A Gestural Interaction Design Model for Multi-touch Displays

Songyang Lao
National University of Defence Technology
laosongyang@vip.sina.com

Xiangan Heng
National University of Defence Technology
xianganh@hotmail

Guohua Zhang
National University of Defence Technology
zghnudt@gmail.com

Yunxiang Ling
National University of Defence Technology
yxling@tom.com

Peng Wang
National University of Defence Technology
pwang@computing.dcu.ie

ABSTRACT

Media platforms and devices that allow an input from a user's finger/hand touch are becoming more ubiquitous, such as Microsoft Surface and DiamondTouch, as well as numerous experimental systems in research labs. Currently the definition of touch styles is application-specific and each device/application has its own set of available touch types to be recognized as input. In this paper we attempt a comprehensive understanding of all possible touch types for touch-sensitive devices by constructing a design model for touch interaction and clarifying their characteristics. The model is composed of three structural levels (action level, motivation level and computing level) and the relationships between them (mapping).

In action level, we construct a unified definition and description of all possible touch gestures, first by analyzing how a finger/hand touch on a surface can cause a particular event that can be recognized as a legitimate action, and then using this analysis we define all possible touch gestures, resulting in touch gesture taxonomy. In motivation level, we analyze and describe all the direct interactive motivation according to applications. Then we define the general principles for mapping between the action and motivation levels. In computing level, we realize the motivation and response to gestural inputs using computer languages.

The model is then used to illustrate how it can be interpreted in the context of a photo management application based on DiamondTouch and iPod Touch. It allows to reuse touch types in different platforms and applications in a more systematic and generic manner than how touch has been designed so far.

Categories and Subject Descriptors

H5.2. [**Information interfaces and presentation**]: User interfaces *–Standardization, Theory and methods.* H1.2. [**Models and principles**]: User/Machine systems *–Human factors.*

General Terms

Human Factors, Standardization, Theory.

Keywords

Touch interaction, gestural recognition, mapping rules, interaction model, tabletop, PDA.

1. INTRODUCTION

Currently, multi-touch technology is a popular research area in human computer interaction, gaining momentum with the appearance of commercial products such as Microsoft's Surface and DiamondTouch. Previous research on touch/gestural interaction has concentrated on gesture definition and recognition. However, there are issues that need to be addressed if we want to re-use touch types and styles to provide consistency for endusers. For example, the meaning of a touch type can vary according to different applications such as two hands moving closely on a surface meaning *zoom out* in a GIS-based application and *gather scattered items* together in a game interface. Different users and/or different cultures may have different ways they operate and interpret a touch, calling in the possible re-mapping of touch types and their meanings for different users/cultures. Touch may need to be interpreted differently depending on the situation of applications.

Because each platform which supports multi-touch inputs has its own mechanism to locate touch points and provides its own set of toolkit to gain the touch points on the surface and its trajectory, there is not a conventional comprehension of gestures, and the algorithm of gesture recognition varies from different platforms. Interaction designers design gestural interactions based on each specific platform, so that gestures can't be reused throughout different platforms. With the development of interactive technologies and software, there is an urgent requirement to establish a conventional comprehension and definition of gestures which will be benefit to a middleware to make all the multi-touch platforms support gestural interaction. The middleware will also contribute to the next generation of Operating System which will support multi-touch and gesture inputs, illustrated in figure 1.

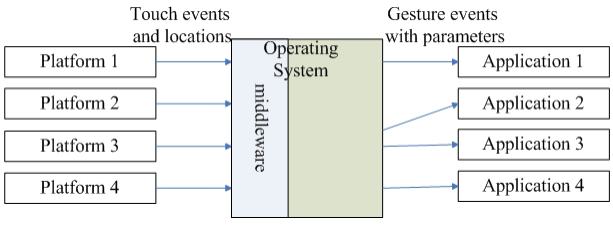

Figure 1: gesture support middleware in Operation System

We have established a model for touch interaction in order to allow a systematic approach in defining a touch and its meaning, and ultimately to allow re-use of touch for different applications, platforms, and use contexts. The model is

comprised of three levels, the action, motivation, and computing levels. We describe each separately, define *mapping rules* and we apply the model to gestural interaction design based on different platforms and applications. In defining and classifying gestures to be used on touch platforms and we provide a foundation for the mapping between a human gesture and the action that it causes thus serving as a useful guideline for designers of touch applications.

Next we summarize related work in categorizing touch/gesture actions. We then describe our interaction model, components, their properties and relationships (mapping rules). In Section 4 we illustrate how this model can be applied and interpreted in the practical cases of a photo management application, and we conclude the paper with our perspective and future work.

2. RELATED WORK

While there are many works on developing different kinds of novel platforms and applications, there is no effort or organized activity on generalizing or standardizing touch interaction other than some definition of available gestures/touch for specific applications.

With recent advances in input sensing technology, researchers have begun to design freehand gestures on direct-touch surfaces. Ka-Ping [1] augmented a tablet computer with a touch screen to enable hand and stylus interaction. Jacob et al. [2] presented a "$1 recognizer" to enable novice programmers to incorporate gestures into their user-interface prototypes. Rekimoto [3] described interactions using shape-based manipulation and finger tracking using the SmartSkin prototypes. Mike et al. [4] presented multi-finger and whole handle gestural interaction techniques for multi-user tabletop displays. Finally, Morris *et al.* [5] presented multi-user gestural interactions for co-located groupware. These are all very useful contributions to the touch/gesture interaction field but a more generalized understanding of touch interaction focused beyond the specific realization of a device, is required. Shruti et al. [6] explored the user's perceptions to a novel interaction method with mobile phones. They studied responses and reactions of participants towards gestures as a mode of input with the help of a low fidelity prototype of a camera mobile phone. The study used an approach inspired by participatory design to gauge the acceptance of gestures as an interaction mode.

Elias et al. [7] presented a multi-touch gesture dictionary which includes a plurality of entries, each corresponding to a particular chord. The dictionary entries can include a variety of motions associated with the chord and the meanings of gestures formed from the chord and the motions. The gesture dictionary may take the form of a dedicated computer application that may be used to look up the meaning of gestures. It may also take the form of a computer application that may be easily accessed from other applications. And it may also be used to assign user-selected meanings to gestures. Mike et al. [8] developed a set of design principles for building multi-hand gestures on touch surfaces in a systematic and extensible manner. They proposed the concepts of gesture "registration", "relaxation", and "reuse", allowing many gestures with a consistent interaction vocabulary to be constructed using different semantic definitions of the same touch. While this is in line with the direction of our work, we attempt to standardize and generalize the whole picture of the interaction where the user's intentions, touch actions, and their mapping to system functionality are understood and specified.

3. A TOUCH INTERACTION MODEL

We structured touch into three levels in our model, as illustrated in Figure 2.The first is the *action* level which is independent of applications or platforms, and only explains what people can do (e.g. organize a photo collection or find a location on a map). The second level is *motivation*, also independent of platforms but specific to applications. This level explains a user's motivation of what they want to do when interacting (e.g. annotate a photo or send an email). This level can be reused by different platforms if they have the same application domain. The third level is the *computing* level, including hardware and software. It is specific to platforms and applications, and links people's actions to functionality in order to react and perform a specific set of tasks. The three levels make up the structural layout in our touch interaction model. When we design a touch interactive interface, we only need to design touch at the action level once, and can reuse in other applications and platforms. Then we define different mapping rules from the action level to the motivation level according to the application domain.

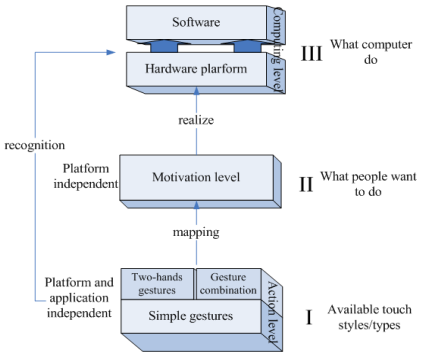

Figure 2: A touch interaction model

3.1 Gesture definition and description

Considering there are many multi-touch platforms each of which has its own mechanism of acquiring touch locations and gesture recognition. There is a requirement to establish a conventional comprehension of gesture definition and description so that gesture interactions can be widely used not constrained by platforms. The definition and description also can contribute to the Operation System which will support multi-touch and gestural inputs.

Firstly, we summarize all the possible gestures according to ergonomics and presence. We consider single hand the basic unit of gestures according to ergonomics and distinguish between two kinds of gestures: simple gestures and complex gestures which are made up by simple gestures. We define simple gestures below:

- It's a single hand action.
- It can't contain repeated actions.
- It can't contain other simple gestures.

We consider that there are two main elements to describe simple gestures, the touch styles between hand and surface, and the hand movement types, which are suitable for all the multi-touch platforms. Actually, there are some other elements such as pressure and touch depth specific with platforms, we will consider these as parameters of gestures, which will be described in next session.

There are two kinds of touch styles according to the contacting part between hand and surface: continuous contact and discrete contact, which have different movement types with each other. Continuous contact includes touching with one finger, palm,

half-palm (four close fingers except thumb), fist and vertical hand, figure 3. Discrete contact includes touching with 2 fingers, 3 fingers, 4 fingers and 5 fingers, figure 4.

Figure 3 continuous contact

Figure 4 discrete contact

We define 3 kinds of basic movements which can compose all the possible movements, pressing, tapping and dragging. Pressing means touch the surface and stay. Tapping means touch the surface and lift soon. Dragging means touch and move on the surface. When touch with continuous contact, users only can do basic movements. When touch with discrete contact, users can do more complex movements which are composed by basic movements of each touch finger, but are still limited by ergonomics.

pressing tapping dragging

Figure 5, basic movement

Touch styles and movement types explained so far are summarized in Table 1 as gesture description. When we define a new gesture, we choose a touch style and movement type from Table 1.

Table 1, simple gesture description

Touch styles		Movement types	
continuous contact	1 finger	Tapping Pressing Dragging (towards same direction)	
	Palm		
	Half-palm		
	Fist		
	Vertical hand		
discrete contact	2 fingers		Transition between 2 touch styles Dragging in bi-direction Dragging apart Dragging close One press and the others dragging
	3 fingers		
	4 fingers		
	5 fingers		

For example, when we choose 2 fingers, other than press, tap and drag, there are also one finger press and the other tap, drag in bi-direction, drag apart and drag close, as illustrated in Figure 6.

Figure 6, 2 figures gestures

Complex gesture is a spatial combination or temporal sequence of simple gestures, which is constrained by ergonomics as well. Generally speaking, users are used to gestures which are symmetrical or fixing one and moving another. Actually, the more complicated the gesture, the fewer the people who will be able to perform it. Anyway, it provides another choice in case an application requires multitude of functionality distinctions with subtle finger gestures.

We construct the description of simple gestures we summarized above based on touch events and locations which each platform can acquire by its own mechanism, in order that it can be understood and performed by computer logic. The description can provide a general guideline to define a unified simple gesture recognition algorithm which is compatible for each platform.

Firstly, we define the 3 basic movement types, pressing, tapping and dragging using state-transition diagram, illustrated in figure 7. In the gesture definition, we consider one single continuous touch as the basic element. Different touch styles are defined as different states (circle in figure 7). Then the gestures can be defined by state-transition.

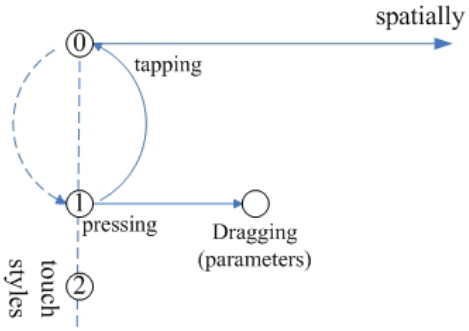

Figure 7, definition of "pressing", "tapping" and "dragging"

Similarly to the definition of pressing and tapping, we define the gestures which are described by movement type "transition between 2 touch styles" using state-transition diagram, illustrated in figure 8. The definition can be performed by detecting the change of touch points which is supported by most of the multi-touch platforms.

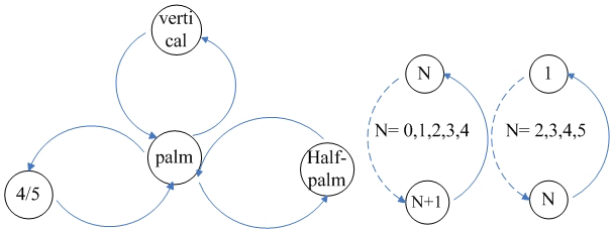

Figure 8, definition of "transition between 2 touch styles" (the circle means the touch styles or finger amount)

As we described in section 3, the movements of discrete touch are composed by basic movements of each continuous touch. Especially for dragging, it includes dragging in bi-direction, dragging close or apart, and dragging towards the same direction. We will define the gestures which are described by movement "dragging" below. We define the gestures by constructing a rectangular coordinate system corresponds to the 2D touch space first. Then gestures can be defined by number of touch points and the movement types, illustrated in figure 9.

Figure 9 illustrates the gestures which are combined by touch style "discrete touch" and movement type "dragging". We conclude 5 dragging states which are illustrated with A, B, C, D and E. We consider such a state as a serial of macroscopic actions which have the same movement tendency. The states are also defined with some specific parameters which explain

the details of movements such as speed, direction and pressure which is supported by some platforms, we discuss the parameters below.

Considering states C and E, according to the track, direction or speed of dragging, we can define them with some possible parameters. The parameter could be an angle which means the drag directions, for example, 0 means dragging left and 90 means dragging up, could be a number which means the speed of dragging, could be a description such as "circle", "clockwise" as well, which means the shapes or tracks. Similarly, speed can always be defined as a parameter for each dragging state, as well as pressure if the platform supports.

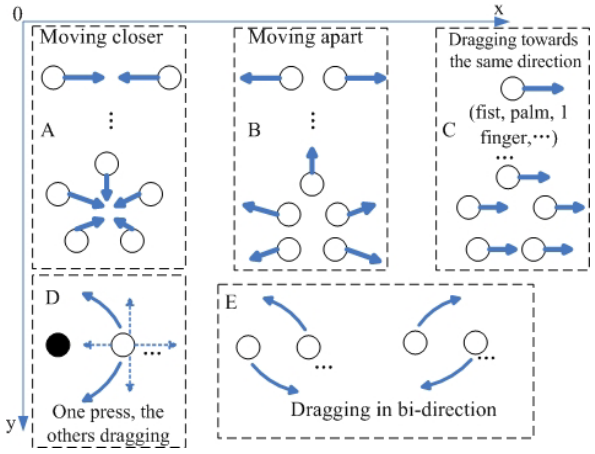

Figure 9, definition of "dragging"

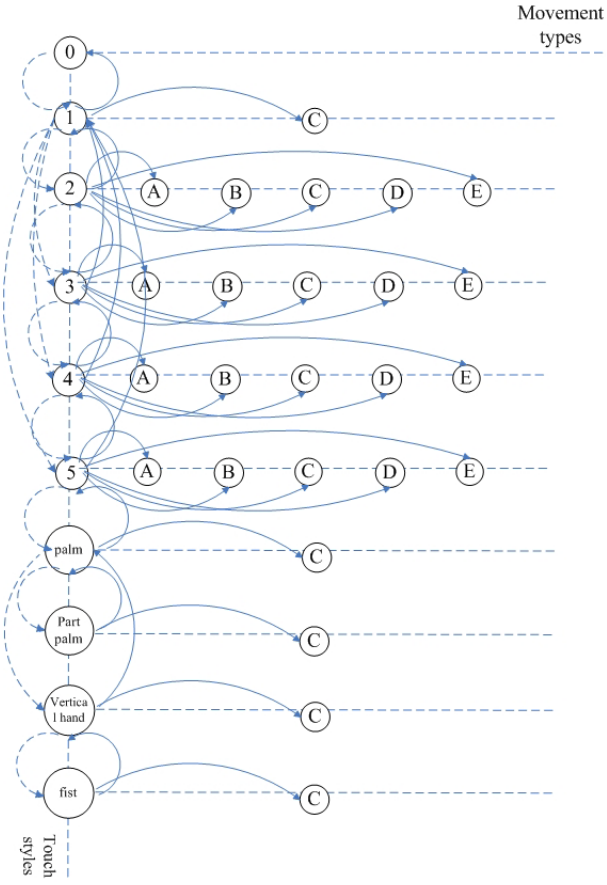

Figure 10, description of simple gestures

Integrating all the definitions above, we construct the gesture definition system below, illustrated in figure 10. When detect the touch events, the programmers can recognize gestures according to the state-transition diagram.

For complex gestures which are composed by simple gestures, they can be recognized as well by combining simple gesture events.

At last of this level, we can design a unified simple gesture recognition algorithm which is compatible for each platform based on touch locations. Figure 11 shows the logic flow of the recognition algorithm.

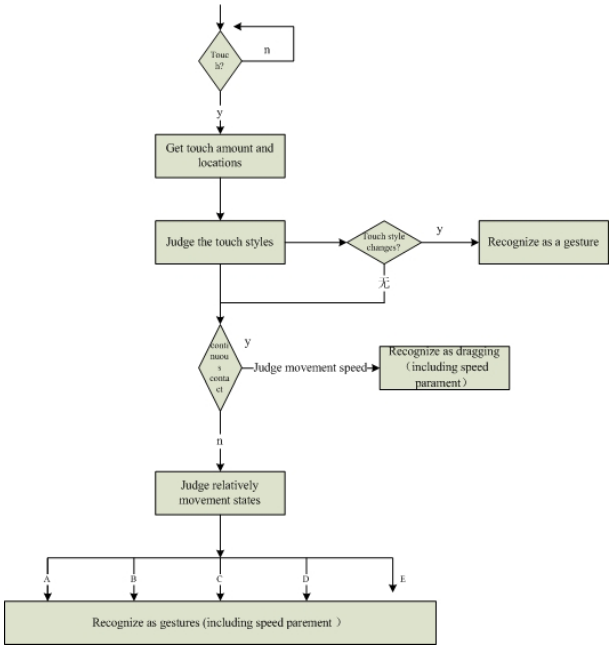

Figure 11, the logic flow of gesture recognition algorithm

3.2 Motivation Level

The motivation level addresses what people want to do and describes people's motivation according to the functions of the application. It is specific to a given application only and independent of platform. When an application is defined, all the motivations people can have when they interact with that particular application are confirmed as well. We use task analysis to descript this level. We consider the basic atom of task is that the smallest computer response that users can feel. For example, a photo is zoomed in is the smallest response that users can feel (not the zoom in button is clicked). We take photo management application as an example, illustrated in figure 12. As this level is independent of platforms, it can be reused for different platforms.

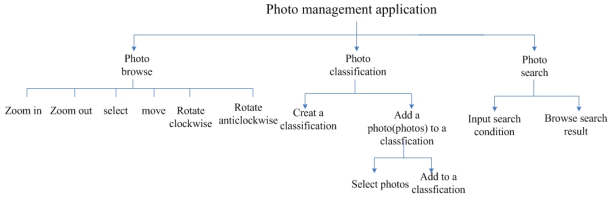

Figure 12: motivations in a photo management application

3.3 Computing Level

This level is concerned with how a computer reacts to a touch action. We divide this into two parts – hardware and software. We read the locations of touch points from the hardware and

recognize gestures according to these locations. The hardware provides runtime touch locations to the software. Although there are many different hardware platforms, algorithms that recognize gestures are always the same. We use a toolkit of gesture recognition algorithms suitable for any platform. Such algorithms will handle the distinction between available and illegible gestures and ignore noises caused by the environment etc. Such work is out of the scope of this paper.

3.4 Mapping Rules

In order to complete the touch interaction design process, we need to define mapping rules between the action and motivation levels and also the recognition and realization algorithm between action, motivation and computing levels. First, we define three general mapping principles below between action level and motivation level, ordered by priority.

1. Intuitive – the mapping rule must be in conformance with human's intuition and cognition.
2. No confusion – when the rules are defined, there should not be misunderstandings either for human or for computers.
3. Simple and direct – it's better to accomplish the motivation by fewer steps or in a simpler way. As we mentioned above, we should map the gestures to the motivations as high as possible in the hierarchy of motivation level.

We have specific cognition in our real lives, for example shaking hands for friendship and nodding for agreement. This is similar for touch gestures so we can't define the mapping rules randomly and we should make them consistent with our intuition. We take the map browsing application as an example. We usually map "two hands moving apart" to "zoom in the map" and "two hands moving close" to "zoom out the map". If we swap these two around, it will feel unnatural.

Sometimes there is a situation that we need to map the same gesture to several motivations. For example, people feel it convenient to pan a map by dragging with the forefinger. However, we also feel it convenient to draw a path on a map by the same dragging action with the forefinger. In this case, we need to make the computer register the same gesture as different motivations, possibly by setting different modes for the interface at the time of interaction. On the other hand, we may need to map several gestures to the one motivation, because different people or the same people at different times or situations might have different gesture preferences. For example, some people like to use two fingers to rotate a photo on a surface and sometimes they like to use three fingers or five fingers to do the same. Thus, between gestures and motivations there can be one-to-many as well as many-to-one mappings. We take a photo management application as an example, the mapping relations are illustrated in figure 13.

We introduce an *interactive context* in order to know which of more than one motivation should be mapped in response to a gesture. This can be considered as a mode or condition at the time of user interaction. For example, when we design a photo management application, when two fingers are above a photo and moving apart, it will register as zoom out the photo, otherwise, it will register as creating a new classification. Another aspect to note is that sometimes a motivation cannot be directly mapped to a gesture. A high-level motivation such as "search the photos in my photo collection" will need to be divided into sub-motivations in order to get any mapping to actual gestures.

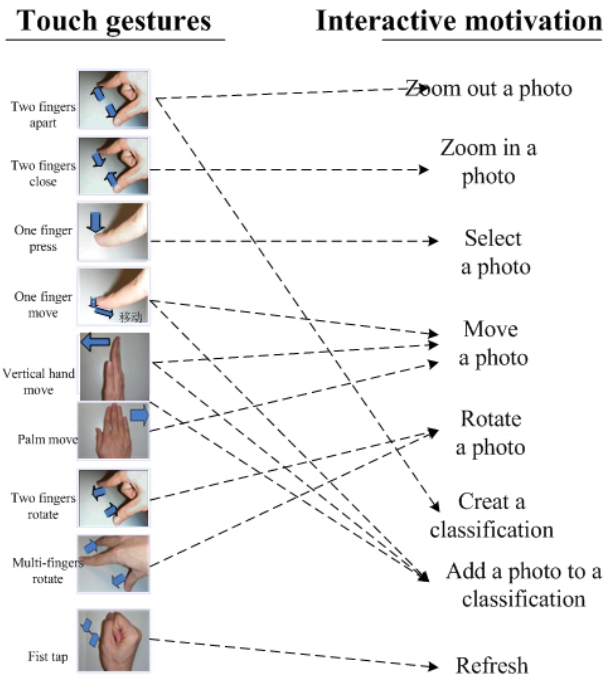

Figure 13 mapping relations between gestures and interactive motivation in a photo management application

Traditionally, we design windows, icons, menus and pointer (WIMP) to construct the interface to an application. This can be unnatural for us because we first have the motivation of clicking some icons or menus and then we need to know what will happen when they click. When we divide motivations into sub-motivations, the sub-motivations are usually exploring specific WIMP elements. We should try to reduce the WIMP elements to make the interaction and interface simple and clear.

Finally, we define the mapping rules from the action and motivation levels to the computing level using calls to the API. As we mentioned above, the computing level can recognize gestures, so what we need to do is to make the appropriate responses to each touch interaction.

4. APPLICATIONS

What we present so far is a general model for touch interactions derived from an extensive set of observations of touch applications on the DiamondTouch, iPod Touch, and other touch devices. Now we apply the model to two different touch platforms – a public tabletop (DiamondTouch) and a private PDA (iPod Touch).

We designed a photo management application for both DiamondTouch and iPod Touch platforms. The functions of application on DiamondTouch and iPod Touch are different m each other. On DiamondTouch, we can move a photo, rotate a photo, copy, zoom in, zoom out and delete a photo, and can reset the table, draw on the table, erase the drawing on the table. On iPod Touch, we can select a photo, zoom in, zoom out, delete, maximize and unmaximize a photo, we can also draw on a photo and erase the drawing.

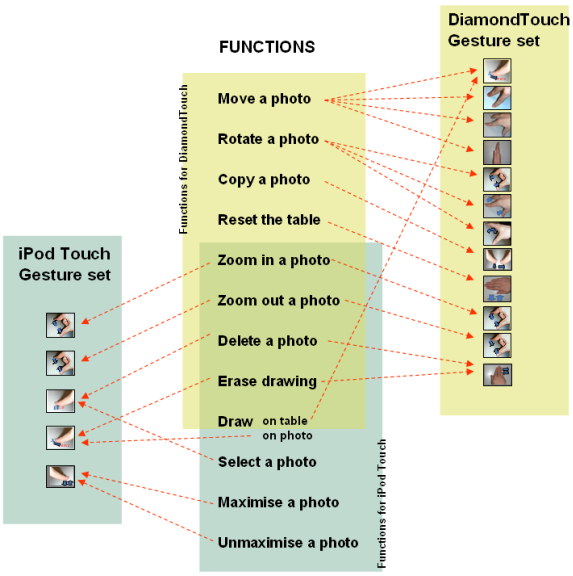

Figure 14, mapping relations in iPod Touch and DiamondTouch

4.1 Tabletops

A tabletop such as the DiamondTouch is usually designed for group decision making and its touch interaction has several characteristics including a large public shared screen so that all group members can gather around and each can use two-handed gestures. A tabletop is usually arranged so people can sit or stand around it so gestural input from different directions should have the same meaning. Applications running on tabletops are quite wide ranging so gestural interactions should be comprehensive. We have examined many tabletop applications in our own lab including games, map browsing, photo browsing and multimedia search but we take map browsing application as an example to explain the model for tabletops. Users can pan, zoom in, zoom out rotate the map, measure the distance between two points, get the location of a point, draw or annotate, etc.

When we design the application, we selected some gestures mapping to corresponding functions according to mapping rules. In order to make the interaction more comprehensive, we sometimes map several gestures to one function; sometimes map one gesture to several functions. For example, to rotate a photo, some people are used to using thumb and forefinger, some are used to thumb and forefinger, middle-finger. To move a photo, some are used to forefinger dragging; some are used to half-palm dragging or two fingers dragging. By contraries, people are used to moving a photo by one finger dragging, also by two fingers dragging or vertical hand dragging. So we defined the mapping rules of tabletop application, illustrated in figure 14.

We implement the application using java with DiamondTouch API. We detect the touch events and get touch points to recognize gestures; we also design some gestures events for some usual gestures, such as zoom in, zoom out, etc.

4.2 PDAs

PDAs are mainly designed for private applications like information management and entertainment. They have a small, private screen and the touch area is the display area. Users

Figure 15, application on DiamondTouch

usually use one hand to hold the device and touch with the thumb of the holding hand and the fingers of the other. There are usually external inputs to the PDA, such as physical buttons.

At the action level, gestures are subsets of gestures for tabletops. Because the screen is small, some touch styles between hands and PDAs can be unified. For example, we unify palm, half palm, fist and vertical hand as one style. At the same time, it's hard to distinguish one-hand and two hand-gestures which use the same number of fingers and have the same movement types. Because users usually have one hand holding the PDA and only the thumb can move, two-handed gestures are limited. Thus the possible gestures for PDAs are a subset of those for a tabletop.

We take a photo management application as an example to explain a model for PDA interactions. People can browse, resize, create classifications, group, search and rotate photos, etc. and we describe the motivations in Figure 14.

We divide motivations until they are considered to be accomplished directly and define the mapping rules in Figure 14. Here we have one gesture mapping to more than one motivation, such as one finger dragging (4^{th} gesture from the top in Figure 14). If there are photos selected, one finger dragging means adding selected photos to a specific class, otherwise, it means scrolling. As we know, there are often physical buttons on a PDA, so we can use these to indicate different interactive contexts. For example, if users press a button and drag on the screen, it means select photos; otherwise it means scrolling.

5. CONCLUSIONS AND FUTURE WORK

We have established an interaction model for touch interaction comprised of action, motivation and computing levels, in order to allow re-use of gestures and promote consistency in user interaction across applications and devices. We have defined a comprehensive understanding of touch gestures and gave a unified description of simple gestures. Then we have defined the principles of mapping rules between touch gestures and interactive motivations. At last, we take a photo management application as an example and illustrated the gesture design process based on two different platforms- iPod Touch and DiamondTouch.

For the future, we hope to standardize each level and the mapping rules, and also the process of gestural interactive interface design. We also plan to develop a number of combined applications for tabletop and PDA with the touch gesture model and its mapping rules in mind from the start. Our ultimate aim is to build gestural recognition middleware for all platforms, rather than retrofit as has been done heretofore.

6. ACKNOWLEDGMENTS

We thank P. Wang and L. Bai for help. This work was fully supported by the National Natural Science Foundation of China under Grant No.60875048 and Chinese 863 project under Grant No.2007AA01Z193.

7. REFERENCES

[1] Yee, K.-P. (2004) *Two-handed interaction on a tablet display*. Extended abstracts of ACM CHI. p. 1493-1496.

[2] Jacob O. Wobbrock, Andrew D. Wilson, Yang Li. (2007). *Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes*. ACM UIST. p. 159-168.

[3] Rekimoto, J. (2002). *SmartSkin: an infrastructure for freehand manipulation on interactive surfaces*. ACM CHI. p.113-120.

[4] Wu, M., & Balakrishnan, R. (2003). *Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays*. ACM UIST. p. 193-202.

[5] Morris, M. R., Huang, A., Paepcke, A., & Winograd, T. (2006). *Cooperative gestures: multi-user gestural interactions for co-located groupware*. ACM CHI. p. 1201-1210.

[6] Shruti, B., & Youn-Kyung, L. (2008). *Exploring gestural model of interaction with mobile phones. ACM CHI. p.2979-2984.*

[7] Elias, John, G., Westerman, Wayne, C., Haggerty, Myra Mary. (2007). *Multi-touch gesture directory.* United States Patent Application. 20070177803.

[8] Wu, M., Shen, C., Ryall, K., Forlines, C., & Balakrishnan, R. (2006). *Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces*. IEEE TableTop. p. 183-190