Technical Disclosure Commons

Defensive Publications Series

July 22, 2019

DETECTING GESTURES UTILIZING MOTION SENSOR DATA AND MACHINE LEARNING

Michael Xuelin Huang

Shumin Zhai

DETECTING GESTURES UTILIZING MOTION SENSOR DATA AND MACHINE LEARNING

ABSTRACT

A computing device is described that uses motion data from motion sensors to detect gestures or user inputs, such as out-of-screen user inputs for mobile devices. In other words, the computing device detects gestures or user touch inputs at locations of the device that do not include a touch screen, such as anywhere on the surface of the housing or the case of the device. The techniques described enable a computing device to utilize a standard, existing motion sensor (e.g., an inertial measurement unit (IMU), accelerometer, gyroscope, etc.) to detect the user input and determine attributes of the user input. Motion data generated by the motion sensor (also referred to as a movement sensor) is processed by an artificial neural network to infer attributes of the user input. In other words, the computing device applies a machine-learned model to the motion data (also referred to as sensor data or motion sensor data) to classify or label the various attributes, characteristics, or qualities of the input. In this way, the computing device utilizes machine learning and motion data to classify attributes of the user input or gesture utilizing motion sensors without the need for additional hardware, such as touch-sensitive devices and sensors.

DESCRIPTION

Techniques are described that enable a computing device to detect gestures performed on a surface of a computing device that is not touch-sensitive, such as the back of the computing device, using data from one or more motion sensors. Motion sensors of the computing device may detect changes in the acceleration of the device caused by the user sliding his or her finger over a surface of the device. For example, the user input may cause the computing device to vibrate and the vibrations may be detected by the motion sensors. The computing device may

differentiate between gesture directions (e.g., up, down, left, right) and location of the gesture based on changes in the motion data generated by one or more motion sensors (e.g., an accelerometer and/or a gyroscope) as the gesture is being performed.

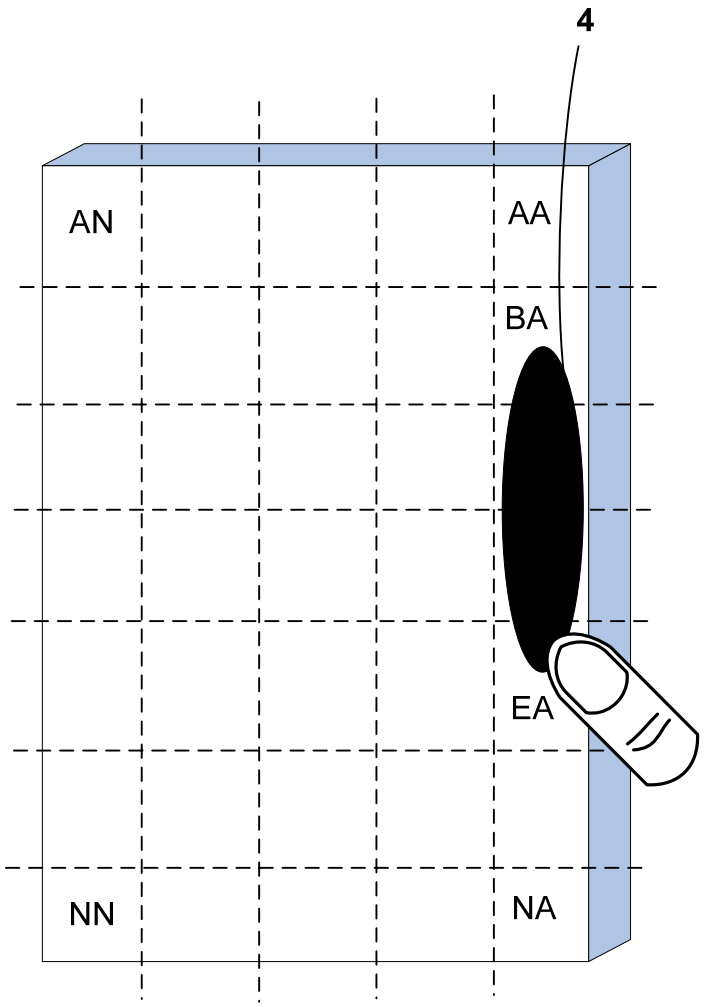

FIG. 1

In the example of FIG. 1, a computing device is configured to detect user inputs on the front, back, and/or sides of the computing device based at least in part on data generated by one or more motion sensors. Examples of computing devices include, but are not limited to, smartphones, tablets, watches, fitness trackers (e.g., wrist-worn, ankle-worn, waist-worn, or other devices worn by users that track various movement or other athletic activity), counter top

computing devices, laptops, or any other computing device. The computing device includes a touch-sensitive device (e.g., a touchscreen) on the front, one or more processors, and one or more motion sensors. Examples of processors include, but are not limited to, digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Examples of motion sensors include an accelerometer, a gyroscope, an inertial measurement unit (IMU), a magnetometer, among others.

In some examples, the front of the computing device may include a touch-sensitive device while the back of computing device does not include a touch-sensitive device. Examples of touch-sensitive devices include touchscreens and touchpads, among others.

The surfaces of the housing of the computing device may be made of a material having a coefficient of friction above a threshold level. For instance, the back and/or sides of the housing may be made of rubber, plastic, fabric, paper, or any other material with a relatively high coefficient of friction (e.g., compared to glass or a smooth metal). In one instance, the computing device includes a case that covers at least a portion of the housing. In such instances, the case may be made of a material with a relatively high coefficient of friction.

A user of the computing device may perform a gesture on the front, back, and/or sides of the computing device. In other words, the user may perform a gesture or user input anywhere along the housing or case of the computing device. As one example, the user may perform a swipe, pinch, tap, double-tap, or any other type of gesture at the back of the computing device (e.g., using his/her finger, a stylus, or other input mechanism that makes physical contact with the computing device). A user of the computing device may perform a gesture 4 by sliding his or her finger along the back surface of the housing of the computing device. As illustrated in FIG. 1, the computing device may be logically divided into a grid of grid elements AA-NN. Computing device may detect the gesture as the gesture begins at a portion of the housing

corresponding to logical grid element BA and continues through a portion of the housing corresponding to logical grid element EA.

The computing device includes one or more motion sensors configured to detect motion of the computing device as the input device contacts the back of computing device while performing gesture 4. When the material for the housing has a relatively high coefficient of friction (e.g., compared to glass or a smooth metal), a user input device (e.g., a finger, a stylus, etc.) may briefly stick to the surface and then slide across the surface for a distance, then stick and slip, as the user applies a force to the housing surface via the input device. This repeated stick and slip motion may cause vibrations that may be detected by the motion sensor.

In one example, the computing device includes an IMU, which may include a 3-axis accelerometer and a 3-axis gyroscope. In such examples, the IMU outputs motion data with six values (e.g., an acceleration value for each of the three axes and a gyroscope value for each of the three axes) each time the IMU samples the motion of the computing device. The computing device may store the motion data as one-dimensional vector (or as any other type of data structure) with six sensor values or elements for each time the IMU samples the motion. The IMU may sample the motion every 1 millisecond, every 2 milliseconds, every 5 milliseconds, and so on. In some examples, the vector includes data for multiple samples (e.g., 10, 20, 50, 100 samples). In this way, the vector may include six sensor values for each of the plurality of samples.

In accordance with techniques of this disclosure, the computing device applies a machine-learned model to the motion data to determine the attributes or characteristics of the user input. The model may be trained via supervised or unsupervised learning. In some examples, the model includes a neural network, such as a convolutional neural network (CNN). Additional examples of machine learning algorithms include clustering algorithms, decision-tree algorithms, regression algorithms, as only a few examples. In one example, the computing

device may apply the vector of motion data to the CNN. In other words, the input to the CNN includes, in some examples, a one-dimensional vector of motion data over time (e.g., the vector may include six values for each time the IMU is sampled within a period of time).

The computing device may apply a multi-layer CNN to the input vector. For example, the CNN may include 2, 3, 4, or more layers. The layers may be partially or fully connected. For each convolution layer, the computing device applies a filter to the input for that layer. As one example, the computing device may multiply the one-dimensional input vector (also referred to as a matrix) by a convolution filter and output a convoluted vector (or matrix) for the first convolution layer. The computing device may input the convoluted vector output by the first layer as an input to the second layer, apply another filter, and output a new convoluted vector, and so on for each convolution layer of the CNN.

The computing device may determine a plurality of attributes of the user input based on the CNN. In other words, in one example, the CNN receives the one-dimensional vector (e.g., with six values for each sample of the IMU) as the input and outputs a plurality of values representative of the attributes of the user input. In some examples, the attributes include a direction of the user input and a location of the user input.

In some examples, because different attributes of the user input may affect the motion data simultaneously, the computing device may apply the multi-layer CNN to jointly learn the mappings from motion data onto these interrelated attributes in a multi-task learning paradigm. In one example, the computing device determines the different attributes of the user input by utilizing one or more shared convolutional layers, which may enable the computing device to detect or extract common patterns that are indicative across user input attributes. Following the shared layers, in some examples, the CNN includes individual layers for each attribute of the user input to extract the attribute-dependent patterns.

In some examples, one or more outputs of the CNN include classifications of the attributes or characteristics of the input. For example, the computing device may classify the direction of the gesture or user input. In the example of FIG. 1, one output of the CNN classifies the direction of the user input as down (or top to bottom). As another example, another output of the CNN classifies the locations of the user input. For example, the output may classify the starting location of gesture 4 as logical grid element BA and the ending location of gesture 4 as logical grid element EA. In another example, the output of the CNN includes a regression output. For example, the output of the CNN may include a specific location of the user input (e.g., rather than a grid element).

The computing device may perform one or more actions based on the output of the CNN. In one example, the computing device controls a pointer or cursor based on the gesture. For example, the computing device may move the cursor around the screen as the user moves his or her finger along the back of the computing device. The computing device may scroll (e.g., up, down, left, right) through a graphical user interface in response to detecting a scrolling gesture. As another example, the computing device may select different targets (e.g., icons) based on the gesture. The computing device may adjust a slider based on the scrolling gesture.

In some instances, the computing device adjusts the volume of a speaker in response to detecting a scrolling gesture near or on the edge or side of the computing device. For instance, the computing device may increase the volume in response to detecting a scrolling gesture in one direction (e.g., towards the top of the device) and decrease the volume in response to detecting a scrolling gesture in another direction (e.g., towards the bottom of the device). In one instance, the computing device controls video playback based on the scrolling gesture. For instance, the computing device may rewind the video in response to detecting a scrolling gesture in one direction (e.g., to the left) and may advance the video forward in response to detecting a scrolling gesture in another direction (e.g., to the right). In another instance, the computing device adjusts

a camera zoom level based on the gesture. For instance, the computing device may increase the zoom level of the camera (e.g., zoom in) in response to detecting a scrolling gesture in one direction (e.g., towards the top of the device) and decrease the zoom level of the camera (e.g., zoom out) in response to detecting a scrolling gesture in another direction (e.g., towards the bottom of the device).

By applying a machine-learned model (such as a CNN model) to the motion data, the computing device may determine one or more attributes of a touch input using existing motion sensors. Utilizing motion sensors to detect touch inputs may enable the computing device to determine attributes of touch inputs without utilizing a touch-sensitive device, such as a touchscreen or touchpad. Detecting user inputs and determining the attributes of the input without utilizing a touch-sensitive device may reduce the cost of the computing device and/or reduce the power consumed by the device, for example, by detecting user inputs without activating a touch-sensitive device. Further, detecting user inputs without utilizing a touch-sensitive device may enable the computing device to detect user inputs at locations of the computing device that do not include a touch-sensitive device, which may open the door to new functionality of the computing device.

REFERENCES

1. Le et al., *InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones,* User Interface Software and Technology Symposium 2018 (UIST '18), October 14-17 2018, DOI 10.1145/3242587.3242605
2. De Luca et al., *Back-of-device Authentication on Smartphones,* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13), April 27-May 2, 2013, DOI: 10.1145/2470654.2481330.
3. Le et al., *Investigating Screen Shifting Techniques to Improve One-Handed Smartphone Usage*, Proceedings of the 9th Nordic Conference on Human-Computer Interaction (NordiCHI'16), October 23-27, 2016, DOI 10.1145/2971485.2971562.
4. Corsten et al., *BackXPress: Using Back-of-Device Finger Pressure to Augment Touchscreen Input on Smartphones*. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17), May 6-11, 2017, DOI 10.1145/3025453.3025565.
5. Holman et al., *Unifone: Designing for Auxiliary Finger Input in One-handed Mobile Interactions,* Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13), February 10-13, 2013, DOI 10.1145/2460625.2460653
6. Wong et al., *Back-Mirror: Back-of-Device One-Handed Interaction on Smartphones*, Proceedings of the SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications Article No. 10 (SA'16), December 5-8, 2016 DOI 10.1145/2999508.2999522.
7. Sun et al., *VSkin: Sensing Touch Gestures on Surfaces of Mobile Devices Using Acoustic Signals,* Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom'18), October 29 - November 02, 2018, DOI 10.1145/3241539.3241568.

8. Tung et al., *Expansion of Human–Phone Interface By Sensing Structure-Borne Sound Propagation,* Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'16), June 26-30, 2016 DOI 10.1145/2906388.2906394.
9. Reyes et al., *SynchroWatch: One-Handed Synchronous Smartwatch Gestures Using Correlation and Magnetic Sensing*, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT'18), December 2017, DOI 10.1145/3161162
10. Chen et al., *Finexus: Tracking Precise Motions of Multiple Fingertips Using Magnetic Sensing,* Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16.), May 07-12, 2016, DOI 10.1145/2858036.2858125
11. Zhou et al., *AuraSense: Enabling Expressive Around-Smartwatch Interactions with Electric Field Sensing,* Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST'16), October 16-19, 2016, DOI 10.1145/2984511.2984568
12. Zhang et al., *Electrick: Low-Cost Touch Sensing Using Electric Field Tomography,* Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17), May 6-11, 2017, DOI 10.1145/3025453.3025842
13. Seipp et al., *BackPat: One-Handed Off-Screen Patting Gestures,* Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services (MobileHCI'14), September 23-26, 2014, DOI 10.1145/2628363.2628396
14. Zhang et al., *BackTap: Robust Four-Point Tapping on the Back of an Off-the-shelf Smartphone,* Proceedings of the adjunct publication of the 26th annual ACM symposium on User interface software and technology (UIST'13), October 8-11, 2013, DOI 10.1145/2508468.2514735
15. Liang et al., *Deep Learning Based Inference of Private Information Using Embedded Sensors in Smart Devices*, IEEE Network, July 2018, DOI 10.1109/MNET.2018.1700349

16. Pan et al., *Surfacevibe: vibration-based tap & swipe tracking on ubiquitous surfaces,* Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'17), April 18-20, 2017, DOI 10.1145/3055031.3055077
17. Zhang et al., *BeyondTouch: Extending the Input Language with Built-in Sensors on Commodity Smartphones*, International Conference on Intelligent User Interfaces, Proceedings IUI, March 2015, DOI 10.1145/2678025.2701374.