May 15, 2017

Techniques for Disambiguation of Uncertain User Input

Philip Weaver

Maya Ben Ari

Techniques for Disambiguation of Uncertain User Input

ABSTRACT

This disclosure describes techniques to perform disambiguation of uncertain or inaccurate user inputs to a computing device. Uncertain input can occur, for example, by an inadvertent touching of the touchscreen of a smartphone in a user's pocket, inadvertent detection of gesture, etc. Current devices often interpret such input incorrectly, and respond in a manner that is unexpected for the user.

Techniques disclosed herein determine a user intent based on the detected gesture. Machine learning is used to disambiguate user-intended input from inadvertent input. The framework described herein detects input and triggers an analyzer. The analyzer receives data from various sources, based on user permission to access such data. The data can include a state of the device, device configuration, user context, sensor data, history, etc. Results from the analyzer are provided to a resolver that determines the user intent and performs an appropriate action, e.g., triggering an application on the device.

KEYWORDS

- User interface
- Gesture recognition
- Inadvertent input
- Uncertain input
- Machine-learning

BACKGROUND

Computing devices such as smartphone, tablets, wearable devices, gaming consoles, etc. permit users to provide input via gestures that are detected by the device, e.g., via a touchscreen display. Gesture inputs enable users to utilize a computing device without entering commands via a keyboard, a pointing device, etc. Touchscreens are sensitive to touch, e.g., can detect touch by one or more human fingers, hand, a stylus, etc.

Capacitive touchscreens are popular in devices such as smartphones and tablets. For example, a capacitive touchscreen includes an insulator such as glass that is coated with a transparent conductor. A human touch on the surface of the touchscreen is detected based on a distortion of an electrostatic field caused by the touch. Such distortion is measurable as a change in capacitance. The detected location of the touch is then processed by the computing device, e.g., to identify the gesture performed by the user.

Touchscreens can detect different types of gestures, e.g., a tap gesture, a drag gesture, a touch gesture performed using multiple fingers simultaneously, a pinch gesture, etc. Modern user interface techniques map gestures to operations on the device, e.g., in response to detecting a tap gesture on an icon, the detected operation is to launch an application that corresponds to the icon.

Inaccurate or ambiguous inputs are caused by gestures that do not correspond to a particular portion of the user interface, e.g., a tap gesture that is detected at a touchscreen location that is between two icons. Further, in certain situations, e.g., when the device is in a user's pocket, the input may be inadvertent. For example, such a situation may occur when the user is in an environment where accurately hitting a target on a screen is difficult, e.g., in a car. Further, some users are unable to perform touch operations with a required level of precision, e.g., due to age, disability (e.g., hand tremor, visual impairment, cognitive disability, etc.) and so on. Other gesture-based input techniques, such as head-tracking, face-tracking, point-scanning,

etc. can also receive inadvertent or imprecise input. Incorrect interpretation of received input can cause a device to perform an operation does not match the user's intent, or to not perform an operation intended by the user. Thus, the user needs to take additional actions, e.g., an additional tap, selecting from a menu, etc. to accomplish a task.

Current techniques to determine user intent when uncertain input is received include displaying a disambiguation screen and/or offering to modify settings that control sensitivity of input devices such as touchscreens, gesture recognitions applications, etc. Such techniques are static, e.g., rely on settings that are only manually changeable, make no use of context, and do not attempt to learn and adapt to user behavior or idiosyncrasies.

DESCRIPTION

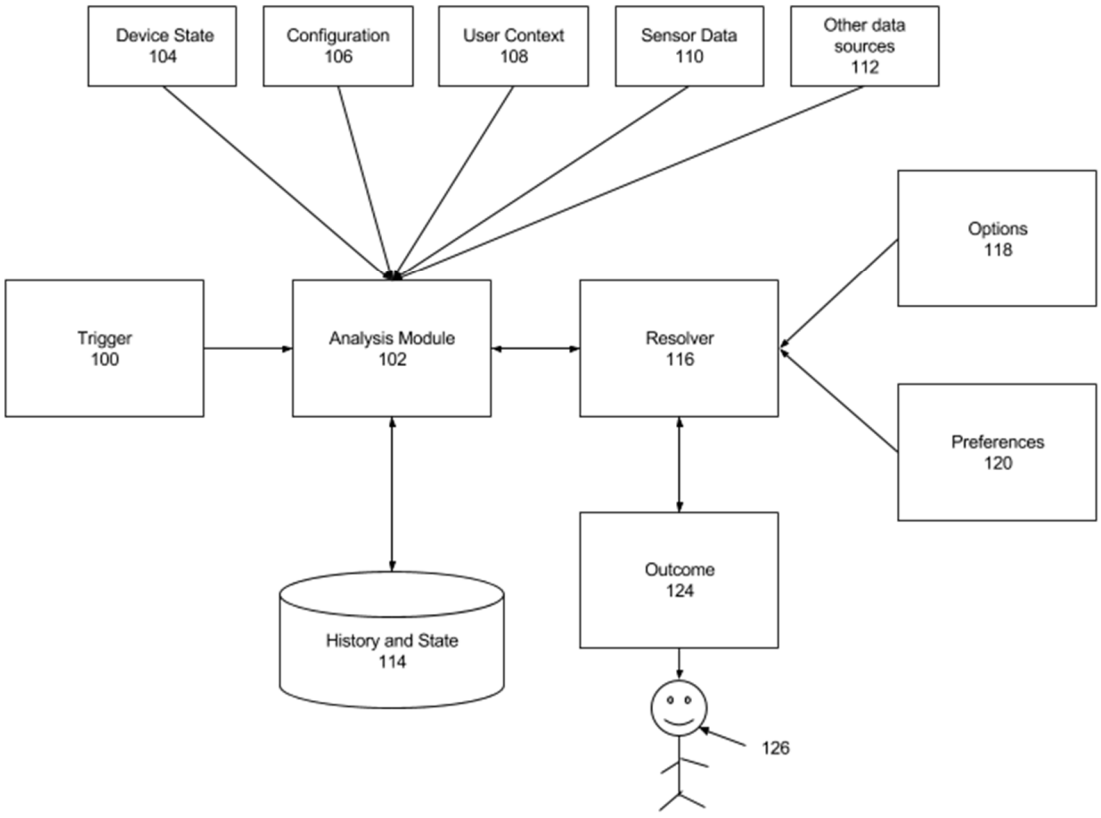


Fig. 1: Framework to disambiguate user input

Fig. 1 shows an example framework in which techniques to handle uncertain input, as described herein, are implemented. The framework can be implemented using software and/or hardware of any computing device, e.g., a mobile device. For example, the framework may be included as part of a mobile operating system.

As shown in Fig. 1, analysis module (102) receives a trigger (100), e.g., based on input from a user (126), or inadvertent input. Based on user permissions to access contextual information, the analysis module receives such information from various additional sources. For example, when permitted by the user, such contextual input can include one or more of device state (104), configuration (106), user context (108), sensor data (110), and data from other data sources (112). Only such contextual inputs as permitted by the users are utilized.

The analysis module analyzes the input, and stores history and state (114) in a repository. The analysis module provides analysis results to resolver (116). The resolver updates the analysis results based on options (118) and user preferences (120) to determine outcome (124). The user can optionally act upon the outcome, e.g., take action that confirms the validity of the outcome, or indicate that the outcome was not what was intended. Such user action serves as a learning update to the analysis module.

The trigger (100) is a user action or uncertain input. The trigger can include, for example, a discrete or continuous gesture, e.g., tap, swipe, drag and drop, pinch, etc., a location or path of gesture, e.g., points on a touchscreen that were touched, and a timing of the gesture, e.g., the beginning, end, or duration of user activity or uncertain input.

On receiving the trigger, the analysis module (102) performs analysis of data received from various sources such as device state, configuration, user context, sensor data, etc. Analysis is performed using one or more of machine learning, neural networks, mathematical modeling,

decision trees, rule sets, etc. The analysis module can operate in discrete mode, e.g., upon a user request for analysis, or in continuous mode, e.g., the analysis module runs in the background.

In continuous mode, the analysis module maintains a state. As the analysis module receives different inputs and triggers, it updates the state. The history and state (114) is stored in a repository. Results produced by the analysis can be actions or gestures that the analysis module determines as the user's intent, or suggestions or corrections to the trigger. In various implementations, the analysis module can provide the results with an associated confidence level that is determined based on the analysis of the trigger, e.g., the analysis module provides a result with confidence level 70%.

Analysis module 102 relies on inputs (104-112) to perform analysis. Device state (104) comprises the current device state, system events occurring within device, etc. based on user permissions to access system data. For example, device state includes the user-interface hierarchy and look-and-feel, status of applications (open or closed), battery levels etc. Configuration (106) comprises device settings, e.g., screen brightness, touch sensitivity, accessibility settings, connected Bluetooth devices, user accessibility settings (e.g., which accessibility services and features are installed or used), user's input mechanism type (e.g. head tracker, face tracker, mouse, direct touch), etc. as permitted by the user.

When a user provides consent to access contextual data for purposes of analysis, user context (108) comprises elements that the user recently tapped on, recently-performed user actions or activities, etc. User context is utilized to understand the most frequently used or preferred actions by the user in similar situations.

Sensor data (110) comprises signals that the user has permitted for use in the analysis and classification of user intent, e.g., touch screen (finger dynamic), GPS (location), clock (time) etc.

For example, sensor data is used to infer information about current user state, e.g., user is driving, device is in the pocket, single-hand usage, etc., or previously performed actions based on user's state, e.g., user habitually opens a specific application, e.g., mapping or navigation application, while in the car, etc.

Other data sources (112) include the state of different applications, local or cloud-based application data, etc. that are permitted by the user. The analysis module also utilizes information provided by the repository (114), e.g., relevant history and state information pertaining to the analysis.

The analysis module provides analysis results to resolver 116. The resolver looks up options (118) and user preferences (120) for any beneficial changes that can be made, for example in system settings, user-interface settings, user preferences, etc., and recommends a preferred outcome. For example, if the user tapped close to, but not quite exactly on, an icon, a possible outcome is to activate the icon as well as to increase a size of the icon. The resolver updates the analysis module on the outcome.

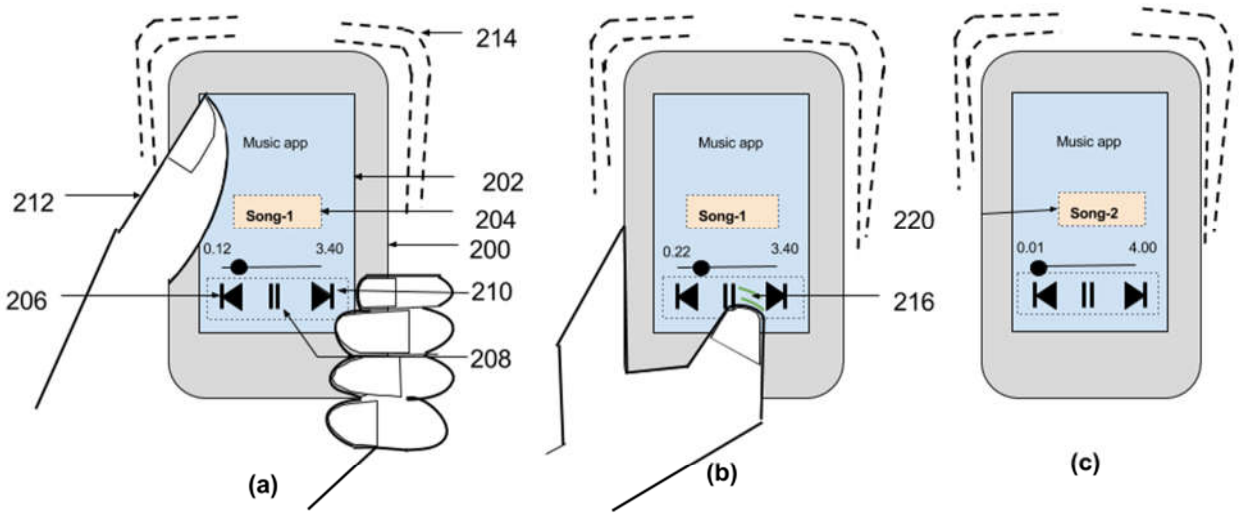
In determining an outcome, a corpus of options (118) is available for the resolver to choose from. Options include changing, for example, touch target size, screen layout, display and text size, touch sensitivity, dwell time for mouse/face tracker input, target position on the screen, etc. User preferences (120) are preferred settings for the device. For example, a user may indicate a preference for the device to change the touch target size rather than a location of the touch target.

The outcome (124) presented to the user by the resolver can be in various forms. For example, an outcome is to perform the user-intended action as determined by the framework e.g., activate an application associated with an icon. Another example is to modify user interface, e.g.,

increase touch target size, move elements closer to where the user tapped, etc. Other examples of actions include modifying settings on the device, e.g., touch sensitivity, brightness, display size, etc., displaying a menu of options/suggestions for the user to choose from, etc.

When permitted by the user, user selections are saved and updated in the analysis module. If the outcome as generated by the resolver turns out to not be user intent, the user can perform additional actions, e.g., cancel or nullify the outcome by tapping “cancel” or “go back” on a menu, etc. If the user consents to utilizing such user action data to update the analysis module, the data is used as a learning update for the analysis module.

The framework described in Fig. 1 disambiguates user-intended actions from inadvertent input. Based on usage by a user, the framework can learn from user behavior, and thus, an error rate of the framework reduces. With greater usage, the framework of Fig. 1 improves in its ability to discriminate better between user-intended actions and inadvertent input. Such a framework can determine user intent without having to perform a second action, e.g., provide a disambiguation menu, a request for confirmation of action (e.g., an additional tap), etc.



(a): Mobile device shaking in hand while user is in motion

(b): Inaccurate input while the cell phone is shaking

(c): Outcome of the framework handling the inaccurate input

Fig. 2: Illustrative implementation

Fig. 2 illustrates an example of a mobile device with the techniques of this disclosure. The framework, as disclosed, executes within the device (200), e.g., a smartphone with a touch screen. In Fig. 2(a), a smartphone is in an environment where it experiences vigorous back-and-forth movement (214), as detected by an on-board accelerometer. For example, such movement may be an indication that the user is running with the smartphone in hand (212). The framework can also determine, from inputs to the analysis module, that the user is listening to music through a music application.

As shown in Fig. 2(a), the device displays the music app (202) with a currently playing song (Song-1, 204). The music app includes navigational buttons, e.g., a "previous" button (206), a "pause" button (208) and a "next" button (210). The user provides input, e.g., to skip the current song by tapping "next." However, due to the movement of the device as the user is running, and due to single-handed usage, the input may be received as a tap gesture in a region between the pause and the next buttons (216), as shown in Fig. 2 (b).

The framework analyses the user input to determine that skipping to the next song is the likely user intent. For example, such determination may be made based on data inputs from various sources, e.g., historical data relating to user activities, user behavior patterns vis-a-vis the music application, device state, user context, recent user activity, user preferences, etc. that are available, based on user consent, to the framework.

Fig, 2 (c) illustrates an outcome of the analysis. The music app skips over to song-2 (220) (e.g., instead of pausing song-1) based on a determination that the user intended to select the "next" button. Further, the device may present additional options to the user, e.g., an option to reduce the phone's touch sensitivity, e.g., when the framework detects that the device is experiencing movement. User selection or rejection of the presented options may be additional input to the analysis module.

In situations in which certain implementations discussed herein may collect or use personal information about users (e.g., user data, information about a user's social network, user's location and time at the location, user's biometric information, user's activities and demographic information), users are provided with one or more opportunities to control whether information is collected, whether the personal information is stored, whether the personal information is used, and how the information is collected about the user, stored and used. That is, the systems and methods discussed herein collect, store and/or use user personal information specifically upon receiving explicit authorization from the relevant users to do so. For example, a user is provided with control over whether programs or features collect user information about that particular user or other users relevant to the program or feature. Each user for which personal information is to be collected is presented with one or more options to allow control over the information collection relevant to that user, to provide permission or authorization as to

whether the information is collected and as to which portions of the information are to be collected. For example, users can be provided with one or more such control options over a communication network. In addition, certain data may be treated in one or more ways before it is stored or used so that personally identifiable information is removed. As one example, a user's identity may be treated so that no personally identifiable information can be determined. As another example, a user's geographic location may be generalized to a larger region so that the user's particular location cannot be determined.

CONCLUSION

The techniques described herein enable disambiguating inadvertent inputs to a device, to determine user intent. Inadvertent input may arise from the particular environment the mobile device is situated in, due to user inability to properly access and activate input sensors, etc. The techniques to disambiguate user input utilize machine-learning to analyze user input, e.g., gestures, voice commands, or touches of a touchscreen. The techniques determine an outcome, e.g., perform the user-intended action such as activating an application, modify user interface e.g., increase touch target size, modify one or more device settings, e.g., touch sensitivity, or display a menu of options, etc. If permitted by the user, a subsequent user action is used to improve the disambiguation, e.g., as training data. Techniques of this disclosure provide a robust user experience for users with different accessibility needs under a wide range of environments.