March 2021

TWO-HANDED TYPING METHOD ON AN ARBITRARY SURFACE

Shumin Zhai

Mingrui Zhang

TWO-HANDED TYPING METHOD ON AN ARBITRARY SURFACE

ABSTRACT

A computing device may detect user input, such as finger movements resembling typing on an invisible virtual keyboard in the air or on any surface, to enable typing. The computing device may use sensors (e.g., accelerometers, cameras, piezoelectric sensors, etc.) to detect the user's finger movements, such as the user's fingers moving through the air and/or contacting a surface. The computing device may then decode (or, in other words, convert, interpret, analyze, etc.) the detected finger movements to identify corresponding inputs representative of characters (e.g., alphanumeric characters, national characters, special characters, etc.). To reduce input errors, the computing device may decode the detected finger movements, at least in part, based on contextual information, such as preceding characters, words, and/or the like entered via previously detected user inputs. Similarly, the computing device may apply machine learning techniques and adjust parameters, such as a signal-to-noise ratio, to improve the accuracy of input-entry. In some examples, the computing device may implement specific recognition, prediction, and correction algorithms to improve the accuracy of input-entry. In this way, the computing device may accommodate biasing in finger movements that may be specific to a user entering the input.

DESCRIPTION

The present disclosure describes detecting user input, such as finger movements resembling typing on an invisible virtual keyboard in the air or on any surface, to enable typing. For example, the computing device may use sensors (e.g., accelerometers, cameras, piezoelectric sensors, etc.) to detect the user's finger movements, such as the user's fingers moving in the air and/or contacting a surface. The computing device may then decode (or, in other words, convert, interpret, analyze, etc.) the detected finger movements to identify corresponding inputs representative of characters e.g., alphanumeric characters, national characters, special characters, etc.).

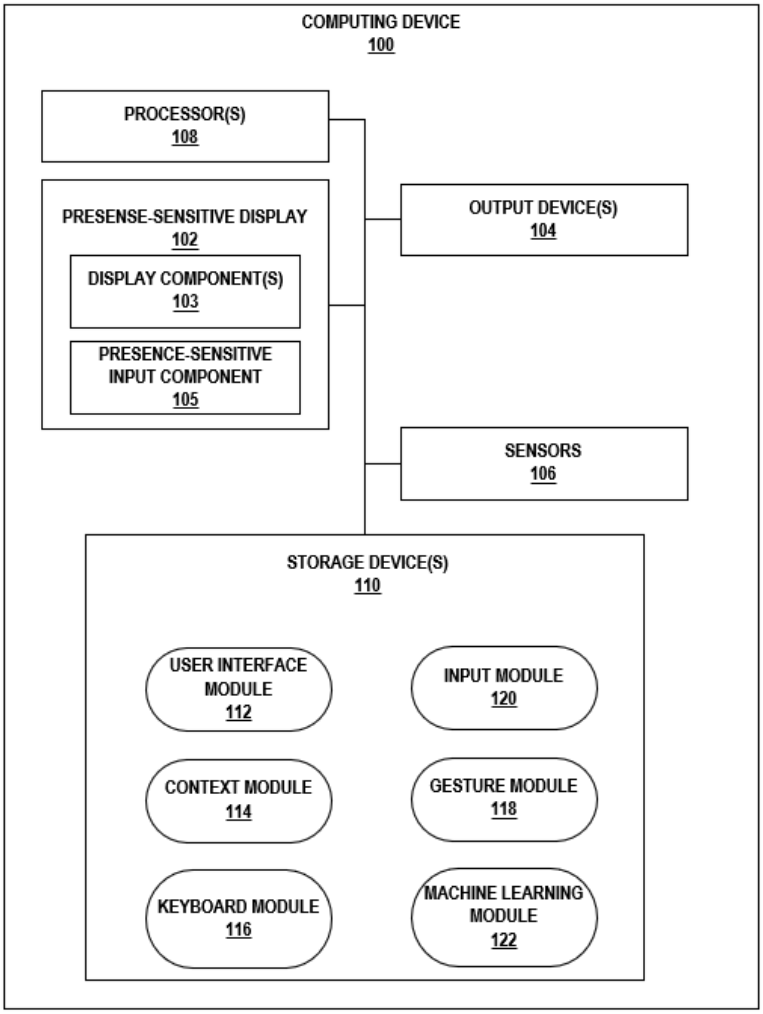

FIG. 1

FIG. 1 is a conceptual diagram illustrating a computing device 100 that detects user input, such as finger movements resembling typing on an invisible virtual keyboard in the air or on any surface, to enable typing. Computing device 100 may be any mobile or non-mobile computing device, such as a cellular phone, a smartphone, a personal digital assistant (PDA), a desktop computer, a laptop computer, a tablet computer, a portable gaming device, a portable media player, an e-book reader, a watch (including a so-called smartwatch), an add-on device (such as a casting device), smart glasses, a gaming controller, and/or the like. As illustrated by FIG. 1, computing device 100 may include a presence-sensitive display 102, one or more output devices 104, one or more sensors 106, one or more processors 108, and one or more storage devices 110. Storage devices 110 may include a user interface (UI) module 112, a context module 114, a keyboard module 116, a gesture module 118, input module 120, and a machine learning module 122.

Presence-sensitive display 102 of computing device 100 may be implemented using various display hardware. In some examples, presence-sensitive display 102 may function as an input device using a presence-sensitive input component 105, such as a resistive touchscreen, a surface acoustic wave touchscreen, a capacitive touchscreen, a projective capacitance touchscreen, a pressure sensitive screen, an acoustic pulse recognition touchscreen, or another presence-sensitive display technology. Additionally, presence-sensitive display 106 may function as an output device 104 using any one or more display components 103, such as a liquid crystal display (LCD), dot matrix display, light emitting diode (LED) display, organic light-emitting diode (OLED) display, e-ink, or similar monochrome or color display capable of outputting visible information to a user of computing device 100.

Processors 108 may implement functionality and/or execute instructions associated with computing device 100. Examples of processors 108 may include one or more of an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), an application processor, a display controller, an auxiliary processor, a central processing unit (CPU), a graphics processing unit (GPU), one or more sensor hubs, and any other hardware configure to function as a processor, a processing unit, or a processing device. UI module 112, context module 114, keyboard module 116, gesture module 118, input module 120, and machine learning module 122 may be operable (or, in other words, executed) by processors 108 to perform various actions, operations, or functions of computing device 100. For example, modules 112-122 may form executable bytecode, which when executed, cause processors 108 to perform specific operations in accordance with various aspects of the techniques described herein.

Storage devices 110 may include one or more computer-readable storage media. For example, storage devices 110 may be configured for long-term, as well as short-term storage of information, such as, e.g., instructions, data, or other information used by computing device 100. In some examples, storage devices 110 may include non-volatile storage elements. Examples of such non-volatile storage elements include magnetic hard discs, optical discs, solid state discs, floppy discs, flash memories, forms of electrically programmable memories (e.g., EPROMs), or electrically erasable and programmable memories (e.g., EEPROMs), as well as other forms of non-volatile memories known in the art.

In other examples, in place of, or in addition to the non-volatile storage elements, storage devices 110 may include one or more so-called “temporary” memory devices, meaning that a primary purpose of these devices may not be long-term data storage. For example, the devices

may comprise volatile memory devices, meaning that the devices may not maintain stored contents when the devices are not receiving power. Examples of volatile memory devices include random access memories (RAM), dynamic random access memories (DRAM), static random access memories (SRAM), and other forms of volatile memories, or memory devices, known in the art. In some examples, the devices may store program instructions for execution by processors 108. For example, the devices may be used by modules 112-122 executing on computing device 100 to temporarily store information during program execution.

Some users of computing devices desire the ability to type on computing devices without using a physical keyboard or a visible virtual keyboard (which may be referred to as a "soft keyboard" configured to enable alphanumeric text entry on present computing devices, such as smartphones). While efforts have been made to create systems for enabling typing without a physical keyboard, some of these systems require that users learn typing methods different from those used on a traditional physical keyboard (e.g., a physical QWERTY keyboard) or soft keyboard. For example, some of these systems may require a user to perform specific gestures with one or more fingers to input corresponding characters of text. Such specific gestures may bear little (and possibly no) resemblance to typing on a traditional keyboard, which may make using such systems difficult for a user to comfortably and quickly enter alphanumeric text.

Rather than implement convoluted input mechanisms (e.g., gestures) for entering alphanumeric text that rely on users becoming proficient in character entry with which such users are unaccustomed (e.g., where such gestures do not mimic input entry via a keyboard, mouse, or other typical input entry device), various aspects of the techniques described in this publication enable computing device 100 to detect user input, such as finger movements resembling typing on an invisible virtual keyboard in the air or on any surface, to enable typing. That is, input

module 120 may output (e.g., via presence-sensitive display 102) text based on finger movements decoded (or, in other words, converted, interpreted, analyzed, etc.) by gesture module 118.

For instance, keyboard module 116 of computing device 100 may map finger movements detected by sensors 106 to corresponding characters. Finger movements may include any movement of any of a user's fingers, such as a downward movement (e.g., swiping the air, tapping a surface, etc.) of the user's right index finger. This downward movement of the user's right index finger may correspond to the alphanumeric character "J" (e.g., relative to an invisible virtual keyboard). Accordingly, gesture module 118 may map this downward movement of the user's right index finger to the alphanumeric character "J." Gesture module 118 may similarly map each finger movement onto a corresponding character.

In some examples, sensors 106 may be accelerometers worn on each finger of the user's hands. In such examples, gesture module 118 may decode (e.g., convert, interpret, analyze, etc.) the acceleration data of the user's finger movements to determine characters to be entered by computing device 100. For example, gesture module 118 may determine the acceleration vectors of the user's fingers and use the acceleration vectors as spatial input data for converting the finger movements into selection of characters for input. In this way, gesture module 118 may use the spatial input data to map the user's finger movements to corresponding characters for input-entry into computing device 100. In some examples, gesture module 118 may implement specific recognition, prediction, and correction algorithms to improve the accuracy of input-entry. In this way, the computing device may accommodate biasing in finger movements that may be specific to a user entering the input.

To reduce typing errors, context module 114 may generate contextual information, such as preceding alphanumeric characters, words, punctuation, and/or the like entered via previously detected user inputs, that input module 120 may use to predict the text the user intends to enter. In this way, input module 120 may decode the finger movements based, at least in part, on the contextual information. For example, if the user enters the text "The French revo," context module 114 may generate contextual information by performing a lookup in a database (e.g., stored locally and/or remotely) for results that have a degree of likelihood (e.g., based on the preceding alphanumeric characters, words, punctuation, etc.) , such as "French," revo," etc.) of being the text the user intends to enter. In such an example, the lookup may produce results, such as "revolution" (e.g., "The French revolution"), "revolutionary," "revolt," and/or the like. The results may be updated (e.g., after a character is entered, a word is completed, punctuation is entered, etc.).

Input module 120 may create a ranking of the results. The rank of each result may indicate a degree of likelihood or a confidence score that the user intends to enter that result. The ranking may be updated (e.g., after an alphanumeric character is entered, a word is completed, punctuation is entered, etc.). Input module 120 may then select one of the candidate inputs (e.g., the candidate input with the highest degree of likelihood of being the text the user intends to enter) for input-entry even if the user enters a different alphanumeric character, word, punctuation, and/or the like (e.g., a typing error) based on the user's finger movements.

For example, if the user enters "The French revp," context module 114 may perform a lookup that still produces results, such as "revolution" (e.g., "The French revolution"), "revolutionary," "revolt," and/or the like, because the degree of likelihood that the user intends to enter "The French revp" is low, and the degree of likelihood that the user intends to enter "The

French revo" (e.g., in order to enter "The French revolution," "The French revolutionary,", etc.) is high. Responsive to additional user input (e.g., the user completing the word, tapping the location of display 102 displaying the text, making a completion gesture that completes the text as suggested, etc.), input module 120 may select the candidate input with the highest degree of likelihood of being the text the user intends to enter (e.g., "revolution") for input-entry even if the user enters "revplution" or some other typing error.

In some examples, machine learning module 122 may operate in conjunction with one or more of modules 112-120. Computing device 100 may store and/or execute machine learning module 122. However, although shown as executed by computing device 100 in the example of FIG. 1, machine learning module 122 may be stored on and/or executed by a remote computing system (e.g., a cloud computing system). Machine learning module 122 may represent one or more artificial neural networks (also referred to as neural networks). A neural network may include a group of connected nodes, which also may be referred to as neurons or perceptrons. A neural network may be organized into one or more layers. Neural networks that include multiple layers may be referred to as "deep" networks. A deep network may include an input layer, an output layer, and one or more hidden layers positioned between the input layer and the output layer. The nodes of the neural network may be connected or non-fully connected.

Machine learning module 122 may operate in conjunction with gesture module 118 to perform functions described herein as being associated with gesture module 118. For example, machine learning module 122 may perform classification to decode (e.g., convert, interpret, analyze, etc.) the acceleration data of the user's finger movements to determine characters to be entered by computing device 100. Machine learning module 122 may perform classification by providing, for each of one or more classes, a numerical value descriptive of a degree to which it

is believed that the input data should be classified into the corresponding class. The numerical values provided by machine learning module 122 may be confidence scores (e.g., for decoding the user's finger movements) that, as described above, are indicative of a respective confidence associated with classification of the input into the respective class.

For example, machine learning module 122 may receive as input data the acceleration vector for a downward movement of the user's right index finger. Machine learning module 122 may be trained (e.g., based on training data) to determine spatial input data (e.g., the finger movement relative to an invisible virtual keyboard) that in turn corresponds to the alphanumeric character "J." Accordingly, machine learning module 122 may map this downward movement of the user's right index finger (based on the acceleration data of this finger movement) onto the alphanumeric character "J."

Similarly, machine learning module 122 may operate in conjunction with input module 120 to determine the degree of likelihood for each candidate input. For example, machine learning module 122 may be trained (e.g., based on training data) to create, based on contextual information (e.g., preceding alphanumeric characters, words, punctuation, etc.) a ranking of results from a lookup in a database. That is, machine learning module 122 may be trained to determine, for each result, a degree of likelihood or a confidence score that the user intends to enter that result. Machine learning module 122 may then determine the candidate input with the highest degree of likelihood of being the text the user intends to enter and enter that candidate input.

In some examples, UI module 112 may cause presence-sensitive display 102 to display a suggestion strip within a graphical user interface (GUI). The suggestion strip may include one or more candidate inputs that input module 120 predicts will be the text the user intends to enter. In

some examples, the candidate inputs may include a primary candidate input. The primary candidate input may be the candidate input with the highest degree of likelihood of being the text the user intends to enter.

The user may review and/or interact with the candidate inputs in the suggestion strip (e.g., directly via presence-sensitive display 102, indirectly via finger movements detected by sensors 106, etc.). For example, responsive to the user entering "revo," the suggestion strip may include candidate inputs "revolution," "revolutionary," and "revolt." The user may select the primary candidate input (e.g., "revolution") to replace "revo" with the primary candidate input, thereby completing the text "revo." In some examples, the primary candidate input may initially and/or always be located in a specific region of the suggestion strip.

If there are too many candidate inputs for the suggestion strip to include in a manner suitable to the user (e.g., without making the candidate inputs so small that the candidate inputs are unintelligible), the suggestion strip may include some, but not all, candidate inputs for display. However, the suggestion strip may visually indicate to the user the existence of additional candidate inputs not currently displayed. For example, the suggestion strip may include a truncated candidate input (e.g., the first few characters of a candidate input).

In some examples, instead of collecting spatial input data, computing device 100 may collect index data. The index data may identify which finger the user is moving to type and the sequence of those finger movements. UI module, context module 114, keyboard module 116, gesture module 118, input module 120, and machine learning module 122 may then operate in a manner similar, if not substantially similar, to that described above to enable typing. In some examples, computing device 100 may adjust parameters, such as a signal-to-noise ratio, to improve the accuracy of input-entry.

One or more advantages of the techniques described in this disclosure include enabling a user to input characters (alphanumeric characters, national characters, special characters, etc.) via finger movements instead of with a traditional physical keyboard, which may be particularly beneficial when typing on a mobile computing device (e.g., smartphone, tablet, smartwatch, etc.) and/or devices with newer form factors (e.g., foldable devices, computing devices with rollable displays, etc.). Another advantage includes reducing typing errors by using contextual information, machine learning, and/or the like, improving the user experience. Yet another advantage includes presenting the user with a plurality of candidate inputs, which increases the degree of likelihood that the user will ultimately interact with one of the suggestion inputs, thereby improving the speed, efficiency, and accuracy of typing.

It is noted that the techniques of this disclosure may be combined with any other suitable technology or combination of technologies, including those listed as references below.

References

1. US Patent Application No. 2020/0050973
2. US Patent Application No. 2015/0084884
3. US Patent Application No. 2013/0314352
4. US Patent Application No. 2014/0201671
5. Suwen Zhu et al., “Typing on an invisible keyboard.” Conference on Human Factors in Computing Systems, April 2018.
6. Jain, “Remulation-based keyboard evaluation system: A review paper.” International Journal for Research in Applied Science & Engineering, June 2015.