Technical Disclosure Commons

Defensive Publications Series

October 02, 2017

Machine Learning for Gesture Recognition in a Virtual Environment

Sandro Feuz

Pedro Gonnet Anders

Machine Learning for Gesture Recognition in a Virtual Environment

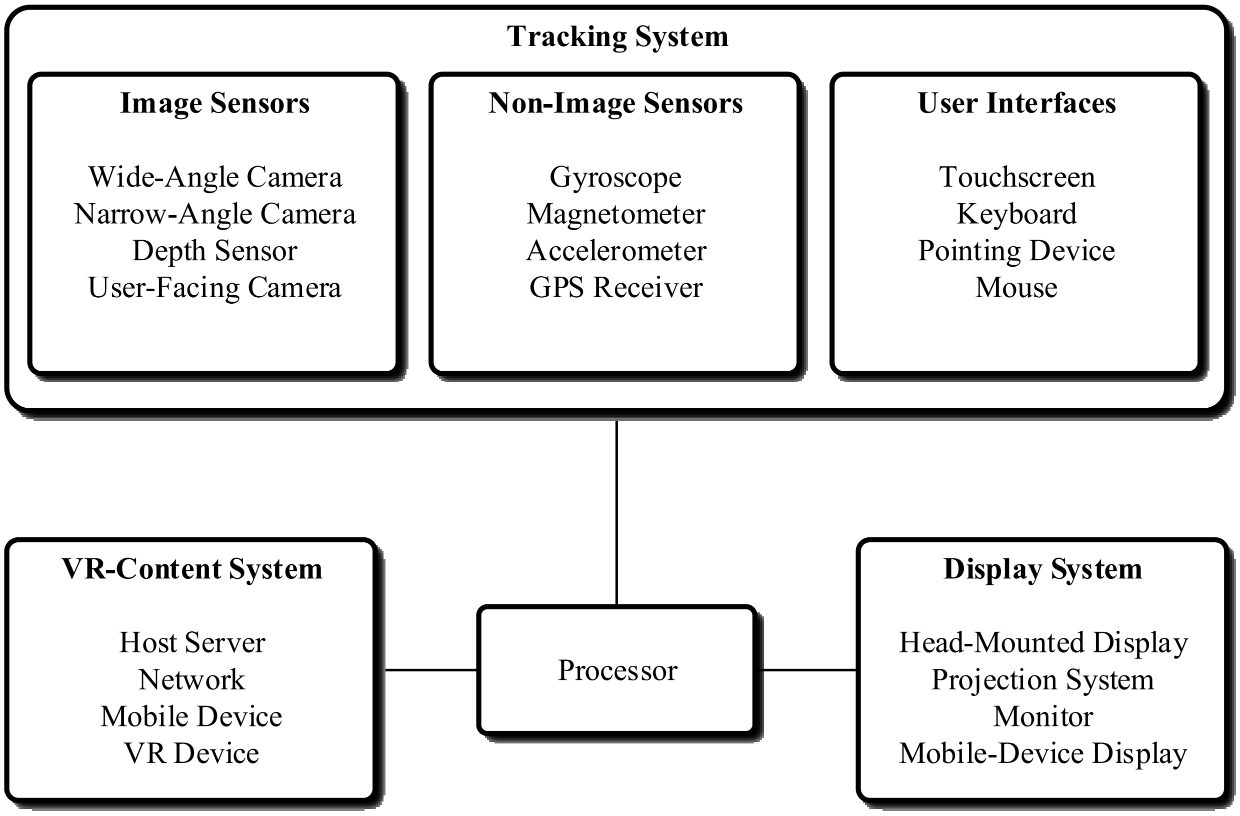
Abstract:

A machine-learning technique is provided that allows a VR system to automatically learn user gestures based on a ground-truth data set collected from real users. The technique includes two steps. First, ground-truth data is collected by observing multiple users intentionally performing a specified action in a virtual environment. For example, an action to move an object from one place to another is recorded through input from different sensors in the VR system (*e.g.*, position, orientation, controller actuations, or force/acceleration data). Second, machine-learning techniques (*e.g.*, a recurrent neural network, a feedforward neural network, or a Hidden Markov Model) are used to allow the VR system to learn to recognize user gestures intended to represent the actions. The system frees developers from having to custom define each gesture and provides users with accurate responses to natural movements.

Keywords: machine-learning, gesture recognition, virtual reality, VR, ground-truth data, control, recurrent neural networks, RNN, fixed heuristics, Hidden Markov Model, HMM, feedforward neural network, feature extraction

Background:

Virtual reality (VR) environments rely on display, tracking, and VR-content systems. Through these systems, realistic images, sounds, and sometimes other sensations simulate a user's physical presence in an artificial environment. Each of these three systems are illustrated below in Fig. 1.



VR System

Fig. 1

The systems described in Fig. 1 may be implemented in one or more of various computing devices that can support VR applications, such as servers, desktop computers, VR goggles, computing spectacles, laptops, or mobile devices. These devices include a processor that can manage, control, and coordinate operations of the display, tracking, and VR-content systems. The devices also include memory and interfaces. These interfaces connect the memory with the systems using various buses and other connection methods as appropriate.

The display system enables a user to “look around” within the virtual world. The display system can include a head-mounted display, a projection system within a virtual-reality room, a monitor, or a mobile device’s display, either held by a user or placed in a head-mounted device.

The VR-content system provides content that defines the VR environment, such as images and sounds. The VR-content system provides the content using a host server, a network-based device, a mobile device, or a dedicated virtual reality device, to name a few.

The tracking system enables the user to interact with and navigate through the VR environment, using sensors and user interfaces. The sensors may include image sensors such as a wide-angle camera, a narrow-angle camera, a user-facing camera, and a depth sensor. Non-image sensors may also be used, including gyroscopes, magnetometers, accelerometers, GPS sensors, retina/pupil detectors, pressure sensors, biometric sensors, temperature sensors, humidity sensors, optical or radio-frequency sensors that track the user's location or movement (*e.g.*, user's fingers, arms, or body), and ambient light sensors. The sensors can be used to create and maintain virtual environments, integrate "real world" features into the virtual environment, properly orient virtual objects (including those that represent real objects, such as a mouse or pointing device) in the virtual environment, and account for the user's body position and motion.

The user interfaces may be integrated with or connected to the computing device and enable the user to interact with the VR environment. The user interfaces may include a touchscreen, a keyboard, a pointing device, a mouse or trackball device, a joystick or other game controller, a camera, a microphone, or an audio device with user controls. The user interfaces allow a user to interact with the virtual environment by performing an action, which causes a corresponding action in the VR environment (*e.g.,* raising an arm, walking, or speaking).

The tracking system may also include output devices that provide visual, audio, or tactile feedback to the user (*e.g.*, vibration motors or coils, piezoelectric devices, electrostatic devices, LEDs, strobes, and speakers). For example, output devices may provide feedback in the form of blinking and/or flashing lights or strobes, audible alarms or other sounds, songs or other audio files, increased or decreased resistance of a control on a user interface device, or vibration of a physical component, such as a head-mounted display, a pointing device, or another user interface device.

Fig. 1 illustrates the display, tracking, and VR-content systems as disparate entities in part to show the communications between them, though they may be integrated, *e.g.*, a smartphone mounted in a VR receiver, or operate separately in communication with other systems. These communications can be internal, wireless, or wired. Through these illustrated systems, a user can be immersed in a VR environment. While these illustrated systems are described in the VR context, they can be used, in whole or in part, to augment the physical world. This augmentation, called “augmented reality” or AR, includes audio, video, or images that overlay or are presented in combination with the real world or images of the real world. Examples include visual or audio overlays to computing spectacles (*e.g.,* some real world-VR world video games or information overlays to a real-time image on a mobile device) or an automobile’s windshield (*e.g.,* a heads-up display) to name just a few possibilities.

Using systems similar to those described in Fig. 1, VR (and AR) developers try to build immersive experiences with as few links to the external world as possible. Typically, a user is wearing a VR headset (goggles) and headphones, and has a small input controller in each hand. The controllers are used to interact with the virtual world, mostly through gestures that the developer must manually pre-define. These gestures may not always reflect what a user would intuitively use for the action. For example, a developer may define a gesture for moving virtual objects that requires the user to point the controller within a certain distance of an object and then move the controller to the left. Similarly, a developer may define a gesture that allows a user to perform an action like drawing a triangle by requiring a user to make controller movements and stops at certain points that form a triangle. When the user performs the gesture, within a tolerance, the VR system can tell that the user is drawing a triangle.

This method limits developers because each gesture has to be defined manually. Further, because developers typically lack the time and resources to consider a large number of ways a user might perform the gesture, conventionally defined gestures may be less expressive. The manual method also presents challenges to developers because gestures are difficult to precisely define to match the user's intent. For instance, in the triangle example, the developer must make assumptions about where the user will start, how precise the corners will be, how big the triangle will be, and so forth. This can frustrate developers, slow development time, and reduce the quality of the user's VR experience.

Description:

To address the problem of developing accurate and natural gestures for virtual reality (VR) environments, a machine-learning technique is provided that allows a VR system to automatically learn user gestures based on a ground-truth data set collected from real users. The technique includes two steps. First, ground-truth data is collected by observing multiple users intentionally performing a specified action in the virtual environment. For example, an action to move an object from one place to another is recorded through input from different sensors in the VR system (*e.g.*, the tracking system of Fig. 1, using position, orientation, controller actuations, or force/acceleration data). In this way, a set of data is generated for which the relationship between particular gestures and the user's intended action is known. Second, machine-learning techniques (*e.g.*, a recurrent neural network) are used to allow the VR system to learn to recognize gestures that actual users perform that are intended to represent the actions in the ground-truth data.

Fig. 2 illustrates the concept in a block diagram form. In the example shown in Fig. 2, a recurrent neural network (RNN) is the machine-learning technique that is used. Other techniques may also be used, such as a feedforward neural network that includes a feature selector or techniques that use a Hidden Markov Model.

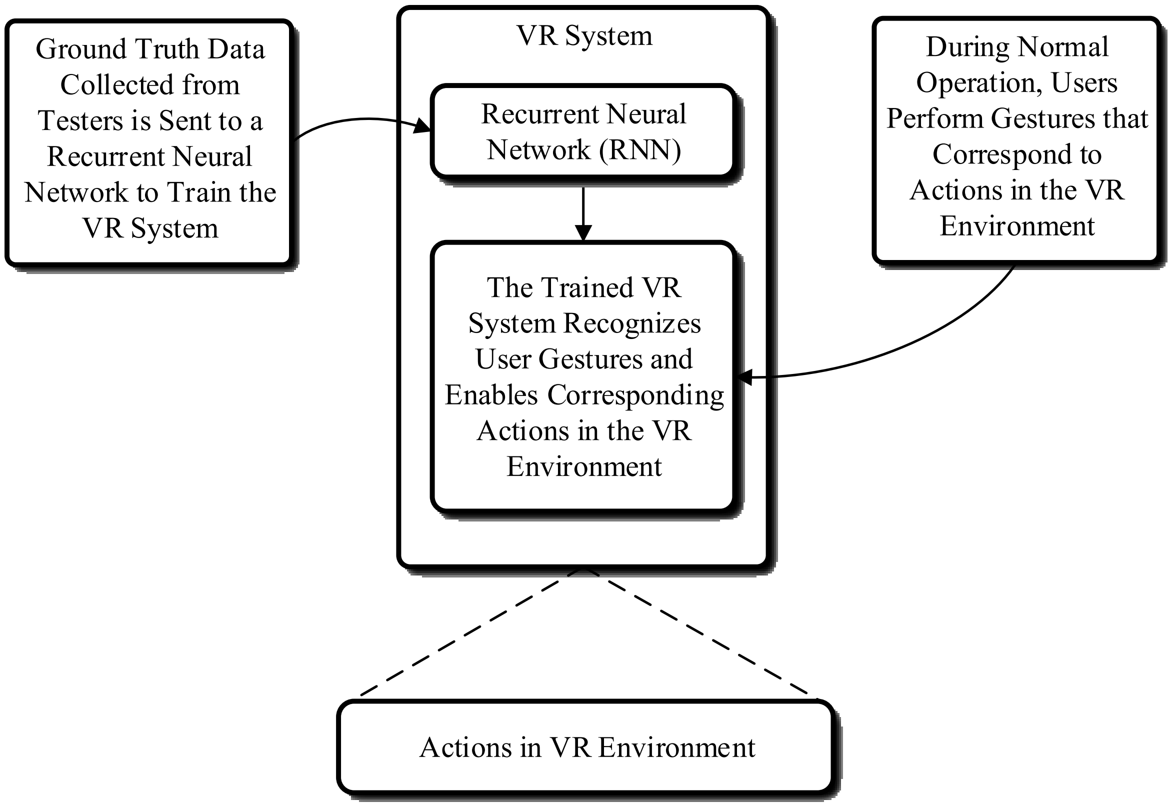

Ground Truth Data Collected from Testers is Sent to a Recurrent Neural Network to Train the VR System
VR System
Recurrent Neural Network (RNN)
The Trained VR System Recognizes User Gestures and Enables Corresponding Actions in the VR Environment
During Normal Operation, Users Perform Gestures that Correspond to Actions in the VR Environment
Actions in VR Environment

Fig. 2

First, a human tester uses the full VR system, including controllers, in a sample VR scene in which the gesture to be learned can be performed. The user is instructed to repeat a certain action, corresponding to the gesture for which data is being collected. The user is given only minimal guidelines regarding the gesture. For instance, the user may be told to move a ball from one bin to another. The system records the data related to the user’s movements. The recorded data may include a sequence of location and orientation coordinates from the input controller, controller buttons pressed, acceleration data, force data (*e.g.*, how hard the user squeezes the controllers). To provide data on how the users hold or use the controllers when no action is being performed, the users may also be asked to remain in the VR scene without performing any actions or to remain in the VR scene and interact with other users, again without performing actions. The process is repeated with a set of testers that perform each action multiple times. For example, a set of twenty testers may perform each requested action dozens or hundreds of times to collect data about the gestures used to perform each action.

Fig. 3 illustrates an example of how different users may employ different gestures to perform the same action. Continuing with the above example, in which users have been asked to move a ball from one bin to another, Figure 2 depicts two users performing the first part of the gesture, grasping the ball. In the figure, the users are holding VR controllers in each hand, and the ball is represented with a red dashed line. User A starts with arms spread wide apart and then brings them closer and lower. In contrast, the gesture of User B is subtler, with less change in grasp width and not as much downward motion.

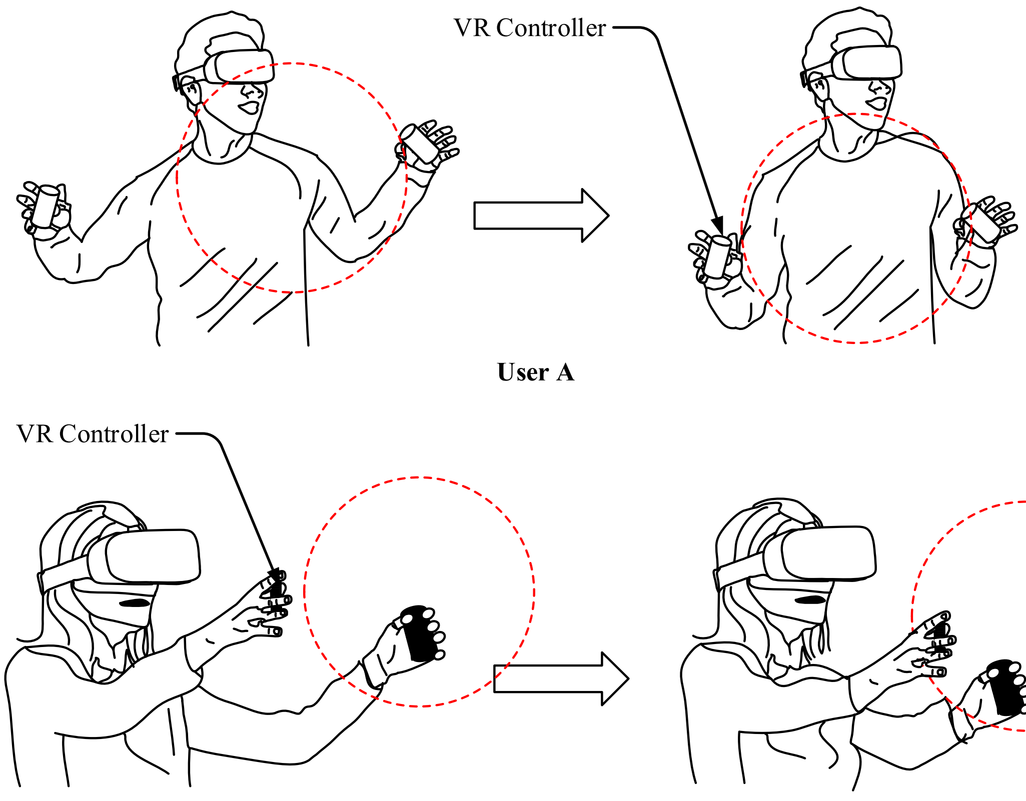


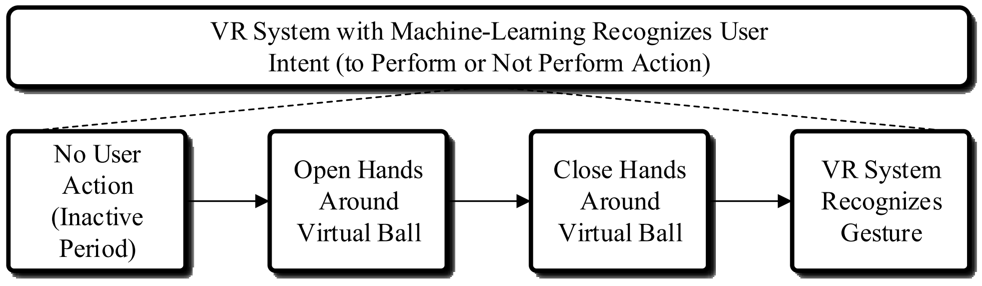
Fig. 3

The data collected for each gesture may be a sequence of numbers representing the location and orientation of the user's hands, the cursor, the pointer, and/or other control mechanism being used to interact with the VR scene. As noted, other information may also be collected, such as pressing of buttons, force data, and/or acceleration data. The collection process provides a time sequence of the measured data.

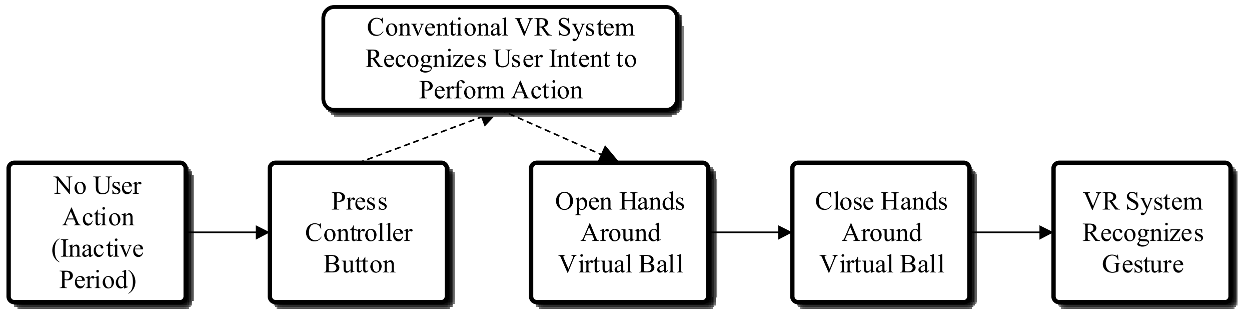
In the second step, a VR system learns to recognize gestures in an immersive three-dimensional virtual environment using the collected ground-truth data. The collected ground-truth data is used to train a machine-learning model, such as the recurrent neural network of Fig. 2. In this way, the VR system can receive a sequence of inputs from a user and feed it to the model trained with the ground-truth data to predict the gesture most likely matching the user's intent. The VR system can thereby learn to distinguish particular gestures, such as picking up a ball, from other gestures or from no particular gesture. For example, the ground-truth data provides the machine-learning model with known data about gestures that represents a user's intent to move a ball, which the model can use to predict that another user's gesture is also intended to move a ball.

As noted above, the model may also contain a class for "no gesture" actions, which represent a default background state. This improves the VR system's ability to determine when the user is simply holding the controller without any specific intent, which improves accuracy. Further, the user can rely on more natural gestures because the VR system can determine whether the user is intending an action without requiring the user to push a button or activate some other control. For example, in a conventional VR system, a user may have push a button to tell the system that the next movement is an action (*e.g.*, grasping a ball). With the machine-learning system, the user merely has to pick up the ball, and the system knows that a period of inactivity has ended and the user intends the system to respond to the gesture and pick up the ball.

Fig. 4 illustrates this concept. In view 4A, the VR system includes the machine-learning techniques described above. As shown, the VR system recognizes that the user is in an inactive period (*e.g.*, waiting for other users or observing the VR scene without acting). The user can simply grasp the virtual ball and the VR system will recognize the gesture because it can distinguish the user's "no gesture" state from the user's intent to perform an action. In contrast, view 4B shows a convention VR system in which the user must press a controller button before the VR system recognizes the user's intent to perform an action. The described VR system thus enables actions to be recognized and performed using fewer and more natural gestures.


VR System with Machine-Learning Recognizes User Intent (to Perform or Not Perform Action)
No User Action (Inactive Period)
Open Hands Around Virtual Ball
Close Hands Around Virtual Ball
VR System Recognizes Gesture

4A


Conventional VR System Recognizes User Intent to Perform Action
No User Action (Inactive Period)
Press Controller Button
Open Hands Around Virtual Ball
Close Hands Around Virtual Ball
VR System Recognizes Gesture

4B

Fig. 4

Optionally, the system may use the ground-truth data to break more complex gestures down into sequences of basic gestures, such as controller up, controller down, rotate controller, and so forth. The system can learn what different sequences mean (*e.g.*, grasp, open, move, throw) and then determine user intent by recognizing sequences of the basic gestures as full gestures.

A wide variety of gestures can be learned with the described machine-learning system, including basic gestures for VR environments, such as pointing at an object to pick it up, using a wrist-flick motion to drop the object, moving an object, pushing an object, or throwing an object. More-complex real-life actions are also possible, such as opening or closing a door gently, slamming a door, ringing a bell, or playing a musical instrument. In gaming environments, the system can learn gestures that trigger specific events. For instance, in a fantasy game, a protagonist may invoke various magical actions using different wand movements. In another game, a user may have to learn different gestures and apply them in different contexts to advance in the game. The described techniques may also be used to enable user identification and/or authentication by gestures, such as a secret gesture that represents the user's passcode.

Depending on the application, gestures can be pre-learned before the system is shipped to the user or the system can learn and/or refine gestures specifically for a particular user. For example, in a game application, the system can be ready to use right of the box by recording data and training the model offline before it is shipped to the user. In other applications, the system is able to learn new gestures after the user receives it, such as if a gesture is to be used as a form of authentication or for new/different types of interactions in the VR environment. The user-specific learning can happen locally on the device/controller to protect the user's privacy.