

[0046] Multi-touch gesture-based user interfaces allow multiple gestures in parallel. For example, a user can resize two images at the same time using both hands. Thus, data input can include a single gesture or multiple gestures that occur simultaneously. One method to distinguish these two cases consists in associating each contact points to a user interface element. A user interface element can be any classic WIMP widgets (button, slider, etc.) or advanced components. If two contact points are associated with the same user interface element, they will be considered as part of the same gesture. Otherwise, they will be considered as part of two distinct gestures. This filtering technique is particularly efficient to reduce the gesture recognition to a small set of gestures which are likely for a given object of the user interface. It also allows detecting and recognizing a plurality of gestures that are concurrently performed on a multi-touch input device.

[0047] In one embodiment, the filtering technique is a two steps process, as illustrated in FIG. 5. The first step consists in finding the underneath application for each contact point. For example, in FIG. 7, contact point “1” belongs to “Application A”, and contact points “3” and “4” belong to the same application, “Application B”. The second step is to associate each contact points within an application to a user interface element of this application. For example in FIG. 8, contact points “2” and “3” are part of the same gesture whereas contact points “1” and “4” are part of two distinct gestures. This filtering technique allows each application and each user interface element of an application to specify its set of supported multi-touch gestures to the recognition engine. By tightly coupling applications to the multi-touch gesture recognition engine, we can reduce the gesture recognition to a small set of gestures and, hence, effectively reduce the required computer system resources (e.g. CPU cycles, memory, etc.).

[0048] Another filtering technique is to group contact points based on their spatial proximity, i.e. their relative distance. If the distance between two contact points is smaller than a threshold, they will be considered as part of the same gesture. Otherwise, they will be considered as part of two distinct gestures. For example in FIG. 9, the contact points labelled “1” to “5” can be split into three groups labelled “A”, “B” and “C” based on a spatial segmentation.

[0049] In a multi-touch application, multiple different gestures can be performed on a single user interface element. For example, in a multi-touch photo sharing applications, a user can perform a pinch gesture to resize a photo or a rotate gesture to rotate it. A significant problem in gesture recognition systems, in particular for multi-touch gesture recognition, is how to accurately/reliably and speedily discriminate a gesture being made from a dataset of candidate gestures.

[0050] In one embodiment, the recognition of multi-touch gestures is preferably based on a multi-agent approach, as illustrated in FIG. 6, which enables separate gesture recognizers to work in parallel on the interpretation improving the probability of correct gesture discrimination and reducing the response time. A set of independent agents, each with its own recognition algorithms and supported gestures, analyzes the user interaction sequence to determine the likelihood of a gesture. The agents then independently determine the confidence of their respective findings and a distributed arbitration, the recognition manager, resolves the interpretation through a confidence sorting and a rejection threshold filtering. The multi-agent approach provides an effective way to solve prob-

lems which are difficult or impossible for a monolithic system to solve. It is also easily extensible to new gestures and recognition algorithms.

[0051] As explained above, a key element of effective gesture interaction is the ability to interact without lifting all fingers of the multi-touch surface, i.e. the ability to perform a continuous sequence of temporally contiguous multi-touch gestures instead of a collection of isolated gestures.

[0052] To deal with this problem, the invention proposes a gesture recognition method which considers continuity of motion and transitions between gestures to provide a more natural interaction to end-users.

[0053] In one embodiment, the continuum between gestures can be controlled by a finite state machine (FSM): for example, an FSM which states correspond to the number of contact points and transitions between these states are touch down and lift off events. FIG. 10 shows a simplified process to control the continuum between a single finger gesture (here a drag-n-drop (DND)) and a two fingers gesture (here a translation, rotation and scale (TRS) gesture) using a FSM with four states (“0”, “1”, “2” and “3 or more”): the current state of the FSM is linked to the number of contact points. Whenever two contact points are detected, the gesture will be interpreted as a TRS. Whenever only one contact point is detected, the gesture will be interpreted as a drag-n-drop. It should be appreciated that the FSM can be driven by many different types of events: either external events such as keyboard or network events, or internal events such a timer or previous recognition results.

[0054] In a similar vein, the continuum between gestures can be controlled by the “switch case” statement that exists in most programming language: the number of contact points is used to control the flow of gesture recognition. A number of such techniques (Petri nets, Markov models, etc.) are available to the person skilled in the art.

[0055] In another embodiment of the invention, the multi-touch gesture recognition engine may know multiple different gestures performed with the same number of contact points: for example, FIG. 11 illustrates three different static gestures that can be performed with three fingers. FIG. 12 and FIG. 13 shows two dynamic gestures performed with two fingers. FIG. 12 illustrates a pinch gesture: the user places two fingers (e.g. its thumb and its index) on the multi-touch surface and moves them closer to each other mimicking a pinch. FIG. 13 illustrates a rotation gesture: the user places two fingers (e.g. its thumb and its index) on the multi-touch surface and either turn them clockwise or counter clockwise.

[0056] FIG. 14 illustrates a process to dynamically distinguish a two finger pinch gesture from a two finger rotation gesture using a set of simple heuristic rules based on threshold values. Whenever two contact points are detected, their current positions are saved. Then, the recognition engine monitors these two contact points and calculates two key parameters, the distance and the angle between contact points, each time the position of a contact point changes. If an absolute distance variation greater than a given threshold is detected, a zoom event is triggered: a zoom in event if the distance increases and a zoom out if it decreases. If an absolute angle variation greater than a given threshold is detected, a rotation event is triggered according to direction and magnitude of roll of contact points. Whenever an event is triggered, the current positions of contact points are saved and the recognition engine starts a new recognition process. Thus, the user can perform any gesture sequences without lifting off its fingers: