C++ and the application size is less than 200Kbyte. The system requirements of hardware are more than 1Mbyte heap memory and more than 0.3 mega pixels built-in camera.

4.1 Handwriting

Fig.5 shows a simple handwriting application. The black square below the screen will move to the same direction when a user moves the mobile phone. If user holds down the select button and moves the device, the blue trail will be drawn on the screen.



(a) **(b)**

Fig. 5 Demo of handwriting

4.2 3D viewport controls

Fig.6 shows the demo of 3D viewport control. User can control the virtual camera in the 3D scene without clicking any button such that the device is moved up or down, left or right to pan the viewport and also brought towards and further from the object in order to zoom the viewport as well as to rotate the mobile when a rotated image of the viewpoint is required.

Fig. 6 Demo of 3D viewport control

5 Conclusion

In this paper, we presented an approach to enhance the HCI on Camera-Equipped Mobile Phones. We proposed some feasible and efficient method to implement the 2D and 3D motion estimation. Although the performance of the current work is satisfactory but there are areas in this work which can be further improved for better performance and reliability in order to provide a more pleasant and user friendly interaction experience.

References:

[1] Rohs, M.: Real-world interaction with camera-phones. *International Symposium on Ubiquitous Computing Systems.* (2004).

[2] Drab, S., Artner, N.: Motion detection as interaction technique for games & applications on mobile devices. *Extended Abstracts of PERVASIVE: Workshop on Pervasive Mobile Interaction Devices.* (2005) 48-51

[3] D. Beier, R. Billert, B. Brderlin, D. Stichling, and B. Kleinjohann. Marker-less vision based tracking for mobile augmented reality, *In The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE, Inc.*, 2003. pp.258-259.

[4] Odobez J. and Bouthemy P. Separation of moving regions from background in an image sequence acquired with a mobile camera. In H.H. Li, S. Sun, and H. Derin, editors, *Video Data Compression for Multimedia Computing,* chapter 8, pages 283–311. 1997.

[5] Carron, T. Lambert, P., Color edge detector using jointly hue, saturation and intensity. *Image*

Processing, 1994. Proceedings. ICIP-94., IEEE International Conference

[6] Touzi, R. Lopes, A. Bousquet, P., A statistical and geometrical edge detector for SAR images. *Geoscience and Remote Sensing, IEEE Transactions on* Vol.26, No.6, 1988, pp. 764-773

[7] Hecht-Nielsen, R, *Theory of the backpropagation neural network*, Neural Networks, 1989. IJCNN., International Joint Conference on, Vol.1, 1989, pp. 593-605.

[8] D.E Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," D.E. Rumelhart and J. McClelland, editors, *Parallel Data Processing*, Vol.1, Chapter 8, The M.I.T. Press, Cambridge, MA, 1986, pp. 318-362.