March 01, 2019

Robotic Game Playing Internet of Things Device

Melissa Daniels

George Bacon

Robotic Game Playing Internet of Things Device

Inventors: Melissa Daniels and George Bacon

Summary

Generally, the present disclosure is directed to a robotic device that can play certain games with a user, such as, for example, interactive hand-gesture-based games such as rock-paper-scissors, gesture matching, and/or gesture mirroring. In particular, in some implementations, the systems and methods of the present disclosure can include or otherwise leverage an Internet of Things (IoT) system or device to control a robotic hand based on inputs from a camera that are processed by a machine-learned model such as a gesture recognition model.

In one particular example, the robotic device can initiate a game of rock-paper-scissors based on detection, by processing imagery captured by a camera, of a user's hand proximate to the robotic device. The robotic device can then engage the user in a game of rock-paper-scissors, keeping track of the number of times the user wins and generating an indication when either the user or the robotic device has won a predetermined number of games (e.g., best two out of three).

Alternatively or additionally to rock-paper-scissors, the robotic device can be configured to play other games such as a gesture matching game in which, for example, the robotic hand is controlled to perform a series of different gestures. Thereafter, the human player can attempt to match the sequence of gestures using his own hand. The human player can be scored/evaluated on his ability to match the sequence of gestures. In some embodiments, the number of gestures included in the sequence can gradually increase (e.g., one additional gesture for each instance in which the human player is successful). This may be similar to a "Simon Says" game. In another example, the robotic device can be configured to perform a gesture mirroring game, in which the

robotic device attempts to recognize a gesture made by a human player and to control the robotic hand to mirror the gesture made by the human player.

In some implementations, the robotic device can use machine-learning techniques to improve the effectiveness with which the robotic device recognizes objects and gestures. For example, machine-learning techniques can be used to improve the recognition of a hand and hand gestures by the robotic device.

Thus, aspects of the present disclosure are directed to the application of robotics and sensors to create a robotic device that can detect objects, recognize gestures, and play a game with a user. As such, one aspect of the present disclosure is directed to a vision system that is trained to recognize object states (e.g., without the use of sensors other than camera). The vision system can combine camera(s) with machine-learned models to recognize object states. While the primary example given herein is the recognition of hand gestures performed by a human, the vision system can be generalized to recognition of many different object states such as eye contact, facial expression, industrial device operational status, and many others.

Another example aspect of the present disclosure is directed to a system which enables physical IoT objects to be controlled based on the recognized object state such as a recognized user gesture. For example, various different IoT objects can be controlled, manipulated, or otherwise operated differently in response to recognition of different object states. Thus, while the primary example given herein is a robotic hand that can be controlled in association with or in response to detected human gestures, the control system can be generalized to control of many different IoT devices in response to recognized object state. Examples of many different IoT devices are provided below.

One particular example of the aspects described herein includes the robotic device that plays a game of rock-paper-scissors by detecting a user's hand and providing game feedback versus a robotic hand that is controlled by the device. By using a robotic hand and interacting with a user in the same manner that a human player would interact with the user, the robotic device provides an interactive example of how machine-learning can be applied to robotics and/or IoT devices.

Example Figures

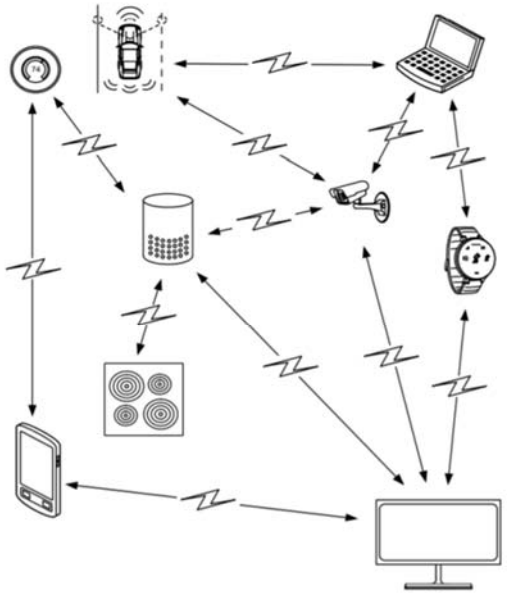


Fig. 1

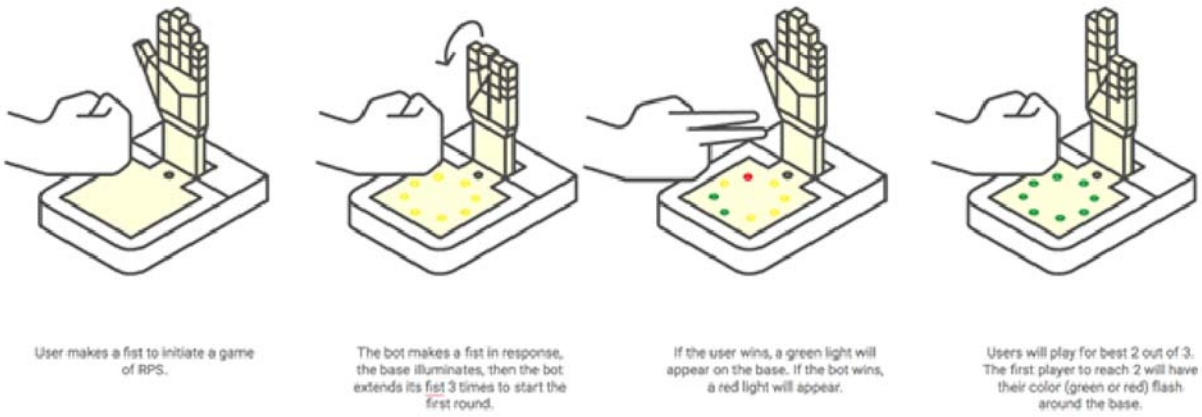


Fig. 5

Detailed Description

The present disclosure is directed to an IoT system that can control a robotic device based on and/or in response to recognition of an object state using cameras and machine learning. In particular, one example embodiments includes a robotic device that can play a game of rock-paper-scissors with a user. Further, the robotic device can operate as an Internet of Things (IoT) device within an IoT environment.

In some embodiments, the robotic device can include a base unit with an attached robotic hand. Further, the robotic device can include sensors to detect objects (e.g., a hand) and can perform functions including playing a game of rock-paper-scissors with a user. The robotic device can be built using an embedded operating system platform that can interact with Internet of Things devices.

Referring to Figures 3-4, example embodiments of robotic devices are described herein. The robotic device can include a computing device that uses an operating system that can operate

on embedded hardware of the robotic device. Further, the robotic device can include various sensors including a camera that can detect the area in front of the robotic device as well as gestures by a user (e.g., a user placing their clenched fist in front of the robotic device). In some embodiments, the camera, camera cable, and development board of the robotic device can come in the PICO i.MX7 StartKit.

In some embodiments, the robotic device can include a base to which a robotic hand (e.g., a robotic hand with a palm, wrist, four robotic fingers, and one robotic thumb) is attached. The robotic hand and base can be constructed from various materials including metal, plastic, or glass. In some embodiments, the robotic hand can include moveable, articulating, fingers that can move into three positions corresponding to the rock, paper, and scissors position respectively. In the rock position, the four fingers of the robotic hand are clenched in a fist that is overlapped by the thumb of the robotic hand. In the paper scissors, the four fingers and thumb of the robotic hand are in a fully extended position. In the scissors position, the two fingers closest to the thumb are fully extended, the two fingers furthest away from the thumb are clenched, and the thumb is folded over the palm. For example, the robotic device can include eight servos, five servos for each finger, one servo in the wrist, and two servos in the base of the robotic device.

In other embodiments, the robotic device can be controlled to make additional gestures such as Fighter jet (index + ring bent); Spiderman (middle + ring bent); English 3 (thumb + pinky bent); German 3 (ring + pinky bent); Loser (middle + ring + pinky bent); The Carpenter (index + middle bent); OK; Hang Loose; Peace Sign (Scissors); and/or Finger Wag (e.g., extends index finger if users make rude gesture).

In some embodiments, the robotic device can include one or more sensors including a camera. For example, the one or more sensors of the robotic device can include a camera that can

be used to detect when a user's hand is in front of the camera. Further, the robotic device can include indicators (e.g., LEDs on the base of the robotic device) to indicate the number of times that a user or the robotic device has won a round of rock-paper-scissors. For example, the robotic device can include colored LEDs that light up a green LED when the user wins a round of rock-paper-scissors and a red LED when the robotic device wins a round of rock-paper-scissors. When the user or the robotic device wins a predetermined number of rounds of rock-paper-scissors, the robotic device can cause the indicators to blink to indicate that the user or the robotic device has won the series of games.

In some implementations, the robotic device can include a machine-learned model that has been trained to perform various operations including object recognition and gesture recognition. For example, the robotic device can be trained to receive camera outputs and recognize when a user's hand is in front of the robotic device as well as the position of the user's hand (e.g., rock, paper, or scissors position). By way of further example, the robotic device can use a neural network trained with Tensorflow to recognize when a user makes a rock, paper, or scissors gesture.

Training the device can include using images of hands in the rock, paper, and scissors positions from the perspective of robotic device's camera. The images can then be used to train the Tensorflow network on an external computer. When the training is complete, Tensorflow can output a graph file that is uploaded to an Android Things board. This tells the robotic device how to classify what it sees through the camera, allowing it to recognize user gestures.

Further, the machine-learned model of the robotic device can determine different gestures or positions of the user's hand. For example, after the conclusion of a predetermined number of rounds of rock-paper-scissors or a predetermined time period without detecting a user's hand, the robotic device can enter a mode in which the robotic device is prepared to initiate a new game of

rock-paper-scissors based on detection of a user's clenched fist. Upon initiating a new game of rock-paper-scissors, the robotic device can determine when a user is in the process of performing a gesture (e.g., a rock, paper, or scissors gesture) and respond by performing a random rock, paper, or scissors gesture. Further, upon determining whether the user or the robotic device won a round of rock-paper-scissors, the robotic device can keep track of the number of times the robotic device or the user has won and notify the user when a certain number of rounds have been won by one player.

In some embodiments, the robotic device can be run on the Android Things operating system. For example, in some embodiments, the device can run on an i.MX7D Pico Development board, running Android Things. Android Things is an operating system that can run on embedded hardware, like a single board computer. It can run on a Raspberry Pi 3 or the i.MX7D board. In some embodiments, the code for the robotic device can be Java code written for Android Things in Android Studio, similar to developing an Android application for a smartphone. Android Things also supports Kotlin. In some embodiments, the servos of the robotic device can be controlled using a PWM hat that was designed for the Raspberry Pi. The i.MX7D board has the same GPIO pinout as the Raspberry Pi, so peripherals and hats can be used for it interchangeably.

In some embodiments, the body of the robotic device is made out of bead-blasted acrylic, and the fingers are 3D printed in ABS plastic. The string running through the fingers can be 100 pound fishing line. In some embodiments, the robotic hand can be constructed from folded nylon sheet material to create a robust, but easily recreatable formfactor. The flexibility of the nylon will make for simple construction (no mechanical joints) while retaining the ability to make complex hand gestures.

In some embodiments, the hand may be able to articulate each finger as well as extend forward and pivot along its wrist. The base will mark the area where users should make their hand gestures. The base can also illuminate with LED's, indicating the status and scoring of the different games the robotic device can play.

In some embodiments, the machine-learned models of the device can be re-trained before each public display event (e.g., conference) to improve accuracy. For example, images can be gathered at a specific location, classified, and then used to retrain the model(s). Training usually takes between 2 and 4 hours. Thus, in order to retrain the model, new pictures at an event must be taken to add into the machine-learned model. The model takes a few hours to train, so this should happen the day before the event during the setup day.

In some embodiments, the robotic device can be further connected (e.g., using a network interface to a wired and/or wireless network) to a cloud computing system. One example cloud computing system is Google Cloud IoT Core. The robotic device can send telemetric and status data to the cloud for remote monitoring and analytics. For example, the device and cloud computing system can operate together to track data such as number of wins, number of losses, various measures of hardware performance, and/or other operational data. The cloud system can track the information for a specific robotic device and/or over multiple different robotic devices deployed as a fleet.

Figure 5 shows an example user flow for the robotic device to play rock-paper-scissors. Figure 6 shows how the robotic device demonstrates an error state. Figure 7 shows an example user flow for the robotic device to play a matching game.

Figure 1 depicts a block diagram of an example IoT environment according to example implementations of the present disclosure. As illustrated in Figure 1, in some implementations, the

IoT environment includes a plurality of different devices, each of which can be referred to as an IoT device. An example IoT device can be an intelligent, environmentally-sensing, and/or network-connected device configured to communicate with a central server or cloud service, a control device, and/or one or more additional IoT devices to perform any number of operations (e.g., in response to received commands). IoT devices can, in some instances, also be referred to as or include "smart" devices and/or "connected" devices.

Each IoT device can be a stand-alone physical device or can, in some instances, be an embedded device that is embedded within a larger device or system. Each IoT device can include electronics, software, sensors, actuators, and/or other components, including various components that sense, measure, control, and/or otherwise interact with the physical world. An IoT device can further include various components (e.g., a network interface, antennas, receivers, and/or the like) that enable the IoT device to send and/or receive data or other information from one or more other IoT devices and/or to a central system.

In particular, the various IoT devices can be communicatively connected to one another via one or more communications networks. The networks can be wide area networks, local area networks, personal area networks, piconets, cellular networks, other forms of networks, and/or combinations thereof. The networks can be wired and/or wireless. The networks can be private and/or public. As examples, two or more of the IoT devices can communicate with one another using a Wi-Fi network (e.g., a home network), Bluetooth, Bluetooth Low Energy, Zigbee, Radio-Frequency Identification (RFID), machine to machine connections, inductive communications, optical communications, infrared communications, other communications techniques or protocols, and/or combinations thereof. For example, an IoT device might communicatively connect with a

first nearby device using Bluetooth while also communicatively connecting with a second nearby device using Wi-Fi.

In some implementations, each IoT device can have a unique identifier. For example, the identifier for each IoT device can include and/or be based on an Internet Protocol address associated with such IoT device, a manufacturer associated with such IoT device, a location at which such IoT device is positioned, a model number of such IoT device, a functionality of such IoT device, and/or other device characteristics. In some implementations, a given IoT device can locate and/or communicate with another IoT device based on its unique identifier. In some implementations, the identifier assigned to an IoT device can be modified by a user and/or owner of such IoT device.

In particular, in some implementations, a user can assign one or more identifiers to the IoT devices within a device topology representation. The device topology representation can describe and/or organize a group of IoT devices (e.g., based on location with one or more structures such as one or more homes, offices, vehicles, and/or the like). The identifiers can be chosen by the user and associated with the respective IoT devices within the device topology representation. The identifier(s) can include but are not limited to names, nicknames, and/or aliases selected for the IoT devices by the user. In this manner, the identifiers can be names or aliases of the respective IoT devices that the user is likely to use when identifying the IoT devices for requested control or command operations (e.g., “turn on the kitchen lights”).

An IoT device can be mobile or can be stationary. In some implementations, an IoT device can be capable of autonomous or semi-autonomous motion.

In some implementations, an IoT device can be controlled or perform operations in response to communications received by the IoT device over a network. For example, an IoT

device can be controlled by a control device that is communicatively connected to the IoT device. The control device can communicate directly with the IoT device or can communicate indirectly with the IoT device (e.g., over or using a mesh network). The control device can itself be an IoT device or the control device can be a device that is not considered part of the IoT environment. For example, the control device can be a server device that operates as part of cloud computing system. The commands can be in response to or generated based on a user input or can be generated without user input.

Thus, in one example, an IoT device can receive communications from a control device and the IoT device can perform operations in response to receipt of such communications. The performed operations can be internal operations (e.g., changing an internal setting or behavior) or external operations (e.g., interacting with the physical world in some way). The IoT device and the control device can be co-located or can be remotely located from each other.

As an example, the control device can be or include a user device such as a smartphone, tablet, computing device that is able to be worn, laptop, gaming console or device, virtual or augmented reality headset, and/or the like. As another example, the control device can be a server computing device. As another example, the control device can itself be an IoT device. For example, the control device can be a so-called “smart speaker” or other home control or automation device.

In some implementations, a user can interact with a control device (e.g., which can be an IoT device) to input data into or otherwise control the IoT environment. For example, the control device can include and execute a software application and/or other software programs that provide a user interface that enables entry of user input. The software applications can be executed entirely at the control device or can be web applications where some portion of the program or functionality is executed remotely (e.g., by a server connected over the Internet) and, in some implementations,

the client-side logic runs in a web browser. Thus, in some implementations, a web server capable of sending, receiving, processing, and storing web pages or other information may be utilized.

In some implementations, a cloud service may be used to provision or administer the IoT devices. For example, a cloud computing system can enable or perform managed and/or integrated services that allow users to easily and securely connect, manage, and ingest IoT data from globally dispersed IoT devices at a large scale, process and analyze/visualize that data in real time, and/or implement operational changes and take actions as needed. In particular, in some implementations, the cloud computing system can employ a publication subscription model and can aggregate dispersed device data into a single global system that integrates seamlessly with data analytics services. An IoT data stream can be used for advanced analytics, visualizations, machine learning, and more to help users improve operational efficiency, anticipate problems, and/or build rich models that better describe and optimize the user's home or business. The cloud system can enable any number of dispersed IoT device to connect through protocol endpoints that use automatic load balancing and horizontal scaling to ensure smooth data ingestion under any condition.

In some implementations, the cloud system can include or implement a device manager. For example, the device manager can allow individual IoT devices to be configured and managed securely in a fine- or coarse-grained way. Management can be done through a console or programmatically. The device manager can establish the identity of a device and can provide the mechanism for authenticating a device when connecting. The device manager can also maintain a logical configuration of each device and can be used to remotely control the device from the cloud.

In some implementations, an IoT device can include an artificial intelligence-based assistant or software agent. A user can interact with the artificial intelligence-based assistant via a control device, directly through the IoT device, or any other method of interaction. The artificial

intelligence-based assistant can perform tasks or services based on user input and/or contextual awareness (e.g., location awareness), including acting as a control device to control other IoT devices. In some implementations, an IoT device (e.g., an artificial intelligence-based assistant on such device) can access information from a variety of online sources (such as weather conditions, traffic congestion, news, stock prices, user schedules, retail prices, etc.).

The artificial intelligence-based assistant or software agent can be stored and implemented by a single device (e.g., a single IoT device) or can be spread across multiple devices and implemented by some (e.g., dynamically changing) combination of such multiple devices.

In some implementations, an IoT device can include (e.g., as part of an artificial intelligence-based assistant) one or more machine-learned models that assist in understanding user commands, determining context, and/or other actions. Example machine-learned models include artificial neural networks such as feed-forward neural networks, recurrent neural networks, convolutional neural networks, autoencoders, generative adversarial networks, and/or other forms, structures, or arrangements of neural networks. Additional example machine-learned models include regression models, decision tree-based models (e.g., random forests), Bayesian models, clustering models, linear models, non-linear models, and/or other forms, structures, or arrangements of machine-learned models. Machine-learned models can be trained using supervised learning techniques or unsupervised learning techniques. Machine-learned models can be stored and implemented on the IoT device or can be stored and implemented in the cloud and the IoT device can leverage the models by communicating with cloud devices. Feedback or other forms of observed outcomes can be used to re-train models to improve their performance. Models can be personalized to one or more users or environments by re-training on data specific to such users or environments.

In some implementations, the artificial intelligence-based assistant can perform concierge-type tasks such as, for example, making dinner reservations, purchasing event tickets, making travel arrangements, and/or the like. In some implementations, the artificial intelligence-based assistant can provide information based on voice input or commands (e.g., by accessing information from online sources). In some implementations, the artificial intelligence-based assistant can automatically perform management or data-handling tasks based on online information and events, including, in some instances, without user initiation or interaction.

In some implementations, a control device (e.g., which may be an IoT device) can include components such as a mouse, a keyboard, buttons, knobs, a touch-sensitive component (e.g., touch-sensitive display screen or touch pad), and/or the like to receive input from the user via physical interaction.

In some implementations, the control device can include one or more microphones to capture audio signals and the device can process the audio signals to comprehend and respond to audio commands (e.g., voice commands) provided by a user or by some other device. Thus, in some implementations, the IoT devices can be controlled based on voice commands from a user. For instance, a vocalization from a user can be received by a control device. The vocalization can be a command spoken by a user proximate to the control device. The control device can control itself and/or one or more of the IoT devices based on the vocalization.

In some implementations, one or more vocalization(s) may be used as an interface between a user and an artificial intelligence-based assistant. For example, a user may vocalize a command which the artificial intelligence-based assistant may identify, process, and/or execute or cause to be executed. The vocalized command may be directed at the artificial intelligence-based assistant.

As one example, the vocalization may indicate a user desire to interact with or control another IoT device (e.g., lowering a thermostat setting, locking a door, turning off a light, increasing volume, etc.). The artificial intelligence-based assistant may communicate the command to the desired IoT device which can respond by executing or otherwise effectuating the user command. As another example, the vocalization can include a user commanding the artificial intelligence based assistant to perform a task (e.g., input an event into a calendar, retrieve information, set a reminder, make a list, define a word, read the first result of an internet search, etc.).

In some implementations, speech recognition or processing (e.g., natural language processing) can be performed on the vocalization to comprehend the command provided by the vocalization. For instance, data indicative of the vocalization can be provided to one or more language models (e.g., machine-learned models) to determine a transcription of or otherwise process the vocalization.

In some implementations, processing the vocalization or other user input can include determining one or more IoT devices to control and/or one or more actions to be performed by the selected IoT devices. For instance, a semantic interpretation of the vocalization (e.g., a transcript of the vocalization) can be determined using one or more semantic interpretation techniques (e.g., natural language processing techniques). The semantic interpretation can provide a representation of the conceptual meaning of the vocalization, thereby also providing an interpretation of the intent of the user.

In some implementations, the interpretation of the vocalization can be determined based at least in part on the device topology representation. For instance, the device topology representation can be accessed to determine the one or more selected IoT devices and/or actions to be performed.

As one example, the device topology representation can be accessed and compared against a transcription of the vocalization to determine a match between one or more terms included in the transcription and one or more terms associated with the IoT device topology representation (e.g., “kitchen lights”). In some implementation, the identity of the speaker can be ascertained and used to process the vocalization (e.g., such as to process commands that include possessive modifiers: “brew a cup of my favorite roast of coffee”).

In some implementations, the control device (e.g., which may be an IoT device) can include a vision system that includes one or more image sensors (e.g., visible-spectrum cameras, infrared cameras, LIDAR systems, and/or the like) that capture imagery. The device can process the imagery to comprehend and respond to image-based commands or other input such as, for example, gesture commands provided by the user. In some implementations, the vision system may incorporate or perform facial movement identification (e.g. lip reading) capabilities while, in other implementations, the vision system may additionally or alternatively incorporate hand shape (e.g. hand gestures, sign language, etc.) identification capabilities. Facial movement and/or hand shape identification capabilities may allow a user to give commands a control device in addition or alternatively to voice control.

In some implementations, in response to the image data of the facial or hand gesture, the control device can determine one or more IoT devices to control and/or one or more actions to be performed (e.g., by the selected IoT devices). Interpretation of image data that depicts lip reading and/or sign language may be achieved through any method of image data analysis. The interpretation can provide a representation of the conceptual meaning of the image data. In this manner, the interpretation of the image data can provide an interpretation of the intent of the user in performing the gesture(s).

In some implementations, the interpretation can be determined based at least in part on the device topology representation. For instance, the device topology representation can be accessed to determine the one or more selected IoT devices and/or the action to be performed. For example, the device topology representation can be accessed and compared against the image data to determine a match between one or more aspects of the image data and one or more aspects associated with the IoT device topology representation (e.g., the user may be pointing to a specific IoT device when providing a voice command or a gesture command).

In further implementations, gaze recognition can be performed on the captured imagery to identify an object or device that is the subject of a gaze of the user. A user command (e.g., voice or gesture) can be interpreted in light of (e.g., as applied to) the object or device that is the subject of the gaze of the user.

In some implementations, the vision system may be used as an interface between a user and an artificial intelligence-based assistant. The captured image data may be interpreted and/or recognized by the artificial intelligence-based assistant.

In some implementations, the selected IoT devices and/or the actions to be performed can be determined based at least in part on contextual data (e.g., location of user, day of the week, user data history, historical usage or command patterns, user wardrobe, etc.) For instance, in response to receiving a command from a user, a location of the user, a time of day, one or more past commands, and/or other contextual information can be determined. The location can be determined using various suitable location determination techniques. The location determination technique can, for example, be determined based at least in part on the control device to which the user provides the vocalization.

As one example, if the control device is an IoT device that is specified in the device topology representation, the user location can be mapped to the structure and/or room to which the control device is assigned in the device topology representation. As another example, if the control device is a user device not specified in the device topology representation, the user location can be determined using one or more location determination techniques, such as techniques using wireless access points or short range beacon devices associated with one or more IoT devices, and/or other suitable location determination techniques. In some implementations, the user location can be mapped to one or more structures and/or rooms specified in the device topology representation. In some implementations, the control device and/or other IoT devices can also process audio signals and/or imagery to comprehend and respond to contextual information. As examples, triangulation and/or beamforming techniques can be used to determine the location of the user based on receipt of the voice command at multiple different audio sensors. In some implementations, multiple possible user commands or requests can be disambiguated based on the contextual information.

Further to the descriptions above, a user may be provided with controls allowing the user to make an election as to both if and when systems, devices, or features described herein may enable collection of user information (e.g., information about a user's preferences, a user's activities, or a user's current location), and if the user is sent content or communications from a server. 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. For example, a user's identity may be treated so that no personally identifiable information can be determined for the user, or a user's geographic location may be generalized where location information is obtained (such as to a city, ZIP code, or state level), so that a particular location of a user cannot be determined. Thus, the user may have

control over what information is collected about the user, how that information is used, and what information is provided to the user.

Figure 2 provides a block diagram of an example software stack that can be included on an IoT device. The software stack shown in Figure 2 is provided as one example only. Various different IoT devices can have any number of different software and/or hardware configurations which may be of greater or lesser complexity to that shown in Figure 2.

In some implementations, an IoT device can include and execute one or more computer applications (also known as software applications) or other computing programs. The IoT device can execute the application(s) to perform various functions, including collection of data, communication of data, and/or responding to or fulfilling received commands. In some implementations, the software applications can be native applications.

In some implementations, the software application(s) on an IoT device can be downloaded and installed by or at the direction of the user. In other implementations, the software application(s) can be default applications that come pre-programmed onto the IoT device. In some implementations, the software application(s) can be periodically updated (e.g., via download of update packages). The software application(s) can be closed source applications or can be open source applications. The software applications can be stand-alone applications or can be part of an operating system of the IoT device. The software applications can be embodied in computer-readable code or instructions that are stored in memory and then accessed and executed or followed by one or more processors of the IoT device.

In some implementations, the software application(s) on an IoT device can be user-facing applications such as a launcher or a browser. In other implementations, the IoT device does not

include any user-facing applications but, for example, is instead designed to boot directly into applications developed specifically for the device.

More particularly, in some implementations, an IoT device can include or otherwise be implemented upon or in conjunction with an IoT platform that includes a number of elements. The IoT platform can include an operating system. The operating system can, for example, have been optimized for use in the IoT environments (tuned for faster boot times and/or lower memory footprint). The operating system and other platform elements may be able to receive secure and managed updates from the platform operator. The IoT platform can include hardware that is accessible and easy to integrate.

The IoT platform can also enable application developers to build applications using a rich framework provided by an operating system software development kit (SDK) and platform services, including, for example, the same user interface toolkit, multimedia support, and connectivity application programming interfaces (APIs) used by developers of mobile applications for larger devices such as smartphones. Applications developed for the IoT device can integrate with various services using one or more client libraries. For example, the applications can use the libraries to interact with services such as application deployment and monitoring services, machine learning training and inference services, and/or cloud storage services. The applications can use the APIs and/or support libraries to better integrate with hardware, including, for example, custom hardware. This can include support for peripheral interfaces and device management. The device can also include a number of native libraries, including, for example, C/C++ libraries, runtime libraries, core libraries, and/or the like. Updates to one or more of these components can be deployed over the air and/or automatically when updates are available.

In some implementations, an IoT device (e.g., the software applications executed thereby) can utilize APIs for communicating between a multitude of different software applications, operating systems, databases, libraries, enterprises, graphic interfaces, or any other component of the IoT environment disclosed herein. For instance, a first software application executed on a first IoT device can invoke a second software application via an API call to launch the second software application on a second IoT device.

In some implementations, the applications can run on a single or variety of operating system platforms including but not limited to OS X, WINDOWS, UNIX, IOS, ANDROID, SYMBIAN, LINUX, or embedded operating systems such as VxWorks.

The IoT device can include one or more processors and a memory. The one or more processors can be any suitable processing device (e.g., a processor core, a microprocessor, an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), System on a Chip (SoC), a controller, a microcontroller, etc.) and can be one processor or a plurality of processors that are operatively connected. The memory can include one or more non-transitory computer-readable storage mediums, such as RAM, ROM, EEPROM, EPROM, flash memory devices, magnetic disks, etc., and combinations thereof. The memory can store data and instructions which are executed by the processor to cause the IoT device to perform operations. The IoT devices can, in some instances, include various other hardware components as well, including, for example, a communications interface to enable communication over any number of networks or protocols, sensors, and/or other components.

In some implementations, the IoT device can include or be constructed using one or more System on Module (SoM) architectures. Each SoM can be a fully integrated component that can be dropped directly into a final design. Modules can be manufactured separately and combined to

form the device. In some implementations, the device software can include a hardware abstraction layer and kernel which may be packaged as a board support package for the modules. In other implementations, different, non-modular architectures can be used.

Example IoT devices include or can be associated with an air conditioning or HVAC system, lighting device, a television or other home theater or entertainment system, security system, automatic door or window locking system, thermostat device, home energy manager, home automation system, audio speaker, camera device, treadmill, weight scale, smart bed, irrigation system, garage door opener, appliance (e.g., refrigerator, dishwasher, hot water heater, furnace, stove, fireplace, etc.), baby monitor, fire alarm, smoke alarm, medical devices, livestock tracking devices, cameras, beacon devices, a phone (e.g., smartphone), a computerized watch (e.g., a smart watch), a fitness tracker, computerized eyewear, computerized headwear (e.g., a head mounted display such as a virtual reality of augmented reality display), other types of computing devices that are able to be worn, a tablet, a personal digital assistant (PDA), a laptop computer, a desktop computer, a gaming system, console, or controller, a media player, a remote control, utility meter, an electronic book reader, a navigation system, a vehicle (e.g., car, boat, or plane/drone) or embedded vehicle system, an environmental, food, or pathogen monitor, search and rescue devices, a traffic control device (e.g., traffic light), traffic monitor, climate (e.g., temperature, humidity, brightness, etc.) sensor, agricultural machinery and/or sensors, factory controller, GPS receivers, printers, motor (e.g., electric motor), and/or other suitable device or system.

The technology discussed herein makes reference to servers, databases, software applications, and other computer-based systems, as well as actions taken and information sent to and from such systems. One of ordinary skill in the art will recognize that the inherent flexibility of computer-based systems allows for a great variety of possible configurations, combinations,

and divisions of tasks and functionality between and among components. For instance, server processes discussed herein may be implemented using a single server or multiple servers working in combination. Databases and applications may be implemented on a single system or distributed across multiple systems. Distributed components may operate sequentially or in parallel.

Figures

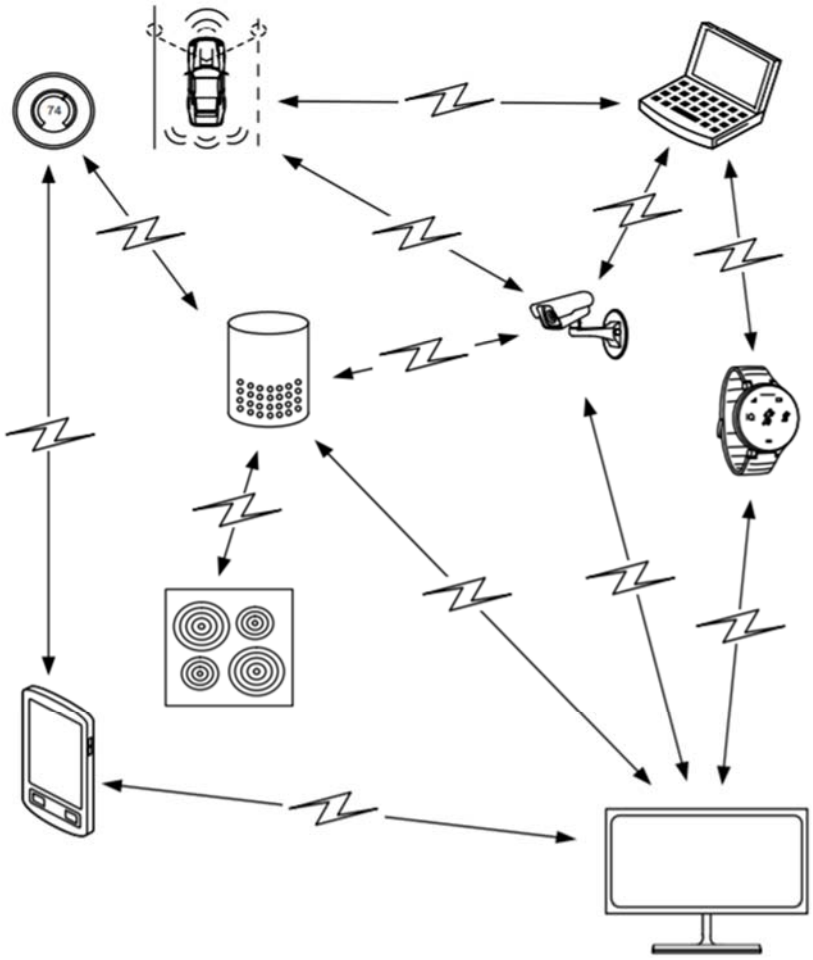


Fig. 1

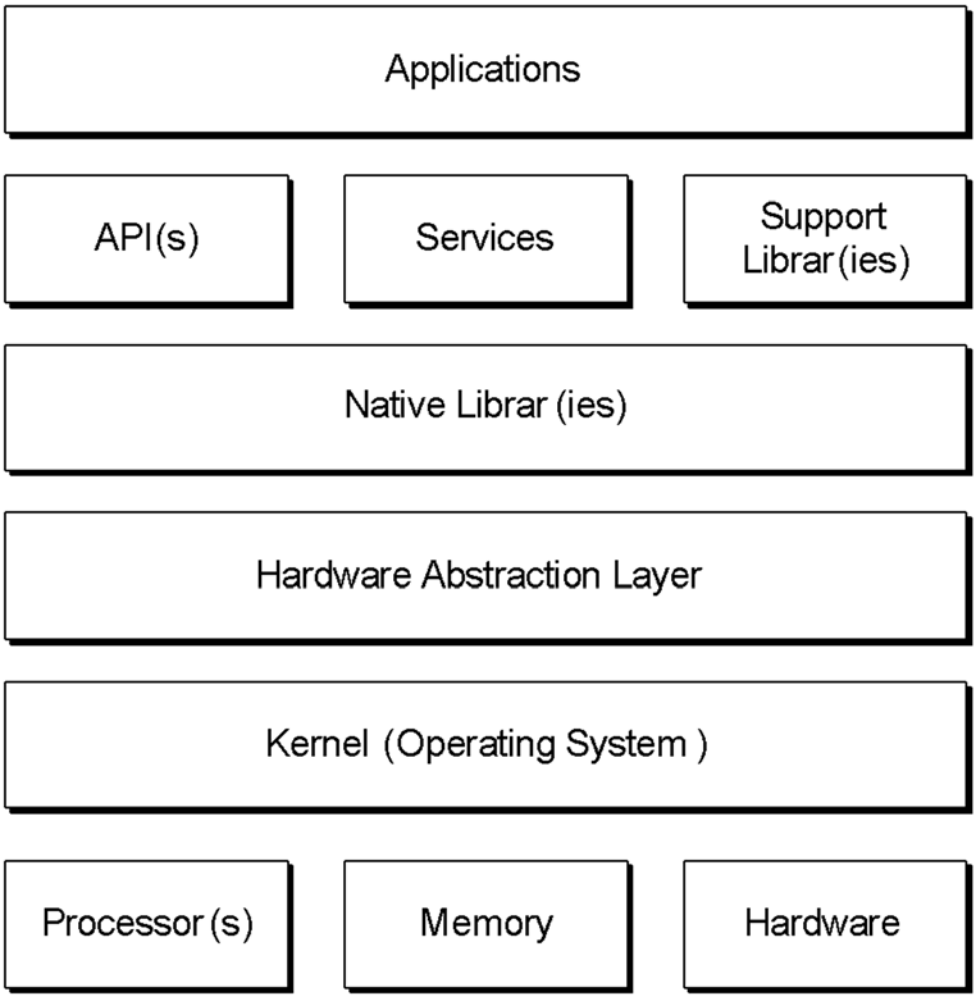

Applications
API(s)
Services
Support Librar(ies)
Native Librar(ies)
Hardware Abstraction Layer
Kernel (Operating System)
Processor(s)
Memory
Hardware

Fig. 2

Fig. 3

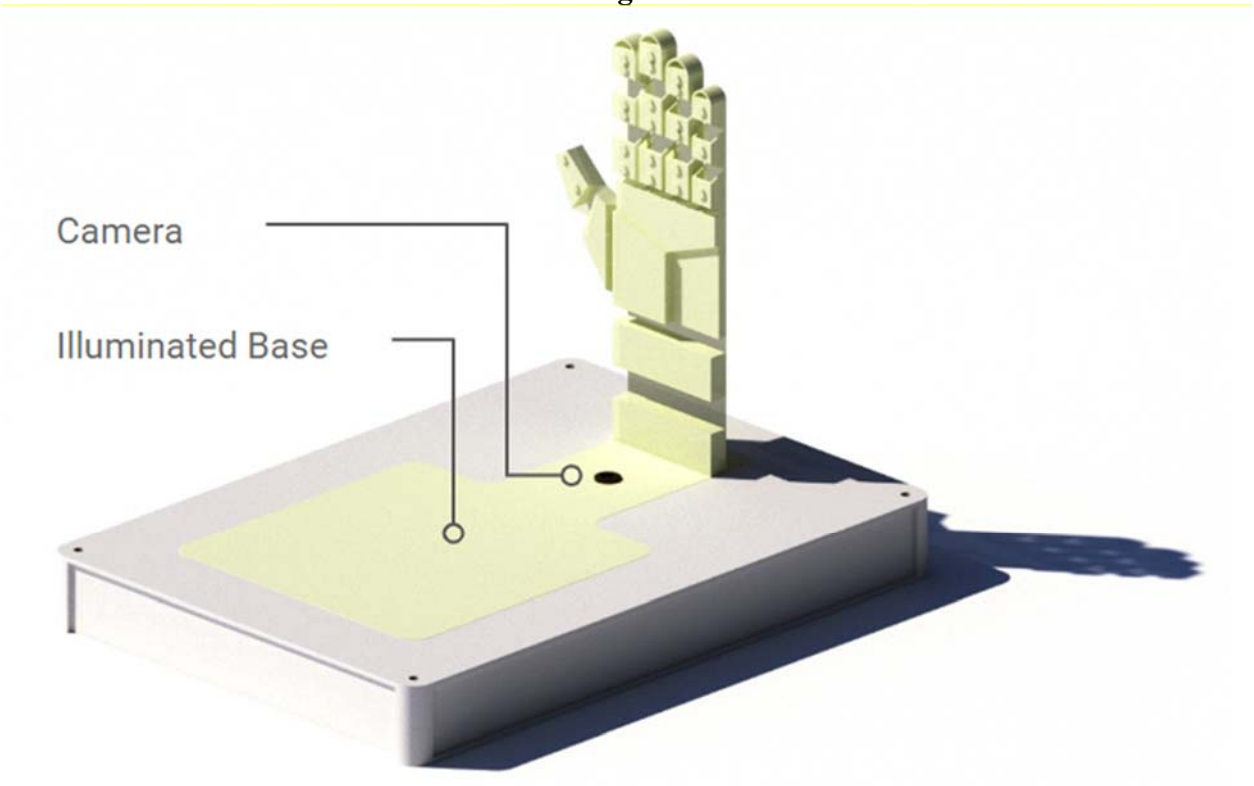


Fig. 4

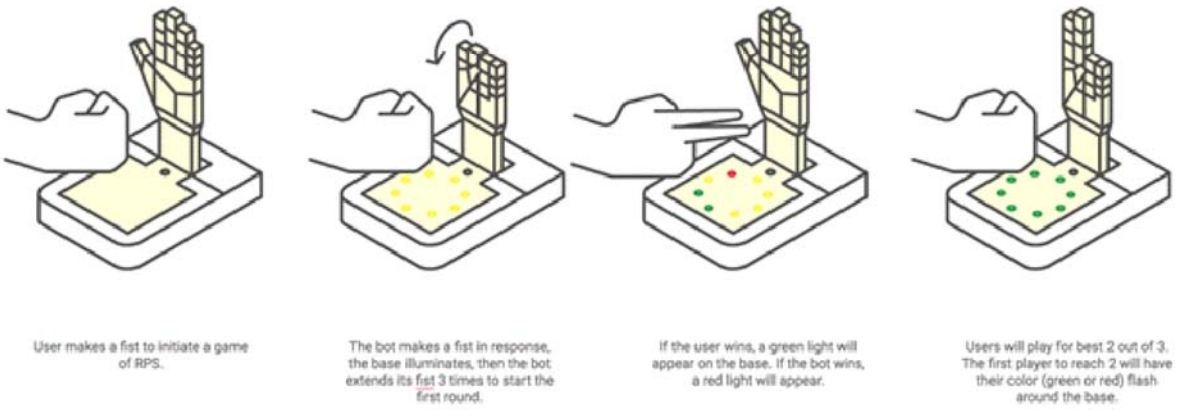

User makes a fist to initiate a game of RPS.
The bot makes a fist in response, the base illuminates, then the bot extends its fist 3 times to start the first round.
If the user wins, a green light will appear on the base. If the bot wins, a red light will appear.
Users will play for best 2 out of 3. The first player to reach 2 will have their color (green or red) flash around the base.

Fig. 5

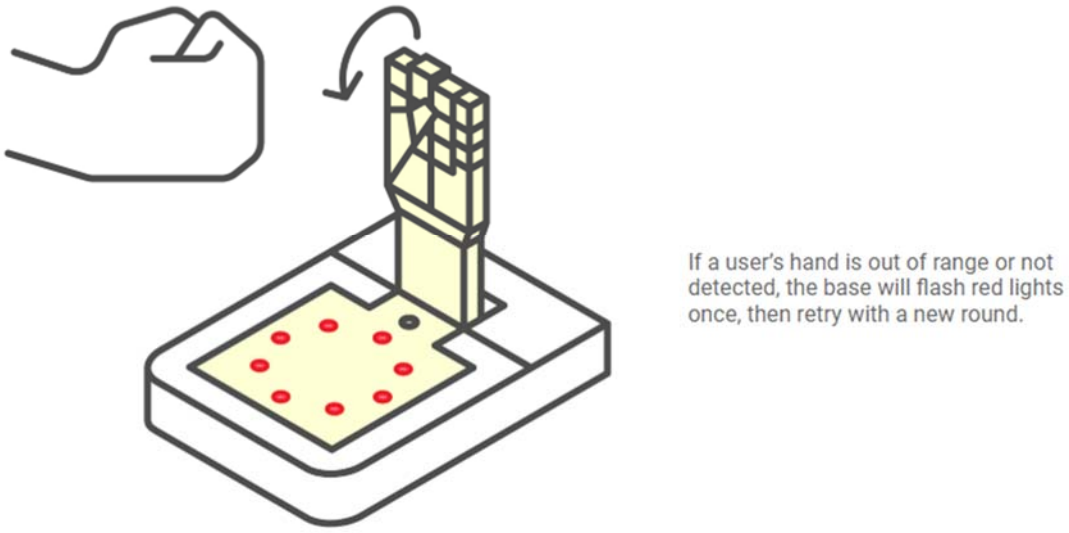
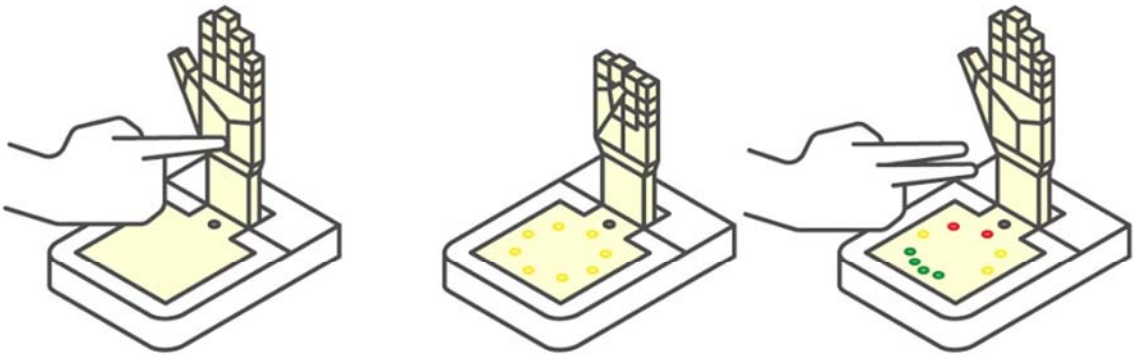

If a user's hand is out of range or not detected, the base will flash red lights once, then retry with a new round.

Fig. 6



User points at the bot to initiate a matching game.

The bot will make a series of random hand gestures. The series of gestures will become longer as the game progresses.

Users must memorize these sequences and repeat them. If a user completes 5 sequences they will win, however if they make 3 errors they will lose.

Fig. 7