Technical Disclosure Commons

Defensive Publications Series

March 04, 2019

Interactive Device that Performs Output Based On Human Movement and/or Human Emotion Detected via Machine Learning

Melissa Daniels

Nhat Vu

Daniele Midi

Ed Coyne

Jiyoung Ko

See next page for additional authors

Follow this and additional works at: https://www.tdcommons.org/dpubs_series

Recommended Citation

Daniels, Melissa; Vu, Nhat; Midi, Daniele; Coyne, Ed; Ko, Jiyoung; Schilit, Bill N.; and Bacon, George, "Interactive Device that Performs Output Based On Human Movement and/or Human Emotion Detected via Machine Learning", Technical Disclosure Commons, (March 04, 2019)
https://www.tdcommons.org/dpubs_series/2000

This work is licensed under a Creative Commons Attribution 4.0 License.

This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.

Inventor(s)
Melissa Daniels, Nhat Vu, Daniele Midi, Ed Coyne, Jiyoung Ko, Bill N. Schilit, and George Bacon

This article is available at Technical Disclosure Commons: https://www.tdcommons.org/dpubs_series/2000

Interactive Device that Performs Output Based On Human Movement and/or Human Emotion Detected via Machine Learning

Inventors: Melissa Daniels, Nhat Vu, Daniele Midi, Ed Coyne, Jiyoung Ko,
Bill N. Schilit, and George Bacon

Summary

Generally, the present disclosure is directed to a device capable of providing an output that is specific to and/or based on nearby human movement and/or human emotion detected by one or more machine-learned models.

As one example, the device can be part of an artistic installation such as, for example, a mechanical and/or digital representation of one or more flowers provided at a museum, conference, or the like. The flower(s) can change colors, open or close (e.g., "bloom" or not), change orientations, generate different scents, play different audio (e.g., music and/or environmental sounds), and/or otherwise change their output or operating characteristic(s) based on detected human movement and/or emotion.

Although the use of the disclosed technology in an art installation is given as the primary example in this document, the disclosed technology has broad applicability to any human-machine interactions in which human motion and/or emotion is used as an input to control the behavior/output of a robotic device, which may generally be referred to as affective computing. For example, the technology described herein can be applied to render a household robotic device which may include or interoperate with a "smart speaker" or a "smart assistant" to be more visually responsive to human motion and/or emotion.

More particularly, in one example, the device can include a first machine-learned model that can perform person tracking and/or facial tracking to detect movement of a nearby human (e.g., based on imagery captured by a camera of the device). The device can alternatively or additionally include a second machine learned model that can perform emotion recognition (e.g.,

based on imagery of the human's face captured by the camera). The device can change its orientation so that it is continuously facing a human who is walking past the art installation. Alternatively or additionally, the device can also change its color to match the detected human emotion (e.g., red for angry, blue for sad, etc.). In another example embodiment, using artistic style transfer, the device can generate artistic portraits that are based on the detected human emotions. For example, style transfer techniques (e.g., neural style transfer techniques) can be used to render an image of a user in a "happy" artistic style if it is detected that the user is smiling.

Thus, one example aspect of the present disclosure is directed to a system where the machine may identify a user's facial features and track movements. For example, the machine may continuously capture (or be operated to capture) a video and control at least a part of a robotic machine (e.g., a flower) based off of movements captured in the video. For example, the robotic machine can move a portion of the device (e.g., flower stem) to turn the device to face a user. In addition, in some implementations, machine learning techniques may be used to utilize the continuously captured video to predict an emotion present on the user's face and actuate at least a part of the robotic machine (e.g., petals of the flower) based off of the predicted emotion (e.g., open and close the petals, color of the petals, etc.).

Furthermore, aspects of the present disclosure are directed to mobile robotic devices that are capable of reproducing captured imagery (e.g., images captured by a camera that is on-board the robotic device) through the use of machine learning. For example, the robotic device can capture (or be operated to capture) an image (e.g., a self-portrait photograph which is also known as a "selfie") and can then reproduce the image (e.g., a stylized version thereof) by predicting the emotion present on a user's face. Thus, the final reproduced image can be a stylized version that is tailored to the predicted emotion.

Thus, aspects of the present disclosure are directed to the application of robotics and motors to create a visual representation of human motion and/or emotion using on-device processing. One particular example of the aspects described herein includes a flower that can react to the emotions expressed by a user. The flower may change colors and open and close petals depending on the emotion expressed by the user. By using colors and petal movements the flower may interact with a user. The robotic device provides an interactive example of how machine-learning can be applied to robotics. Examples embodiments of the aspects described herein can be built using an embedded operating system platform such as the Android Things operating system, thereby demonstrating the capabilities of Android Things and leveraging its flexibility as an operating system.

Example Figures

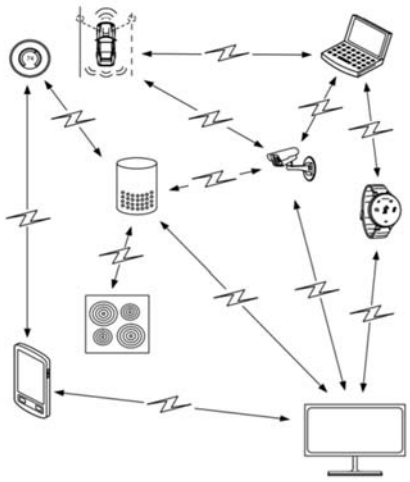

Fig. 1

Fig. 5

Detailed Description

Figs. 3-6 show example embodiments of the robotic flowers described herein, which may be referred to as “continuum flowers” (e.g., when motion tracking is performed) and/or “emotion flower” or “expression flowers” (e.g., when emotion detection is performed).

As described above, the present disclosure is directed to configuring an output based off of human facial movement and/or emotion through the use of device locomotion and/or color. 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 actuate a machine based on a user’s facial cues.

Fig. 7 shows a block diagram of one example system configuration for the robotic devices described herein. In some implementations, the robotic device can include and run on a Raspberry Pi 3B, running Android Things. In some implementations, the robotic device can also

use a camera module connected via USB as well as APA102-LED strips. For example, the camera can be embedded in the head of the flower or other portion of the device that will be tracking the human's motion.

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. A user (e.g., developer) can write Java code for Android Things in Android Studio, similar to the development process for developing an Android application for a smartphone. Android Things also supports Kotlin.

In some implementations, the robotic device can be powered by splitting the Raspberry Pi 3 Power Supply Charger. One side powers servos that control motion, and the other powers the LED cables that control lighting effects.

In some implementations, the robotic device can have components (e.g., flower petals) that are built from laser-cut and/or vinyl cut light diffuser film and wood. In some implementations, there are 3D printed parts (e.g., flower head). The fiber optic cables and LED strips can be off-the-shelf parts available to hobbyists. The robotic device can be stationary or can be portable.

In some implementations, the robotic device can include one or more machine-learned models that have been trained to perform various operations including face detection, facial gesture recognition, emotion recognition, motion tracking, and/or combinations thereof. For example, the robotic device can be trained to receive camera outputs and recognize when a user's face is in front of the robotic device as well as the position of the user's facial features (e.g., smile, frown, etc.). In some embodiments, the machine-learned models can be neural networks trained with and/or for implementation by an on-device machine learning library (e.g., Tensorflow or Tensorflow Lite).

In some embodiments, once the robotic device has received imagery of a user (e.g., video and/or image) from the USB camera, the robotic device can include and implement an on-device machine-learned model (e.g., “facesdk”, “feelnet”, face emotion NN estimator, or the like) using an on-device machine learning library (e.g., TensorFlow and/or TensorFlow Lite) to generate inferences on the robotic device. Thus, the robotic device can run machine-learned models “on-device”. As one example, the robotic device can run certain machine-learned models (e.g., neural networks) on-device in order to detect a face in the captured image. For example, a heuristic algorithm may be used to select a face deemed most likely to be interacting with the machine. For instance, in a crowded scenario, a face most centered and/or closest to the camera may be selected and tracked.

In some implementations, training the robotic device may occur offline or online. Training the robotic device can include using images of a face making expressions indicating emotions from the perspective of robotic device’s camera. The images can then be used to train the model 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 expressions and/or other characteristics of human motion and/or emotion.

In some implementations, a face detection model can provide an output that indicates whether a face is detected in the image and, if so, may attempt to predict an emotion expressed by the face (e.g., happy, sad, surprised, angry). For example, an emotion estimation neural network can parse the scene and may predict an output emotion.

Furthermore, subsequent processing may map the predicted emotion to servo motor actuation (e.g., flower petal movement) and/or color changes. For example, a Servo Controller

may receive I2C commands from the Raspberry Pi to control actuation (e.g., flower petal angles). The I2C commands may be sent to the servos. Likewise, an LED Controller may receive the SPI commands from the Raspberry Pi to adjust individual LEDs. In one example, there may be an aggregation of a cluster of components (e.g., 10 flowers). In such instances, for example, there may be an equal number (e.g., 10) of APA102-LED strips that may be controlled individually and/or in parallel.

In some implementations, a separate LED Controller may be utilized in order to control all of the aggregations (e.g., 10 flowers). For example, a LED Controller may include a demuxing chip to demux the SPI clock signal from the Raspberry Pi. A clock signal may be passed to each aggregation. For example, the clock signal may write new color instructions, and then move onto the next aggregation. Thus, an arbitrary number of LED strips may be controlled using a single SPI port. Fig. 8 provides a schematic diagram of one example wiring that can be used to build the robotic devices described herein.

The software can be an Android Things application that includes components to process the camera system, infer the face emotions/motion, and create commands for output actuations. The software can be open sourced or can be proprietary.

For example, face detection can be performed using Google's facesdk to detect and track the faces in the scene. A heuristic algorithm can be used to select the face deemed most likely to be interacting with the flowers. Typically in a crowded scenario, the face most centered an closest to the camera is selected and tracked.

Emotion detection can be performed using Google's face emotion NN estimator to infer the expressed emotion. The emotions can be post-processed and mapped to a number (e.g., 4) of

primary emotions (e.g., happy, sad, angry, and surprise). The threshold to trigger an action can be set relatively high to create a more playful UX.

The servos can be controlled using a PCA9685 chip. A custom java driver can be used for the chip. Each flower in the cluster can have a servo at its base that is connected to this chip. Each flower's petals can be actuated by moving the corresponding servo at the base. The Pi can send an I2C signal to the chip and can specify the angle to set for each servo. As examples, surprised can be set to fully open, happy to medium open, sad to medium closed, and angry to fully closed.

The LEDs can be controlled by first selecting which flower to send the new command. For example, a 74HC4067 demux chip can be used to controls which port is active via four binary signals. The Pi sends the binary signals to select the flower, then sends the SPI signal to control the LEDs for that flower.

In some embodiments, the output actuations (e.g., color and petal movements) can be expressed in terms of sequences. Each detected emotion can result in a different sequence. The sequence can be defined in terms of color change, petal open/close degree, timing of the change propagation throughout the cluster, and/or duration of the sequence.

According to another aspect of the present disclosure, in some implementations, machine learning techniques such as "style transfer" techniques can be used to stylize the captured imagery into a particular style (e.g., a pencil sketch style portrait) prior to reproduction. For example, a machine-learned model can be run to produce brightness, intensity, etc. values for each pixel or for groups of pixels which cause the drawing to exhibit a certain style (e.g., a user-selected style or a style identified based on a detected human emotion).

In some implementations, paintings may be input to the system so that the "style transfer" technique may be used to stylize the captured imagery into a pastiche of the inputted paintings. For example, certain paintings may be categorized as exhibiting characteristics of a certain emotion. For instance, a predominantly red painting may be categorized as angry. In another example, the painting emotions may be dictated by art historians' opinions.

In some implementations, more than one image may be taken and the image with the highest fidelity (e.g., the most clear emotion displayed) may be utilized as the final image that will be stylized.

In some implementations, the stylized portrait may be displayed and/or printed along with (optionally) the original work of art it was based off of along with some text describing in some form the process used to predict the emotion present in the image and/or background of the original work of art.

In some implementations, machine learning techniques are done offline, therefore no data is stored or transmitted.

One example interaction may be as follows: The user may approach the frame and take a series of three images. Next, the portraits may be rendered in succession based off of the emotions predicted in the images and corresponding art (e.g., Edvard Munch's "The Scream" would render for a reaction that was predicted to be sad). The image with the highest fidelity may be selected to be stylized and then displayed and/or printed along with the original work of art the image was stylized after and some text describing the background of the stylizing process. For example, Fig. 9 depicts an example image of a user that was generated through performance of style transfer.

Referring now to Fig. 1, Fig. 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

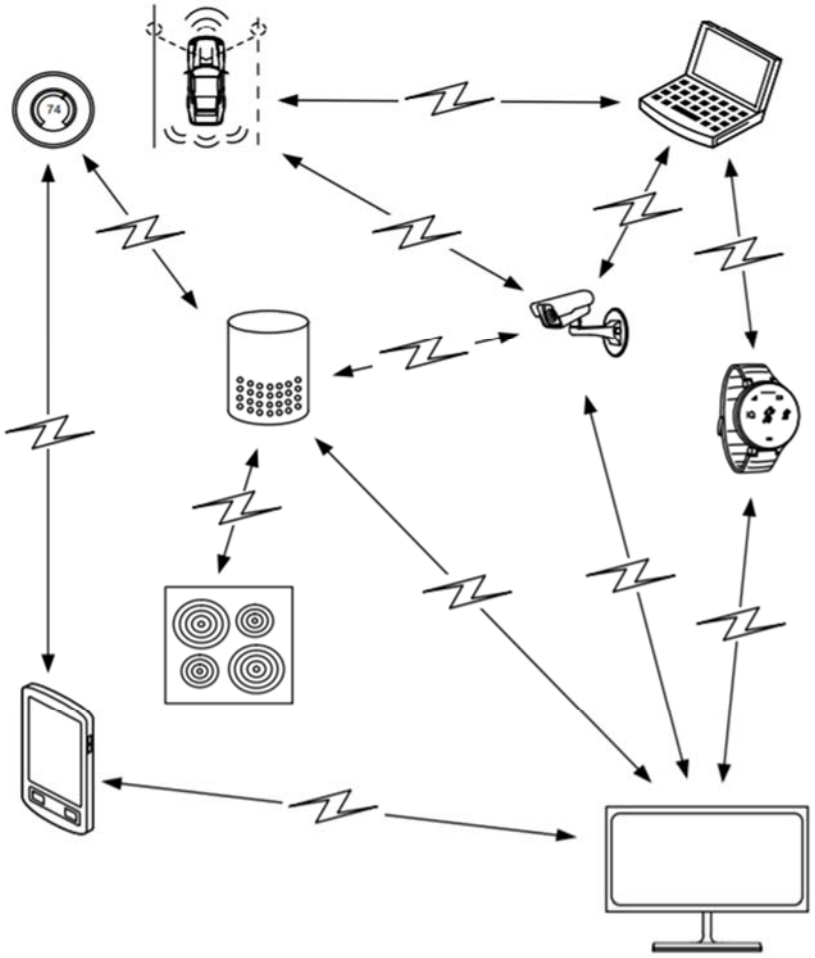

Fig. 1

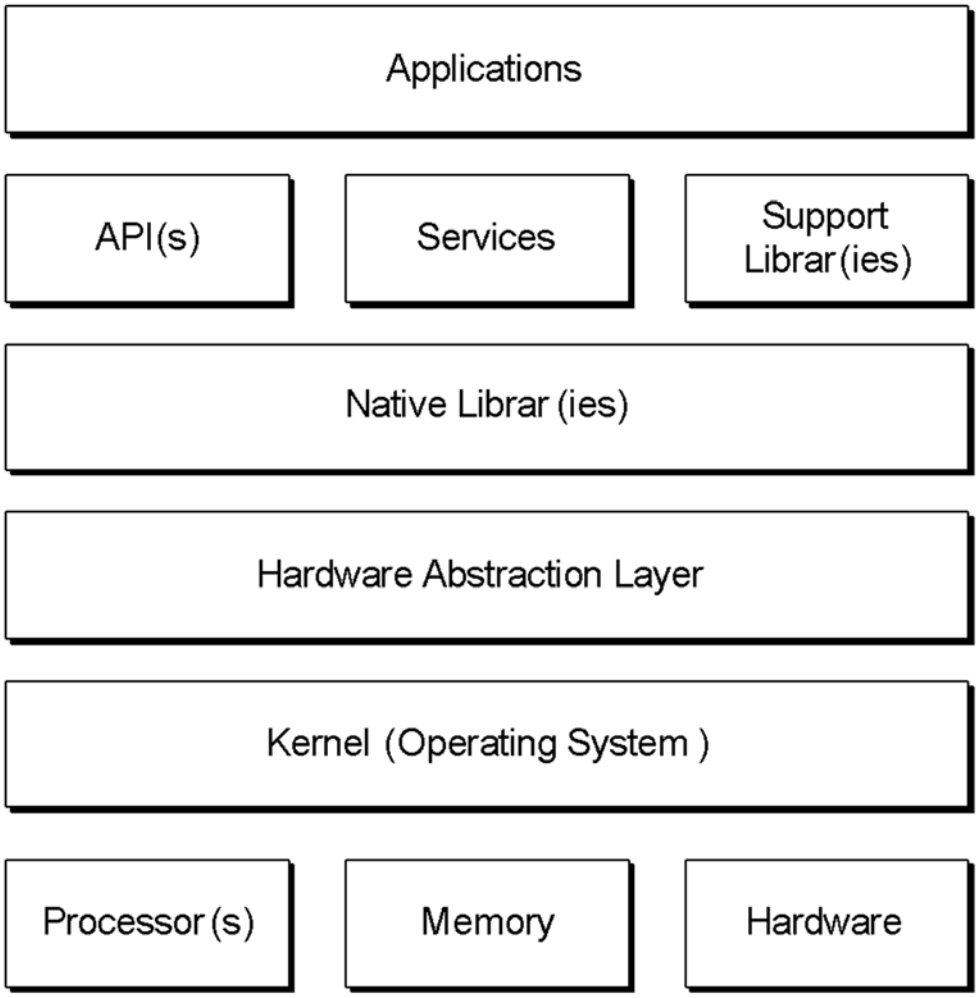
Applications
API(s)
Services
Support Librar(ies)
Native Librar(ies)
Hardware Abstraction Layer
Kernel (Operating System)
Processor(s)
Memory
Hardware

Fig. 2

Fig. 3

Fig. 4

Fig. 5

Fig. 6

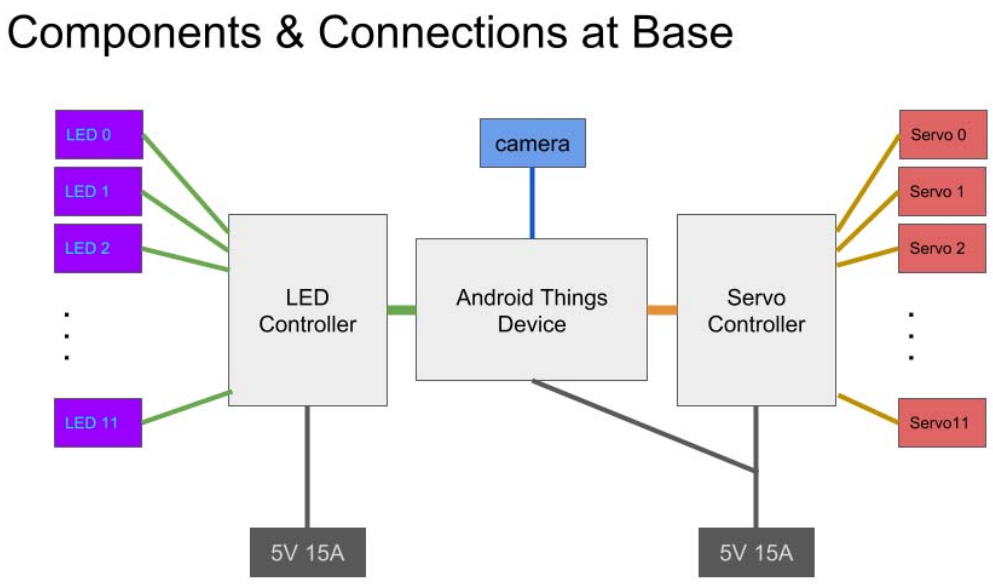
Components & Connections at Base
LED 0
LED 1
LED 2
LED 11
camera
LED Controller
Android Things Device
Servo Controller
Servo 0
Servo 1
Servo 2
Servo11
5V 15A
5V 15A

Fig. 7

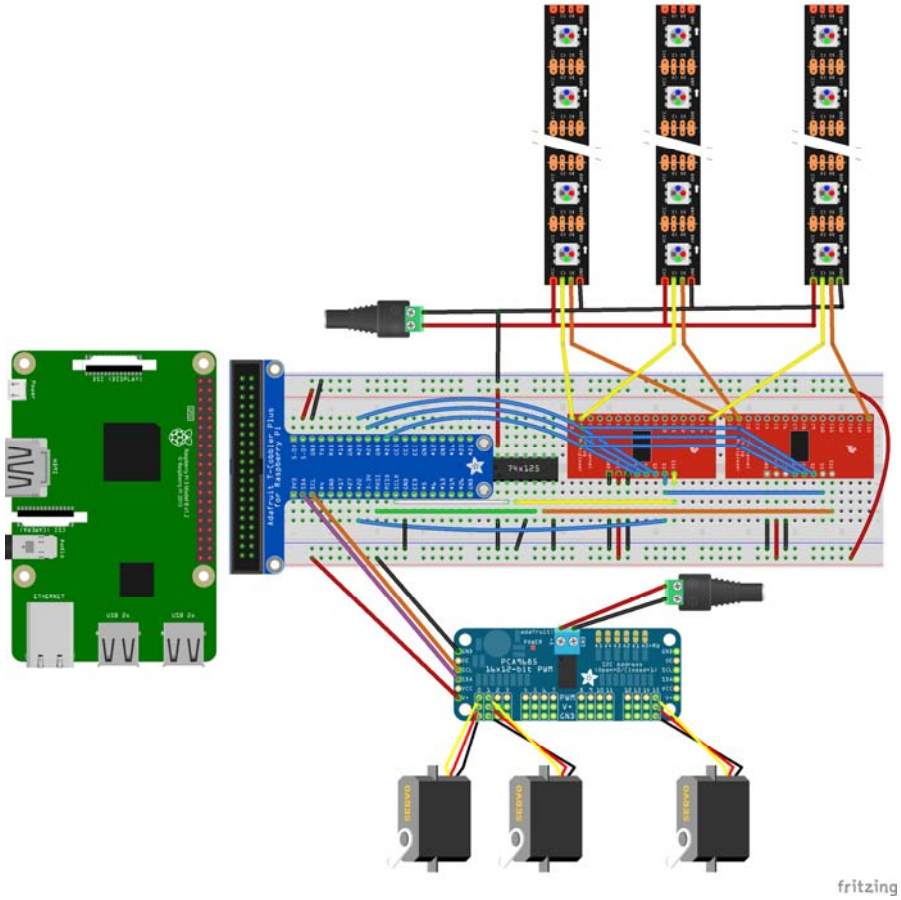
fritzing

Fig. 8

Fig. 9