May 29, 2019

Hybrid On-Device Cloud Scheme for Re-Identification of Persons Based on Shared Embedding Gallery

Nhat Vu

Daniele Midi

Tianchun Yang

Bill N. Schilit

Hybrid On-Device Cloud Scheme for Re-Identification of Persons Based on Shared Embedding Gallery

Inventors: Nhat Vu, Daniele Midi, Tianchun Yang, and Bill N. Schilit

Summary

Generally, the present disclosure is directed to a system of facial and/or person recognition via machine learning and Internet of Things (IoT). In particular, in some implementations, the systems and methods of the present disclosure can include or otherwise leverage a machine learning and IoT system or device to track and/or identify a person based on video images taken by one or more device(s). For example, a hybrid on-device and cloud scheme can enable locally-derived embeddings from multiple camera devices to be sent to a shared cloud space which can cluster the embeddings to generate a person model for a given person. Later, a camera device participating in the scheme can again detect a face and can match an embedding generated for the face against the shared gallery of person models to (potentially) re-identify the previously observed person.

In particular, one aspect of the present disclosure is directed to a hybrid on-device and cloud approach that combines face tracking and recognition with person tracking and recognition, to re-identify people across different cameras, even when their face is not visible in some of the angles. For example, the hybrid architecture can offload as much of the machine learning as possible to be executed on-device, while leaving the heaviest inferences to be performed by a cloud system. The entire pipeline can be coordinated in a data driven fashion.

More particularly, example systems and methods may use a combination of on-device processing, cloud processing, and/or external device processing. For example, a machine may continuously capture (or be operated to capture) a video and detect at least one face that is present in the video. In some implementations, the detected face may be assigned to a person in order to

combine face tracking and recognition with person tracking and recognition. In some implementations, the first machine that captures the video may interact with a second machine that may be continuously capturing (or operated to capture) a video to re-identify people across different machines (e.g., by associating the facial information with a person in a shared embedding gallery). Alternatively or additionally, the first machine may later detect the same face at a second, later time and may be able to re-identify the same individual, even through their face was not visible in a number of interim image frames. In such fashion, face tracking/recognition and person tracking/recognition can be effectively combined to associate a face-based identity to a body model, and then re-identify the same individual even when their face is not visible.

Thus, aspects of the present disclosure are directed to the application of machine learning to create a method of human identification using on-device processing and/or cloud processing. One example of the aspects described herein includes a smart camera with face tracking and people recognition capabilities. The smart camera may communicate with other smart cameras so that a person walking into a different room with a smart camera with their back turned may continue to be recognized based off of data provided by the first smart camera. The machine provides an interactive example of how machine learning and IoT technologies can be applied to robotics in security and other applications. Examples 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

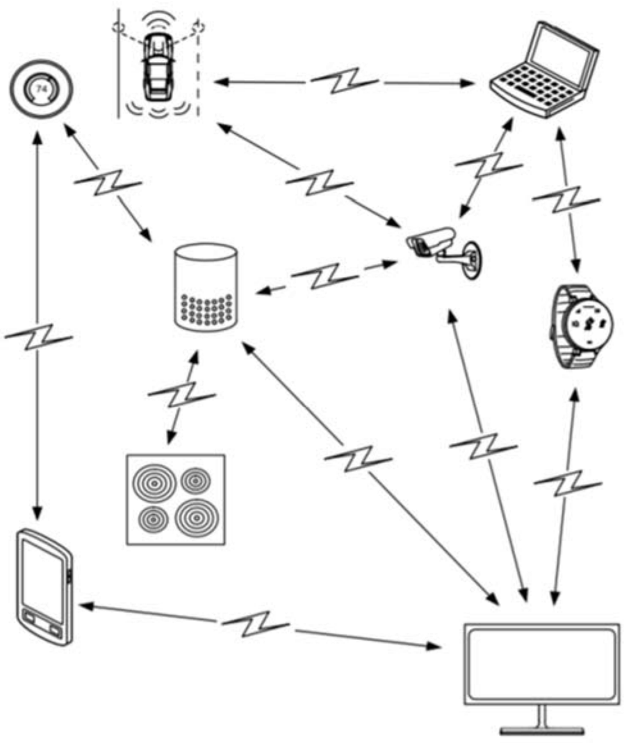

Fig. 1

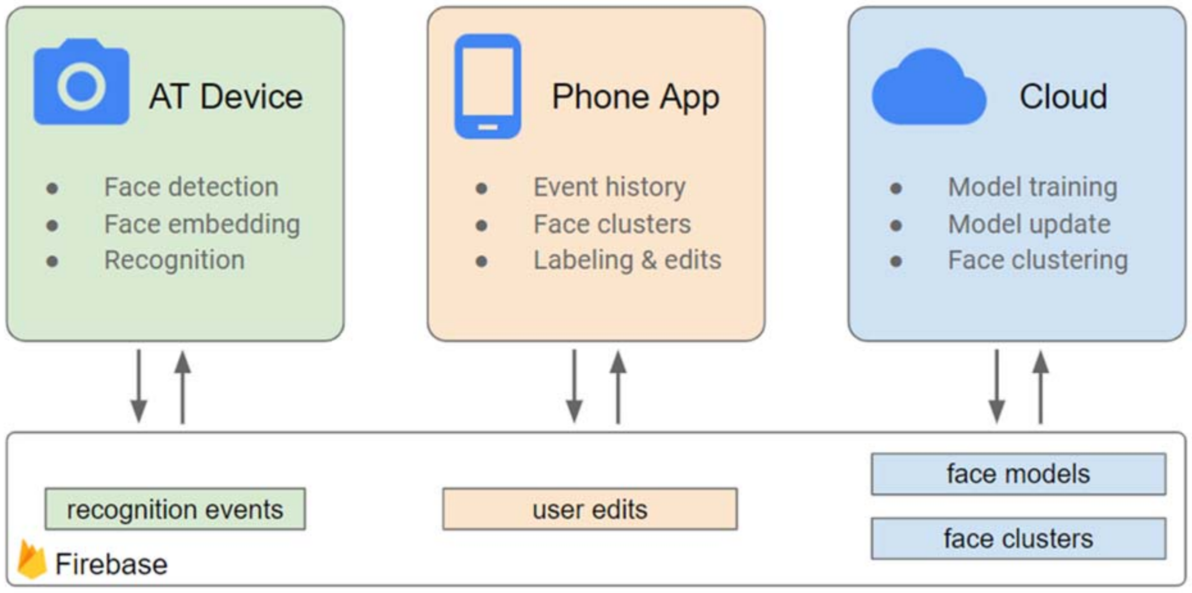

Fig. 3

Detailed Description

As described above, the present disclosure is directed to a system of facial and/or person recognition via machine learning and Internet of Things (IoT). In some implementations, the system can be performed by one or more IoT devices within an IoT environment. In particular, in some implementations, the systems and methods of the present disclosure can include or otherwise leverage machine learning in conjunction with the IoT to track and/or identify a person based on video images taken by one or more than one device(s).

Fig. 3 shows a block diagram of one example system configuration for a hybrid on-device cloud face and person recognition infrastructure. The proposed system can include an AT Device that can perform face detection, face embedding, and recognition. The AT Device can generate recognition events that can be stored in a shared remote database. A block diagram of an example AT Device is shown Fig. 4.

In some embodiments, the AT Device can include various sensors such as a camera that can capture images on which the AT Device can perform face detection, embedding, and recognition. For example, the AT Device can be configured to receive camera outputs and recognize when a person that is depicted in the imagery.

In some implementations, the AT Device can include one or more machine-learned models that have been trained to perform various operations including face detection and face embedding generation. For example, a face detection model can provide an output that indicates whether a face is detected in the image and, if so, where the face is located (e.g., as a bounding box via MultiBox inference model). A face embedding model may generate a face embedding for a detected face in a 128-D feature space.

A face recognition model can recognize a face by comparing an embedding generated for a face to other embeddings previously generated for other faces. As one example, the recognition model can identify one or more previous embeddings that are “closest” to the current face using a L2 distance formula. As an example, as shown in Fig. 5, for example, on-device machine learning may detect faces in the video and/or image taken by the AT Device (or a camera connected to the AT Device. A respective embedding can be generated for each detected face. The AT Device may then categorize the faces as recognized or unrecognized. In some embodiments, the faces that can be recognized are those that are able to be clustered with other face embeddings with some threshold of similarity to indicate that the face may be the same. For example, referring to Fig. 5, one face detected in the image was able to be recognized as “alex”, while the other face was unable to be recognized.

In some embodiments, the machine-learned models can be received from the cloud system and implemented on the AT Device in an “on-device” fashion. In particular, the AT Device can implement an on-device machine learning library (e.g., TensorFlow or TensorFlow Lite) to generate inferences on the AT Device.

Referring again to Fig. 4, the proposed system can also include a companion application (e.g., deployed on a smartphone as illustrated) that can perform event history, face clusters, and labeling and edits. The companion application can generate user edits that can be stored in the shared database. For example, a smart phone may include an application which allows a user to access event history and facial clusters. Further, a user may cluster and/or edit (e.g., images, faces and/or people) from within the application. For instance, the application may be able to interact with Firebase.

Figure 7 shows example actions that can be taken in the companion application. For example, a user may review and correct the categorization done by the “on-device” machine learning model. A user may correct any inconsistencies and/or label any unknowns. For example, as shown on the left-hand side, a user has added a recognition label to a face that was previously detected by not recognized (the user has labeled the face as “stuart”). Further, a user may label clusters and/or refine them. For example, on the right-hand side, a user has added a label to cluster (the label of “alex”) and has also removed two faces that were incorrectly included in the cluster.

Referring again to Fig. 3, the proposed system can also include a cloud system (e.g., one or more server devices) that can perform model training, model update and face clustering. The cloud can generate face models and face clusters that can be stored in the shared database.

In some implementations, training the models may occur offline or online. Training the models (e.g., the detection model) can include labeling images that include faces with bounding boxes that show the location of faces within the images. Training the embedding model can be performed using a triplet training scheme in which two images in a triplet depict a same person’s face while the third image in the triplet shows a different person’s face. Training the models can be performed using backpropagation techniques.

In some implementations, the heaviest inferences may be left on the cloud. For instance, model training, model updating, and/or face clustering may be carried out on the cloud. As an example, Fig. 8 shows performance of clustering and generation of face-specific models on embeddings generated for faces.

In some implementations, information input via the companion application may be uploaded to the cloud in order to supplement the machine-learning models (e.g., modify the clusters and/or the training data). As an example, Fig. 7 shows face model updating and cluster updating based on user input received via the companion application. The updated models can then be sent back to the AT Device for use on newly captured imagery.

In some implementations, the proposed system can include person tracking and/or re-identification capabilities. For example, the system may have capabilities to track a person through multiple areas viewed by multiple video devices. In some cases, a person’s face may not be necessary for the robotic device to identify the person. For example, the “on-device” processing may include person detection and/or person tracking. The cloud system may perform person embedding and person re-identification (e.g., in the form of multi-camera tracking). Further, an external device can provide a user with detection event history as well as alerts (e.g., the person tracking may be utilized as a security system).

Thus, in some embodiments, the system described above with reference to Figs. 4-8 can allow for communication directly between AT Devices or communication between AT Devices that is intermediates by a cloud system. The communication can allow the devices to pass information such as face detections/embeddings and/or person detections/embeddings from one device to another or to a shared central system. This can allow for person re-identification across multiple scenes or cameras.

In one example, for on-device people detection and tracking, people boxes can be extracted from frames using the MultiBox inference model (e.g., at 900 ms on Pi 3). Tracking people across frames can be done with optical tracking for most frames and/or by re-running MultiBox for some frames.

In a gateway or cloud person embedding scheme, people embeddings can be extracted from boxes. For example, the PersonNet inference model can be used. In some implementations, the process of extracting people embedding from boxes may be carried out on the cloud and/or gateway if the cost of including the model is higher than desired to include the machine-learning process “on-device”. However, the people embeddings can be shared. For example, one unique embedding per image may be identified. For example, there may be a synchronized gallery of people embeddings seen by more than one camera.

In some implementations, a person embedding may be mapped to a person model. For example, there may be a model per person while each person may have more than one embeddings. In some implementations embeddings may expire due to appearance being transient (e.g., clothes and hairstyle may change).

In some implementations, a person may be re-identified in a second device after being identified in a first. For example, a second device may create a new embedding of a person which may be compared to gallery models. The comparisons may output a similarity score (e.g., a higher level of similarity may result in a high score). If the similarity score does not reach a certain threshold a new model may be inserted (e.g., the model determines that the person is someone not previously identified).

One example interaction may be as follows: a person may enter an elevator in a building. There may be a robotic device positioned from the perspective of looking into the elevator from the front. The robotic device may take a video of the person and use machine learning to identify the person. The person may exit the elevator and walk down a hallway where another robotic device may be positioned from the perspective of looking down the hallway. The person may walk while looking downwards, thereby not allowing the second robotic device to capture a video image of the person’s face (or only limited video of the person’s face). The second robotic device could use information gathered by the first robotic device and predict that the person walking down the hallway is the same person as the person in the elevator.

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

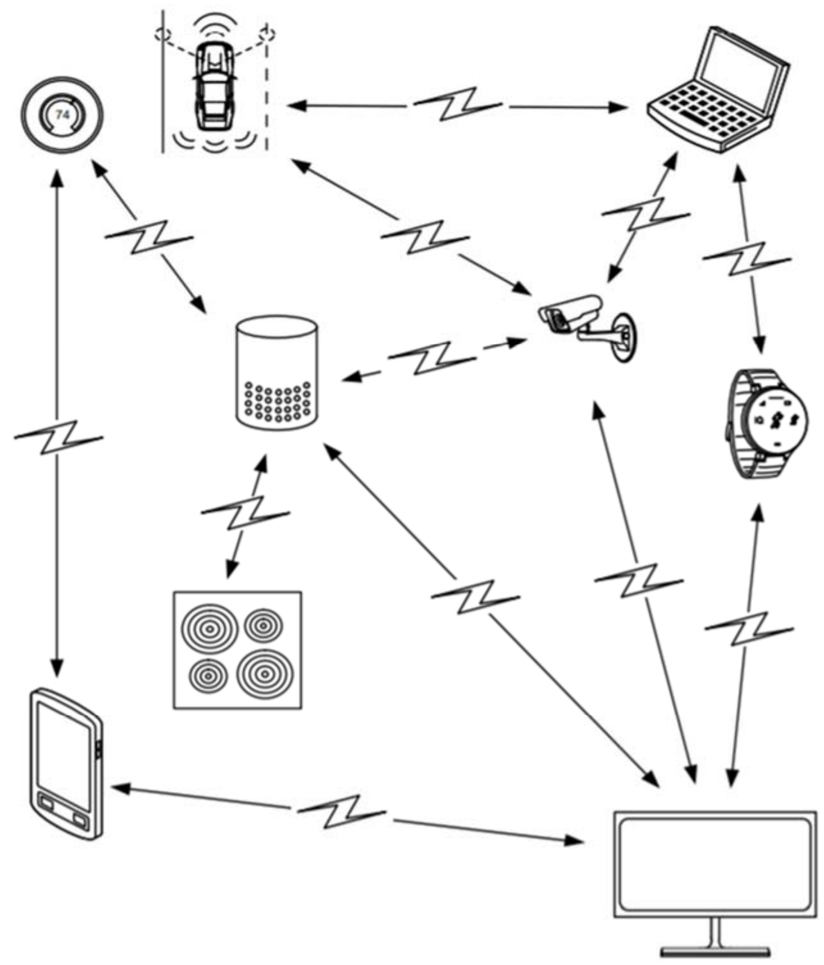

Fig. 1

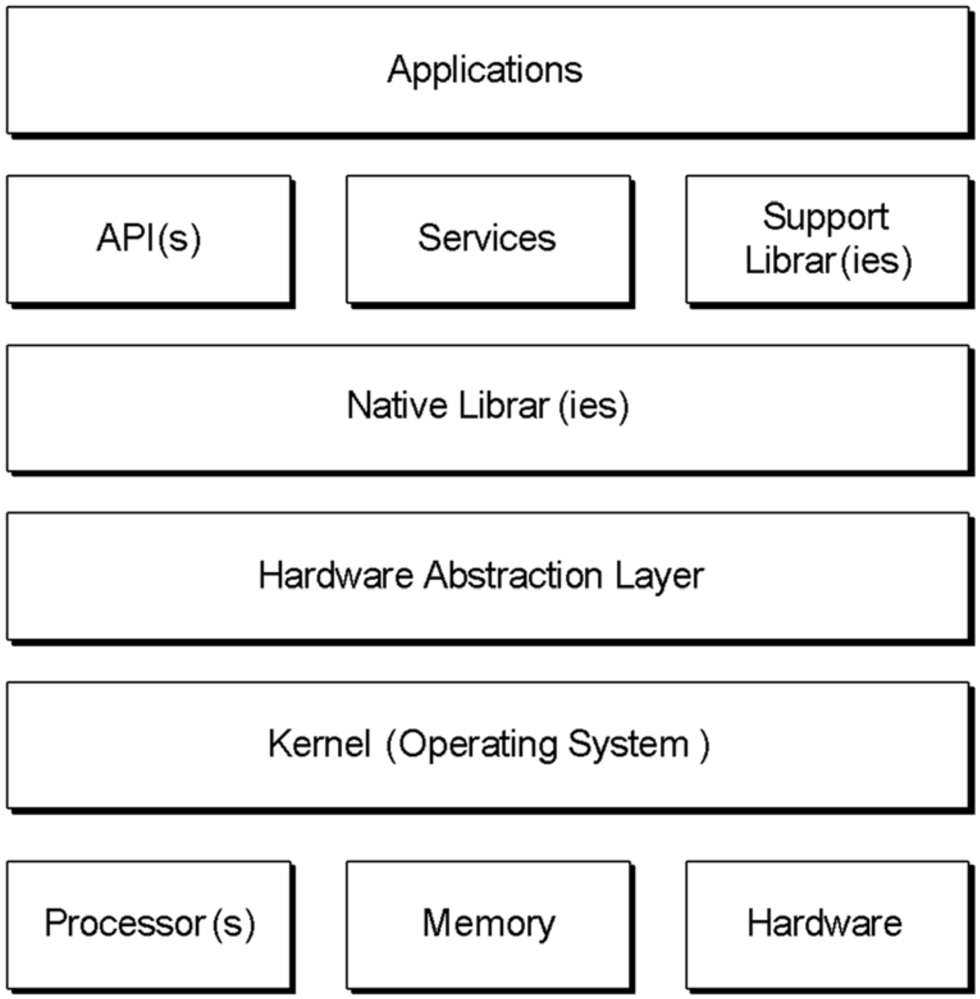
Applications
API(s)
Services
Support Librar(ies)
Native Librar(ies)
Hardware Abstraction Layer
Kernel (Operating System)
Processor(s)
Memory
Hardware

Fig. 2

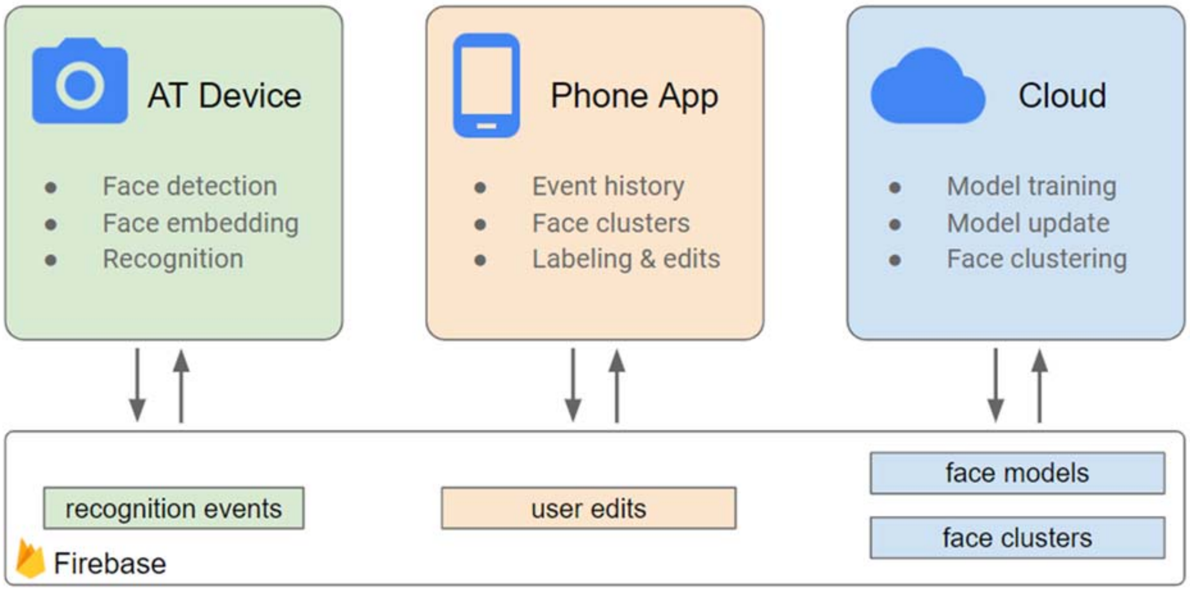
AT Device
Face detection
Face embedding
Recognition
Phone App
Event history
Face clusters
Labeling & edits
Cloud
Model training
Model update
Face clustering
recognition events
user edits
face models
face clusters
Firebase

Fig. 3

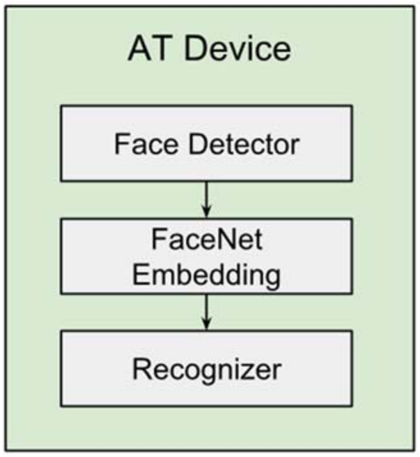
AT Device
Face Detector
FaceNet Embedding
Recognizer

Fig. 4

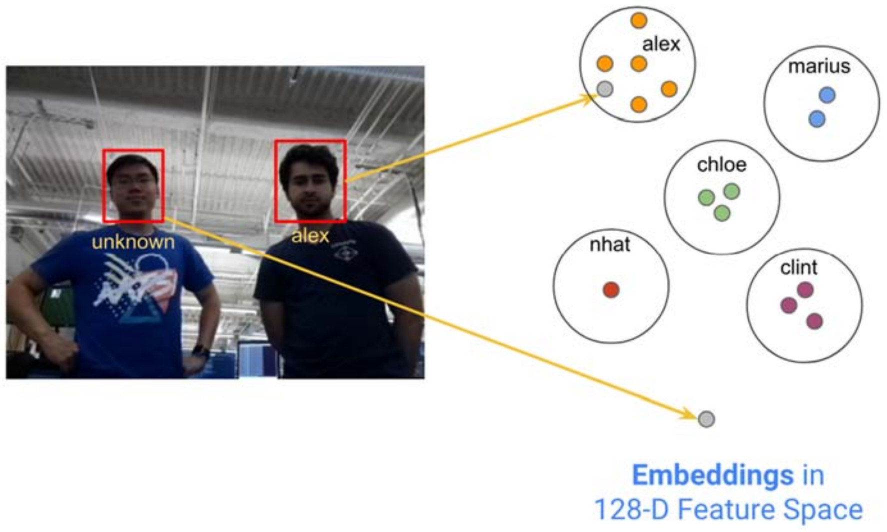
unknown
alex
alex
marius
chloe
nhat
clint
Embeddings in 128-D Feature Space

Fig. 5

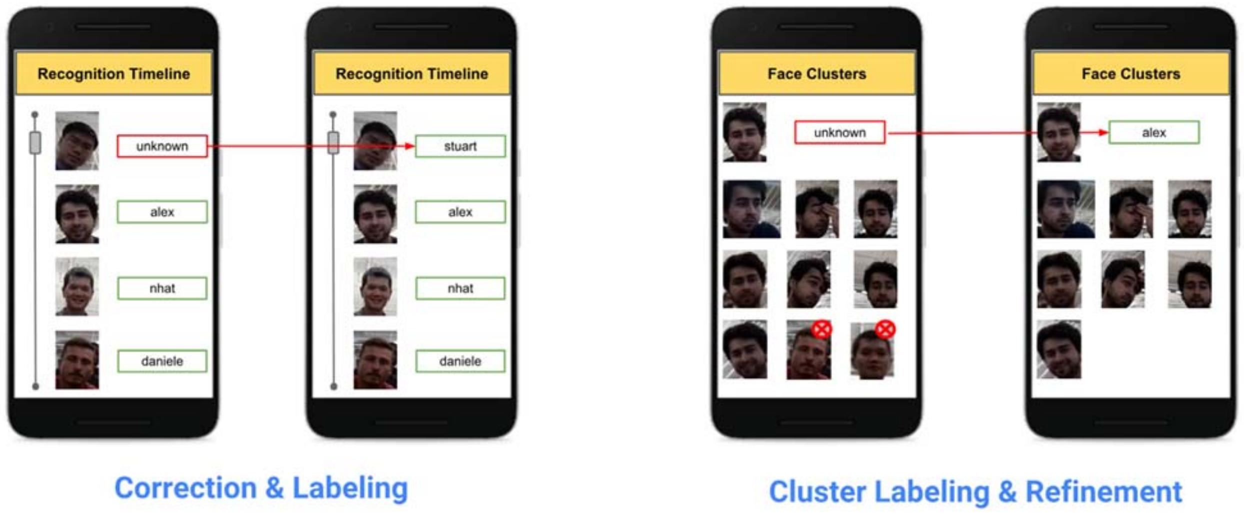
Recognition Timeline
unknown
alex
nhat
daniele
Recognition Timeline
stuart
alex
nhat
daniele
Correction & Labeling
Face Clusters
unknown
Face Clusters
alex
Cluster Labeling & Refinement

Fig. 6

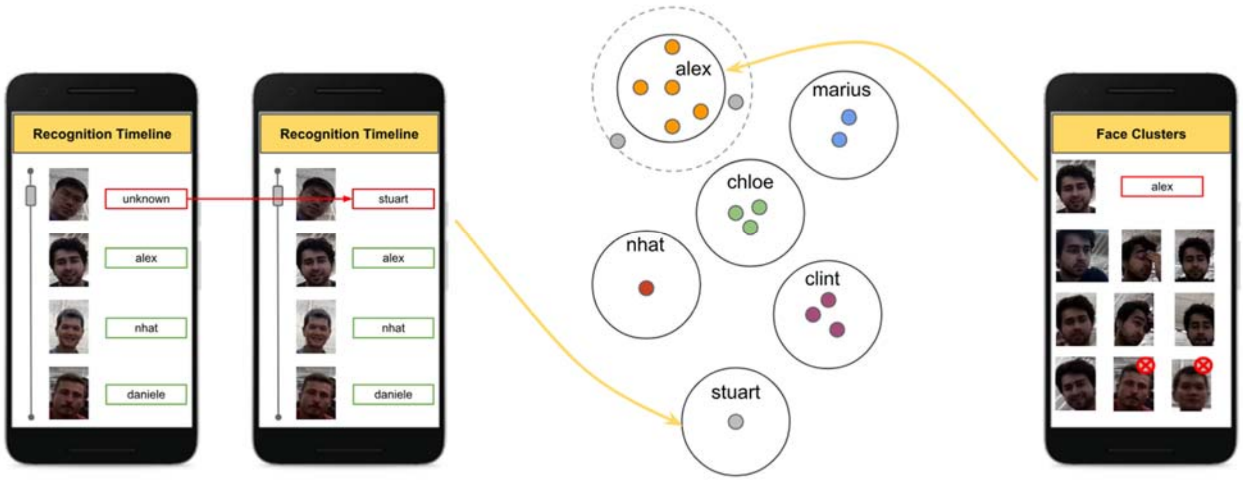
Recognition Timeline
unknown
alex
nhat
daniele
Recognition Timeline
stuart
alex
nhat
daniele
alex
marius
chloe
nhat
clint
stuart
Face Clusters
alex

Fig. 7

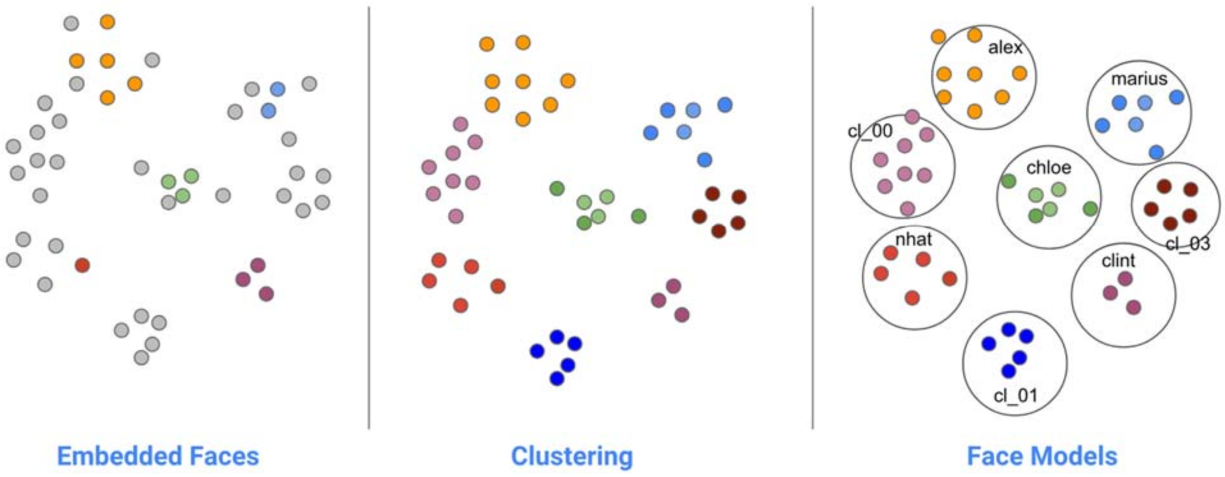
alex
marius
cl_00
chloe
cl_03
nhat
clint
cl_01
Embedded Faces
Clustering
Face Models

Fig. 8