Automatic optimization of image capture on mobile devices by human and non-human agents

Sophie Lebrecht, Mark Desnoyer, Nick Dufour, Zhihao Li,
Nicole A. Halmi, David L. Sheinberg, Michael J. Tarr

1.1 Abstract

We present methods for the programmatic and personalized capture and enhancement of high-valence images and videos in real-time, achieved by running image selection software on a mobile phone or other digital camera to control image selection at the point of capture, as opposed to filtering and selecting images and video that have already been captured. Methods of filtering and selecting images and video with high valence are discussed in further detail in U.S. Patent Application No. 14/382,406, entitled "Method and System for Using Neuroscience to Predict Consumer Preference," filed 04 March 2013, which is incorporated herein by reference as an example and not by way of limitation.

1.2 Background

With over 3 billion images captured and shared on the Internet every day, there is an acute need to be able to programmatically scan through large volumes of video and images to extract content that may connect people to information, as measured by a variety of engagement metrics. Images are used and shared on websites as wide ranging as social media platforms, dating websites, sharing economy websites, eCommerce marketplaces, and news and media sites.

Current image selection methods include: manual selection by an untrained user or a trained brand, image, or video editor; random image selection using algorithms; or the use of deep convolutional neural networks (DCNNs) trained to identify images based on, for example, the content of the image, how similar the image is to other images that have been selected by, for example, video editors, or on the emotional engagingness or valence of the images (see U.S. Patent Application No. 14/382,406, entitled "Method and System for Using Neuroscience to Predict Consumer Preference," filed 04 March 2013 for a longer discussion of methods for automatic selection of images based on valence perception).

Historically, image selection and optimization methods have operated after the point of capture. They have required the presence of existing image(s) or video(s) in order to identify and surface the "best" image for a given context, where "best" is understood as highest valence. The methods presented here describe how highly optimized DCNNs capable of running locally on mobile devices can evaluate the emotional effectiveness of an image or video during capture, and subsequently provide real-time feedback and directions from within the camera to a human or non-human agent, allowing the agent to efficiently capture high valence images in a given environment.

1.3 Summary of Methods

We present a method for improving image capture, video capture, and live video streaming on image capturing devices, including but not restricted to mobile phone cameras and digital cameras, that can be operated by a human or non-human agent. Examples of non-human agents operating image capture devices include but are not restricted to the use of a camera on a drone, or the use of a camera in or on a car (See Figure 1). Improvements in image and video capture and live streaming detailed below include: a method to scan the visual environment, stabilize the image, select the most high-valence image or part of an image, and a method that allows high valences images or video segments to be extracted from a continuous stream of video that is being produced by a camera positioned at a targeted object or event.

We describe how highly-optimized DCNNs can be run locally on different devices and can be used to identify the images with the highest valence in real-time (defined here at 30 frames per second or faster)

as the camera moves and ingests imagery from the environment. The DCNNs described below have been optimized to work efficiently within typical mobile device memory and power constraints while operating with low latency.

We present methods for selecting high valence imagery under conditions where the image capture device is operating with a strong internet signal and has the option to connect to a cloud-based server, and also under conditions where the image capture device does not have access to a strong internet signal, requiring the optimized deep neural network to operate locally on the image capture device. The methods employed in the latter case include a series of optimizations and efficiencies for running deep network models on a mobile device which will be explained in greater detail in sections 3 and 4 below.

We present methods for selecting high valence imagery under conditions where image capture is directed by personalized deep neural networks that have been trained on custom image and video collections. Examples of custom image and video collections include but are not limited to: an individual's personal video and image library on their mobile device, a library of product icons from an e-Commerce website, and a collection of imagery of travel locations from a travel provider website.

In situations where the visual environment does not naturally yield a high-valence image, we present methods for the automatic and guided enhancement of incoming visual content during image capture, which results in higher valence image capture. Examples of automatic and guided enhancement of incoming imagery include but are not limited to: change in camera direction, camera angle, and speed of camera movement/panning. We also present post-hoc methods for automatic or guided enhancements to the image captured. These enhancements include but are not limited to: adjusting exposure, saturation, clarity, and white balance; zoom; adding filters that yield a more positive valence perception; and cropping the image to increase the ratio of high valence regions to low valence regions in the image.

In situations where the image capture device is operated by a human agent, we present a novel User Interface (UI), which provides real-time feedback from within the camera screen to instruct the user where in their visual environment to point the camera, in an effort to maximize the chance of capturing a high-valence photograph or video. The UI also instructs the user about how quickly they should move the camera during video capture or while live streaming video. Additionally, the UI suggests or applies filters to modulate the contrast or warmth in a given image or video to increase the valence of the image or video.

In situations when the image capture device is a smartphone, we present methods for operating our image selection software through the phone's camera and through third-party applications that leverage the phone's camera.

As illustrated in Figure 4, the system incorporates a stabilizer function as described above to ensure that images and video, which are selected from an incoming stream of frames, are stable, motion corrected to minimize the possibility of capturing a blurred image or scene change as the human or non-human agent is moving and manipulating the camera in space. This is a critical part of the system, because blur in the image can cause an image to have a more negative valence perception than an unblurred image of the same scene.

2.1 Movement of camera by human agent

By running a deep neural network locally on an image and/or video capture device, we can evaluate incoming frames to determine which frames may generate the highest valence and lead to the most emotional engagement with a given audience or user. Here, we use the term "frames" to refer to groupings of incoming visual information as a result of video capture or image capture, or as a result of a human or non-human agent moving a camera through space when the camera is in image capture mode, whether or not the camera is actively capturing images. In one example embodiment and not by way of limitation, the image capture device could be a smart phone camera or a digital camera that is

manipulated by a human agent based on feedback generated from the valence predictor, here understood to be a DCNN trained to predict the valence of image and/or video content, at a rate of 30 frames per second or faster, and presented to the user in real-time as they are manipulating the camera. Figure 2 presents a human agent manipulating a smartphone mobile camera device along different planes in space directed by feedback from the valence predictor.

Feedback is provided to the human agent via a graphical User Interface (UI) overlaid on the camera application or screen, or, in the case of a smartphone, presented on either the camera or a third-party application that is leveraging the functionality of the native camera on the mobile device (See Figure 3(b)). The example UI depicted in Figure 3(a) is designed to provide feedback to the user directing them on how to capture the image, video, or live video such that the valence is maximized. For example, the UI could provide visual feedback in the form of a barometer-style indicator or heatmap based on the valence scores that the deep neural network is generating for both images and regions within images at a rate of 30 frames per second. In one embodiment, this UI directs the user to focus on a region of the user's visual environment that contains, for example, objects or faces with high valence, or regions in the environment that look best in certain lighting conditions, angle, zoom, which generate the best image, video, or live video (See Figure 2), where "best" is understood as highest valence.

In one embodiment, the UI presents the user with the option of either manually operating the image capture with the high valence image capture assistance or having the software automatically make the decision of when and how to capture an image (see on/off button in Figure 3(a)). The user can optionally override the automatic mode as desired. When operating in "assisted manual mode", the UI provides feedback to the user, which the user then manually acts upon. This feedback can include physical manipulation of the image capture device to point in a certain direction or at a certain angle. This feedback is given by the UI in order to direct the user to the most high-valence frame in the environment. When the user has correctly framed the highest valence image in an environment, they can then manually capture the image. In "automatic mode", the software again provides feedback to the user, including physical manipulation of the image capture device to point in a certain direction or at a certain angle. This feedback is given by the UI in order to direct the user to the most high-valence frame in the environment. When the user has correctly framed the highest valence image in an environment, the software automatically captures the image. The software then automatically enhances the imagery using filters and adjusting for zoom, white balance, exposure, contrast, and other characteristics that can impact valence in order to obtain the highest-valence image possible.

In situations where the human user is moving the mobile device following input from the camera via the novel UI described above, a stabilizing function allows the image that the user sees on the screen to be smoothed. This is achieved by showing the user only a subset of the total available image on the mobile device screen. As the user moves the mobile device, even if the user's movements are not smooth, the system is able to draw from the total available image to fill in blurry or otherwise unknown visual information, allowing the resulting image that is shown to the user on the mobile device's screen to be stable and unblurred.

To start the process of optimizing image capture, the camera may scan an area in space in order to evaluate the highest valence frames. In one embodiment, the human agent takes a panorama by moving the camera 360 degrees. This provides the Valence Predictor in Figure 4 information about where in space is the highest valence visual information to capture and sends signals to the UI accordingly. In another embodiment, the human agent may focus the camera on one region of the environment in front of the camera. The Valence Predictor may analyze the contents of that framed region and recommend whether the human agent should move the camera to the left, right, up, down, or any number of other subtle movements based on the valence information in a single frame. For example, when high valence visual information is located near the left edge of the frame, the UI may instruct the user to move the camera to the left, in order to determine whether there are higher valence regions near the known high valence region. In one embodiment, the user may scan their surrounding area, and may want to capture an image of a specific scene within the panorama. In the example case where the specific scene is not the highest valence scene available within the panorama, the user may opt to use assisted manual mode

in order to receive feedback from the UI about the valence of the scene they want to capture, and then opt to capture an image of the target scene. In this case, the UI may offer post-hoc valence enhancements for the captured image, including contrast adjustment, image filters, zoom, and cropping. These enhancements may be either manually approved or automatically applied.

2.2 Movement of camera by non-human agent

Similar to the methods described in section 2.1 we can evaluate incoming visual frames for valence and concomitant emotional engagement by running a DCNN locally on a mobile camera device. The DCNN may score incoming frames in real-time at 30 frames a second or greater, which may allow the network to evaluate where in space the highest valence information or activity is located. This feature vector output from the DCNN can be used to program a non-human agent or robotic device to redirect the camera to either different coordinates in space (i.e. move the camera further to the left), a different angle (i.e. tilting the camera up or down), or zooming into a region of the image with high valence within the frame of focus (see Figure 5). For example, but not by way of limitation, non-human agents or robotic devices may include a drone or a camera arm (see Figure 1).

To start the process of optimizing image capture, the camera may scan an area in space in order to locate the highest valence frames. In one embodiment, the non-human agent would take a panorama by moving the camera 360 degrees. This would provide the Valence Predictor in Figure 5 information about the location in space that contains the highest valence information so it could send signals to the Robotic Planner. In another embodiment, the non-human agent would focus on a single frame, and the Valence Predictor would analyze the contents of that frame to send information to the Robotic Planner to move the camera to the left, right, up, down, or any number of other subtle movements, based on the valence information in a single frame. In the situation where the goal is to capture something specific, the Robotic Planner can be programmed to only make minor adjustments to ensure the target remains in focus.

In one embodiment, a camera may be affixed to a stationary object such as a tripod or other base, or otherwise fixed upon a certain viewpoint or frame. The camera may use the methods described in this paper to periodically capture high valence images and video of the scene happening in front of the camera. As an example and not by way of limitation, such a scene could include a sporting event or a graduation. By capturing only high valence imagery, the stationary camera is able to increase the quality of images captured while reducing the amount of memory space required to store images in comparison to the amount of memory that a camera capturing images randomly would require in order to capture a similarly high-valence set of images. Additionally, this method of image capture may reduce the number of human agents necessary for image capture, which is useful, as an example and not by way of limitation, in high pressure image capture scenarios like the Olympics or fast-paced sporting events.

The system incorporates a stabilizer function as described above to ensure that images and video, which were selected from an incoming stream of frames are stable, motion corrected to minimize the possibility of capturing a blurred image or scene change as the human or non-human agent is moving and manipulating the camera in space. This is a critical part of the system, because blur in the image can cause an image to have a more negative valence perception than an unblurred image of the same scene. In Figure 5, the stabilizer function is incorporated into the Robotic Planner.

3.1 Implementation of deep neural network on image capture device

Deep convolutional neural networks are composed of layers of computational constructs that are termed “neurons,” and governed by many hundreds of thousands of parameters. To teach the DCNN to perform this mapping, it is trained. The training process consists of iteratively showing the DCNN many images. For the purposes of the methods described here, the training set is constructed with the goal of representing the variation in videos and images shared across popular image and video sharing websites. The DCNNs primary function is to learn a mapping from images to a vector of *n* values. These vectors

form points in a “feature space”. Because it’s not immediately clear what this feature space is, or how to perform this mapping, we learn it implicitly from a deep convolutional neural network (DCNN) trained on a corpus of images from sources as described above. Potential features that the DCNN can develop are based on the actual pixel values of the images, and therefore can be difficult to describe in human-readable language. Human-understandable features can, but are not guaranteed to, include “sunsets” or “underwater images”.

Once the DCNN is trained, it outputs guesses for the valence, or any other quality being predicted, of an image. These guesses are a series of weights referring to the importance of certain features for determining the valence of an image. These weights are provided in the form of a feature vector. The feature vector is passed to a loss function, which quantifies the degree to which the DCNN is right or wrong. The DCNN accepts this quantity, and sequentially adjusts its parameters in a way to minimize the chance of it making a similar mistake in the future via a process called gradient descent backpropagation.

While the number of layers or “neurons” that a DCNN has can vary (depending on, for example, the application), a current standard number of layers ranges between 64-96 layers of “neurons” for a full-sized DCNN. A typical mobile device has several constraints that make it challenging to deploy DCNNs on mobile phones. Running a DCNN is computationally intensive, so in order to run a DCNN on a phone, a simpler, lightweight classifier based on the output of the full-sized DCNN can be created. By optimizing the DCNN to run on a mobile device, the number of total layers is reduced in comparison to the number of layers of the full-sized original DCNN, generally by at least two orders of magnitude, to a size of fewer than 20 layers ideally, while obtaining the same outputs or responses (See Figure 7).

By reducing the number of layers, the DCNN is able to run more quickly, reducing latency, take up less space in the mobile device’s memory, and require fewer computational and power resources. While storing a DCNN locally on a phone is desirable because it reduces latency and processing time, it is also memory-intensive. In order for a deep net to be run locally on a phone, the deep net may be optimized in such a way that it fits within a given mobile device’s memory constraints. A smaller DCNN with fewer layers can more easily be stored locally in a mobile device’s memory.

Reducing the number of layers of a DCNN may be accomplished in a few ways. The DCNN may be optimized to run on a mobile device by removing layers and retraining the smaller model to achieve the same output as the larger model, but with fewer layers. In order to optimize to obtain a model with fewer layers, layers can be taken out. In one embodiment, little-used layers can be systematically removed in order to end up with a shorter, smaller, approximated version of the original DCNN, which returns the same functional results as the original DCNN. In another embodiment, instead of being removed, layers can be replaced with more compact but harder to train layers. Because the training of the model is done in the cloud, the difficulty of training the smaller model does not impact the memory or resources of the target mobile device. In another embodiment, an exact version of the original DCNN can be used on the mobile device, where the mobile device is equipped with a large enough GPU to handle the processing needs of a full-sized DCNN.

As the human or non-human agent using a device that employs the methods described in this paper moves a camera around, even when the agent is not actively taking pictures, the DCNN may take in images at an ideal rate of 30 frames per second or greater. These frames are analyzed in real time to provide information to the agent about where and how the camera should be moved in space in order to capture the highest valence image or video clip possible. Analyzing 30 frames per second may enable smooth user experience and use as a robotic control signal.

In one embodiment, the optimized deep net may run on a mobile device with available GPUs for processing. This allows the optimized deep net to be run in a parallelized fashion, allowing the scene to be sampled at a frame rate of 30 frames per second (fps) or higher. The higher the framerate, the more real-time the image capture may appear to the user.

In one embodiment, the optimized deep net may be run on a mobile device that does not GPUs available. In this example case, the optimized deep net may be run on the device's available CPU(s). In this case, the deep net is run at a degraded rate. For example, instead of sampling 30 fps or higher, the sampling framerate may be 10 fps. In this example case, the transition between the sampled images may be less fluid, but the deep net may be still provided with enough information about the visual environment to make decisions about which images are associated with the highest valence, and how the mobile device should be manipulated in space to frame the highest valence image(s).

In the cases where the optimized DCNN runs on the mobile device's GPU or CPU, it may be preferable that the DCNN runs locally whenever possible, instead of operating in the cloud. By running locally, the DCNN may return responses in near real-time with less of a drain on the mobile device's battery.

In addition to loading a generalized DCNN onto a user's mobile device, in one embodiment, the user may opt to allow the software to analyze the images that exist in the user's camera roll or photo albums on their mobile device. In this case, the software may develop a personalized model that is capable of predicting and automatically capturing images and video of visual information that the specific user may find most engaging, based on images and video the user has captured in the past. In particular embodiments, the user may specify a subset of images and video of the user that the DCNN may use to train the personalized model. In particular embodiments, the user may indicate that all images and video captured by the user on the mobile device may be used for training.

In one embodiment, a user may have access to a number of personalized DCNNs for different applications. For example, a user may have access to a general purpose DCNN, a DCNN that has been personalized for them based on their existing image and video files, and a number of DCNNs that are trained on the content of third party applications. Third party applications can include, but are not limited to, social networks, ecommerce marketplaces, sharing economy platforms, and dating applications. In one embodiment, each of these third party applications may have a DCNN personalized to suit their use case and their typical users. Depending on how the user is using her mobile device, the DCNN being used to locally predict and automatically capture high valence imagery may be swapped out via methods described below, in order to capture the highest valence imagery for a given use case.

3.2 Updating local version of the deep neural network from versions running in cloud

As discussed in section 3.1, in order to store and run a DCNN locally on a mobile device, a simplified version of the DCNN may be created, as illustrated in figure 7. In one embodiment, while a simplified version of the DCNN runs locally on a mobile device, a full implementation of the DCNN may be running on a cloud-based server. This full-sized, cloud-based DCNN is where the training and learning takes place for both the full-sized and the simplified DCNN. Figure 6 presents the flow of information that allows the DCNN, also called "Valence Predictor" in the case where the DCNN is tuned to predict the valence of images and video, to gather information about the effectiveness of the frames captured. This is indicated by the box labelled "Likes" Figure 6, which represents any form of feedback on how the selected image performs. In one embodiment this feedback may take the form of "likes" on social media sites. In another embodiment, it could be clicks on the image on a sharing economy platform, "dwell time" or speed of scrolling over image, conversions, or purchases on an eCommerce marketplace.

Once images are captured using the methods for optimized automatic high valence image capture described in this paper, any feedback about the images are transferred back to the DCNN in the cloud in the form of feature vectors. By transferring feature vectors instead of full-sized images, it is possible to reduce the strain on the mobile device's battery life and reduce the amount of bandwidth needed to update the full-sized, cloud-based DCNN. The full-sized, cloud-based DCNN takes in the feature vector information for all images for which there is available feature vector information, and uses this information to fine-tune the cloud-based DCNN, increasing the accuracy of the DCNN's predictions for the given user whose images' feature vectors were shared. Periodically, update and prediction refinement information is shared by the cloud-based DCNN with the simplified DCNN stored locally, in the form of updated weights for features. By updating weights for features, the DCNN is able to more accurately predict which features

are important when determining whether a certain image will be perceived to be high valence by a viewer. In one embodiment, this updated weight information may be shared when the user's mobile device is connected to the internet via Wi-Fi. In this way, a user-personalized DCNN can be created in the cloud and updated locally on the user's mobile device. Feature vector and feature weight information is shared with the cloud-based DCNN and the local DCNN asynchronously in order to avoid interfering with the user experience by making the camera laggy or by utilizing too great of a user's bandwidth at any given time.

In another embodiment, the methods described above can be implemented to train and refine a more generalized DCNN that is specific to a given third-party app, instead of being specific to a particular user. For example and not by way of limitation, personalized models can learn to select and/or capture images that align with the brand's visual requirements or the user's personal preferences. In one embodiment, a user's mobile device has access to an interchangeable set of DCNNs that are personalized for a specific application, user, or groups of users (see "Valence Predictor in the Cloud", "VP-1", and "VP-2" in Figure 6). In one embodiment, the user may use their mobile device's native camera application, where automatic high valence image capture is powered by a DCNN that has been personalized to be specific to that user. The user may then choose to use a sharing economy platform application, where a personalized DCNN has been trained for users of that app. In one embodiment, the mobile device may switch between the user-specific DCNN and the third-party app-specific DCNN depending on the context and depending on which application the user is using at a given moment. In one embodiment, multiple simplified DCNNs are stored locally on the user's mobile device, depending on the user's app usage habits. In another embodiment, the user may opt not to share existing images with the DCNN. In this case, the user may have access to a generalized DCNN trained to predict valence for a general audience.

4.1 Optimizing processing efficiencies

In order to run and store a DCNN locally on a mobile device most effectively, a number of processing efficiencies may be employed. The goals of these efficiencies are to: reduce latency, reduce memory load, reduce power requirements, and reduce bandwidth needs.

In one embodiment, to effect lower latency, the DCNN should be stored locally, as operating entirely from the cloud can slow down processing times, depending on the strength and speed of the mobile device's internet connection. The DCNN can also be made more shallow by reducing the number of layers in the DCNN without affecting the accuracy of the output of the DCNN, as described above.

In one embodiment, memory usage can be made more efficient by removing layers from the cloud-based DCNN for the local version of the DCNN. While a local DCNN can be deep, a shallower DCNN results in a smaller and more easily stored DCNN.

In one embodiment, power load can be made more efficient by having the local DCNN transmit feature vectors to the cloud-based DCNN and having the cloud-based DCNN transmit feature vectors to the local DCNN, instead of transmitting image information or image files between the local and the cloud-based DCNN. Because feature vectors are much smaller files than image files, less power and bandwidth are required to transmit this information. If the local and cloud-based DCNNs have been properly trained to output identical results, transmitting feature vectors may theoretically result in the same output as transmitting image files. In another embodiment, optimization between network connectivity and processing on a local device can be done, so that when the mobile device is connected via Wi-Fi to the cloud, some of the processing work can be off-lifted from the local device to the cloud, thereby reducing processing and battery strain on the local device. In another embodiment, the local DCNN can be structured to do a trade-off between total power use and total computation done. For example, but not by way of limitation, when processing is sped up on a single

core GPU, power usage is quadratic, whereas when a DCNN is split efficiently between two or more cores, power usage becomes linear.

In one embodiment, bandwidth usage can be made more efficient by transmitting feature weights or feature vectors from the cloud-based DCNN to the local DCNN periodically when the mobile device is connected to Wi-Fi, and avoiding transferring weights or image data when the mobile device only has access to a cellular data network. In another embodiment, bandwidth usage can be made more efficient by only transmitting feature vectors between the cloud-based and the local DCNN and not transmitting image information, as described above. In another embodiment, bandwidth usage can be made more efficient by optimizing between network connectivity and processing on a local device. This allows a bounded amount of the processing work to be off-lifted from the local device to the cloud when the local device has access to Wi-Fi. By bounding the amount of processing that can be done in the cloud, the amount of bandwidth used can be optimized.

5.1 Applications for these methods

The methods for in-device image capture and optimization described above can be used by a wide variety of groups to automatically capture high valence images and video more quickly, more scalably, and more efficiently than manual or other existing automated methods of capturing and identifying high valence videos and images.

The methods described above can be implemented either using a generalized or general audience DCNN or an interchangeable set of DCNNs that are personalized for a specific application, user, or groups of users. For example and not by way of limitation, personalized models can learn to select and/or capture images that align with the brand's visual requirements or the user's personal preferences. In broad terms, the methods described above are useful for cases where users create and share or post images, video, or live streaming video with the goal of getting other users to see and engage with the content they have posted. The methods described above allow for personalized models to assist in the generation of or automatically direct the generation of images that are cropped to automatically return the subsection of the image with the highest valence, and allow for correction of direction and angle of camera, speed of camera movement, zoom, and post-hoc image filtering and editing including but not limited to changing image warmth, contrast, and brightness. Examples of specific applications that would benefit from having a personalized DCNN for image selection on a mobile device include but are not limited to: social media platforms, eCommerce marketplaces, sharing economy marketplaces, live event video and image capture, brands and advertisers, dating and relationship apps, and security and surveillance video and image capture.

In an example case of social media platforms, the methods for in-device image capture and optimization described above could be incorporated into the platform's mobile application camera function. By employing the methods described here, the platform's users are prompted to capture more high valence (and thereby more visually engaging) images, video, and live streaming video to share on the platform, thereby driving more engagement on the social media platform and increasing the time users spend on the site. Because the methods described above incorporate feedback on image success from users in the form of click data (where success is defined as engagement with or clicks on an image), a personalized model can be created for social media platforms in general, a specific social media platform, or a specific user on the social media platform.

In an example case of eCommerce marketplaces or companies, the methods for in-device image capture and optimization described above could be incorporated into the marketplace's mobile application camera function, in mobile applications that allow users to take a picture or video of the items for sale using the marketplace's mobile app. Using high valence imagery to depict goods for sale results in increased clicks on the goods represented by high valence imagery, which increases the likelihood of purchase. The methods described above guide users to automatically create high valence images, video, and live streaming video. Because the methods described above incorporate feedback on image success from

users in the form of click data (where success is defined as engagement with or clicks on an image), a personalized model can be created for eCommerce marketplaces in general, a specific eCommerce marketplace, or a specific user on the eCommerce marketplace. Personalized models for eCommerce can be created by training on this click data in addition to training on existing images on the eCommerce platform, allowing the model to more accurately predict which types of images may be perceived to have high valence by the specific user groups using the eCommerce marketplace.

In another example and not by way of limitation, sharing economy platforms may allow a user to, for example, put up a listing of their home so that other users can stay in the home for a fee, or allow a user to, for example, offer their car and driving services for a fee. Because sharing economy platforms rely on user generated image and video content, it is crucial for both the platform's success and the user's success on the platform that images be engaging. In the case of sharing economy platforms, the methods for in-device image capture and optimization described above could be incorporated into the platform's mobile application camera function in mobile applications that allow users to take a picture or video of the items or services for share, or the profile picture of the user providing the services using the platform's mobile app. Using high valence imagery to depict available goods or services results in increased clicks on the goods or services represented by high valence imagery, which increases the likelihood of a transaction. The methods described above guide users to automatically create high valence images, video, and live streaming video. By directing users to automatically create high valence images and video, platforms are more likely to be populated by maximally engaging imagery that is "on brand". Automatic image capture also allows the platform to scale to bring on new users while maintaining the quality of the images shared on their site more easily. Additionally, because the methods described above incorporate feedback on image success from users in the form of click data (where success is defined as engagement with or clicks on an image), a personalized model can be created for sharing economy platforms in general, a specific sharing economy platform, or a specific user on the sharing economy platform.

In an example case of image and video capture at live events, the methods described above could be incorporated directly into the cameras being used to capture imagery of the event. Cameras can take multiple forms, including but not limited to: cameras attached to drones and professional digital video and still image cameras being used by photographers or unmanned on tripods at the event (see Figure 1). By incorporating the methods of automatic high valence image capture into cameras used at live events, image capture can be made more efficient, by directing the operation of the non-human agents (in this example, drones, and unmanned cameras) to automatically capture high valence images and video, and by directing human agents to take high valence images and video. Because time is an important factor in live events, it is crucial that a human or non-human agent is efficiently and alertly capturing images as events unfold. The methods described above allow human and non-human agents to take the most engaging, high valence images and video possible at any given moment in time. Additionally, because the methods described above incorporate feedback on image success from users in the form of click data (where success is defined as engagement with or clicks on an image), a personalized model can be created for types of live events in general (for example: sporting events or concerts), or a specific live event (for example: the Olympics or the Burning Man festival).

In an example case of image capture for brands and advertisers, the methods described above for automatic high valence image and video capture can be employed to enable the brands and advertisers to efficiently capture high quality brand content to be used for advertisements or web sites, for example. In the case of image capture by a brand with limited time and financial resources, it is valuable to capture the most useful and high quality content in as little time as possible. The methods described above, when employed on digital cameras or mobile phones, enable brands and advertisers to capture high valence images and video at, for example but not by way of limitation, a photo shoot while minimizing the amount of low valence images and video that are created. Additionally, because the methods described above incorporate feedback on image success from users in the form of click data (where success is defined as engagement with or clicks on an image) and can incorporate existing brand or advertiser images, a personalized model can be created for categories of brands, or a specific brand or advertiser.

In an example case of image capture for dating and relationship websites with mobile applications, the methods described above can be incorporated into the mobile phone camera associated with the mobile app and assist users in taking and sharing the highest valence, and therefore most engaging and appealing, images of themselves to be used as their dating app profile picture or video. In this context, high valence images and video can result in higher (and more successful) user engagement on the app. Additionally, because the methods described above incorporate feedback on image success from users in the form of click data (where success is defined as engagement with or clicks on an image) and can incorporate existing dating website images, a personalized model can be created for dating and relationship websites in general, a specific dating and relationship website, or a specific user on a dating and relationship website.

In an example case of image capture for security and surveillance applications, the methods described above can be incorporated into a mobile phone camera or digital camera that is set up to be used as a surveillance or tracking device. When the methods described above are incorporated into mobile devices or digital cameras, the devices can be automatically directed to focus on and zoom into the areas of high valence and high visual engagingness that are often correlated with faces and action in a scene.

In one embodiment of the methods described above, a certain subject may use multiple mobile web applications with built in cameras that utilize the methods for automatic high valence image capture described above. For example, one subject might use a sharing economy platform, a dating app, and a social media platform. When the subject opens the camera in her sharing economy platform to take pictures of her apartment in order to create a listing for her apartment on the platform, the methods described above may allow the application to detect the user and the use case (in this case, a sharing economy platform's mobile site) and therefore choose the most suitable DCNN for the user and the use case. By using a DCNN that is personalized for the user and/or use case, the subject is able to easily capture relevant and engaging high valence imagery for her particular use case. When the same subject logs onto a dating app and creates a profile there, the methods described above may allow for a second DCNN, personalized for the user and/or use case, to be utilized for automatic image capture as the subject takes her profile picture for her profile on the dating website. In this way, different DCNNs can be employed for different use cases, allowing a subject to consistently create high valence and relevant content across various use cases.

6.1 Systems and Methods

FIG. 9A, and FIG. 9B illustrate exemplary possible system embodiments. The more appropriate embodiment will be apparent to those of ordinary skill in the art when practicing the present technology. Persons of ordinary skill in the art will also readily appreciate that other system embodiments are possible.

FIG. 9A illustrates a conventional system bus computing system architecture 900 wherein the components of the system are in electrical communication with each other using a bus 905. Exemplary system 900 includes a processing unit (CPU or processor) 910 and a system bus 905 that couples various system components including the system memory 915, such as read only memory (ROM) 920 and random access memory (RAM) 925, to the processor 910. The system 900 may include a cache of high-speed memory connected directly with, in close proximity to, or integrated as part of the processor 910. The system 900 may copy data from the memory 915 and/or the storage device 930 to the cache 912 for quick access by the processor 910. In this way, the cache may provide a performance boost that avoids processor 910 delays while waiting for data. These and other modules may control or be configured to control the processor 910 to perform various actions. Other system memory 915 may be available for use as well. The memory 915 may include multiple different types of memory with different performance characteristics. The processor 910 may include any general purpose processor and a hardware module or software module, such as module 1 932, module 2 934, and module 3 936 stored in storage device 930, configured to control the processor 910 as well as a special-purpose processor where software instructions are incorporated into the actual processor design. The processor 910 may

essentially be a completely self-contained computing system, containing multiple cores or processors, a bus, memory controller, cache, etc. A multi-core processor may be symmetric or asymmetric.

To enable user interaction with the computing device 900, an input device 945 may represent any number of input mechanisms, such as a microphone for speech, a touch-sensitive screen for gesture or graphical input, keyboard, mouse, motion input, speech and so forth. An output device 935 may also be one or more of a number of output mechanisms known to those of skill in the art. In some instances, multimodal systems may enable a user to provide multiple types of input to communicate with the computing device 900. The communications interface 940 may generally govern and manage the user input and system output. There is no restriction on operating on any particular hardware arrangement and therefore the basic features here may easily be substituted for improved hardware or firmware arrangements as they are developed.

Storage device 930 is a non-volatile memory and may be a hard disk or other types of computer readable media which may store data that are accessible by a computer, such as magnetic cassettes, flash memory cards, solid state memory devices, digital versatile disks, cartridges, random access memories (RAMs) 925, read only memory (ROM) 920, and hybrids thereof.

The storage device 930 may include software modules 932, 934, 936 for controlling the processor 910. Other hardware or software modules are contemplated. The storage device 930 may be connected to the system bus 905. In one aspect, a hardware module that performs a particular function may include the software component stored in a computer-readable medium in connection with the necessary hardware components, such as the processor 910, bus 905, display 935, and so forth, to carry out the function.

FIG. 9B illustrates a computer system 950 having a chipset architecture that may be used in executing the described method and generating and displaying a graphical user interface (GUI). Computer system 950 is an example of computer hardware, software, and firmware that may be used to implement the disclosed technology. System 950 may include a processor 955, representative of any number of physically and/or logically distinct resources capable of executing software, firmware, and hardware configured to perform identified computations. Processor 955 may communicate with a chipset 960 that may control input to and output from processor 955. In this example, chipset 960 outputs information to output 965, such as a display, and may read and write information to storage device 970, which may include magnetic media, and solid state media, for example. Chipset 960 may also read data from and write data to RAM 975. A bridge 980 for interfacing with a variety of user interface components 985 may be provided for interfacing with chipset 960. Such user interface components 985 may include a keyboard, a microphone, touch detection and processing circuitry, a pointing device, such as a mouse, and so on. In general, inputs to system 950 may come from any of a variety of sources, machine generated and/or human generated.

Chipset 960 may also interface with one or more communication interfaces 990 that may have different physical interfaces. Such communication interfaces may include interfaces for wired and wireless local area networks, for broadband wireless networks, as well as personal area networks. Some applications of the methods for generating, displaying, and using the GUI disclosed herein may include receiving ordered datasets over the physical interface or be generated by the machine itself by processor 955 analyzing data stored in storage 970 or 975. Further, the machine may receive inputs from a user via user interface components 985 and execute appropriate functions, such as browsing functions by interpreting these inputs using processor 955.

It may be appreciated that exemplary systems 900 and 950 may have more than one processor 910 or be part of a group or cluster of computing devices networked together to provide greater processing capability.

For clarity of explanation, in some instances the present technology may be presented as including individual functional blocks including functional blocks comprising devices, device components, steps or routines in a method embodied in software, or combinations of hardware and software.

In some embodiments the computer-readable storage devices, mediums, and memories may include a cable or wireless signal containing a bit stream and the like. However, when mentioned, non-transitory computer-readable storage media expressly exclude media such as energy, carrier signals, electromagnetic waves, and signals per se.

Methods according to the above-described examples may be implemented using computer-executable instructions that are stored or otherwise available from computer readable media. Such instructions may comprise, for example, instructions and data which cause or otherwise configure a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Portions of computer resources used may be accessible over a network. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, firmware, or source code. Examples of computer-readable media that may be used to store instructions, information used, and/or information created during methods according to described examples include magnetic or optical disks, flash memory, USB devices provided with non-volatile memory, networked storage devices, and so on.

Devices implementing methods according to these disclosures may comprise hardware, firmware and/or software, and may take any of a variety of form factors. Typical examples of such form factors include laptops, smart phones, small form factor personal computers, personal digital assistants, and so on. Functionality described herein also may be embodied in peripherals or add-in cards. Such functionality may also be implemented on a circuit board among different chips or different processes executing in a single device, by way of further example.

Typically, a library such as TensorFlow, Caffe, Torch, etc. may be used. These optionally employ another library called CUDA, distributed by NVIDIA, that executes operations directly on a specialized hardware device called a GPU ('graphical' processing unit, owing its name to their original use in rendering frames from video games).

The instructions, media for conveying such instructions, computing resources for executing them, and other structures for supporting such computing resources are means for providing the functions described in these disclosures.

Figure 1

Non-human agents capturing images as guided by automatic high valence image capture methods

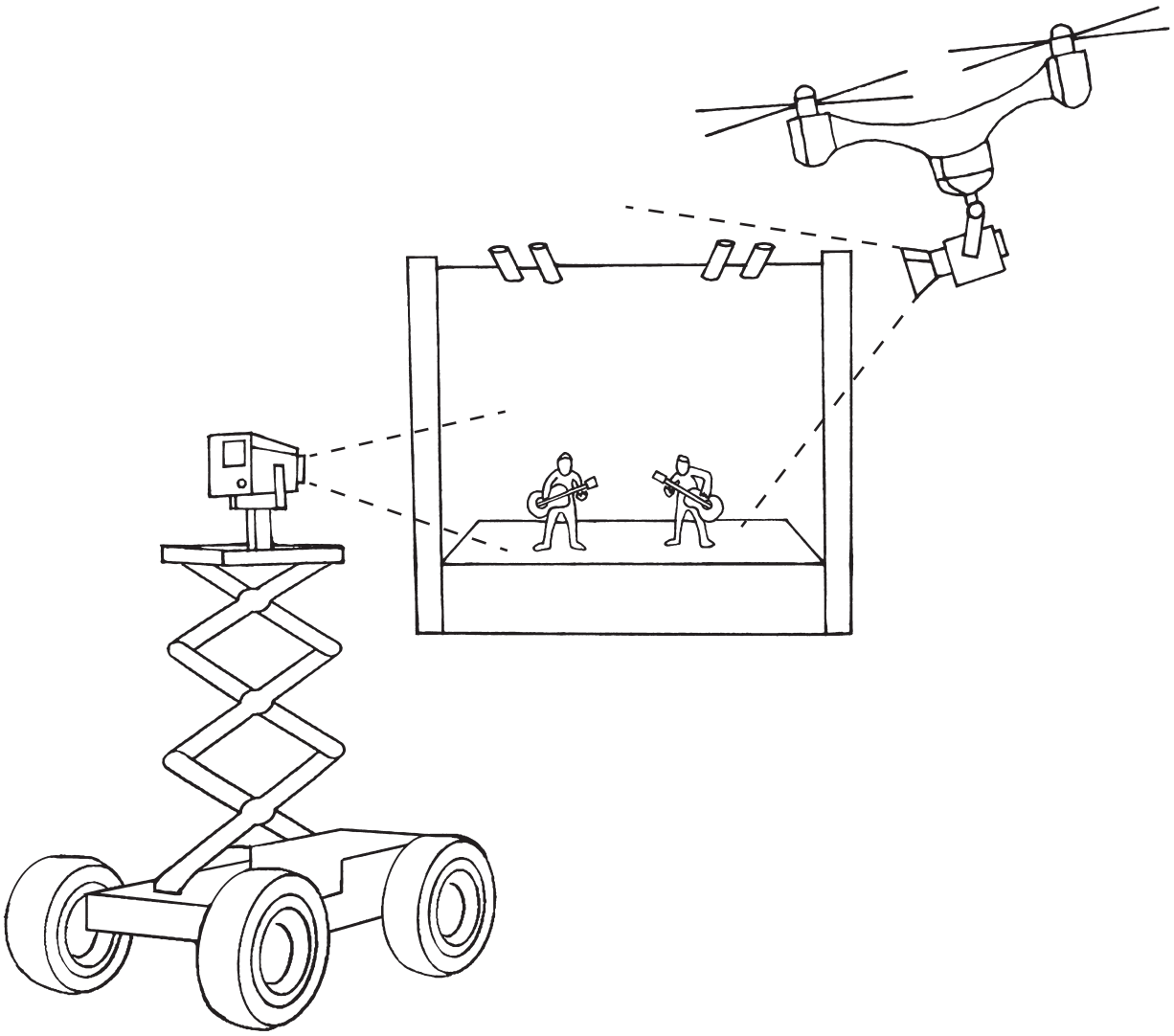


Figure 2

Human agent moving mobile device through space as directed UI powered by high valence automatic image capture methods

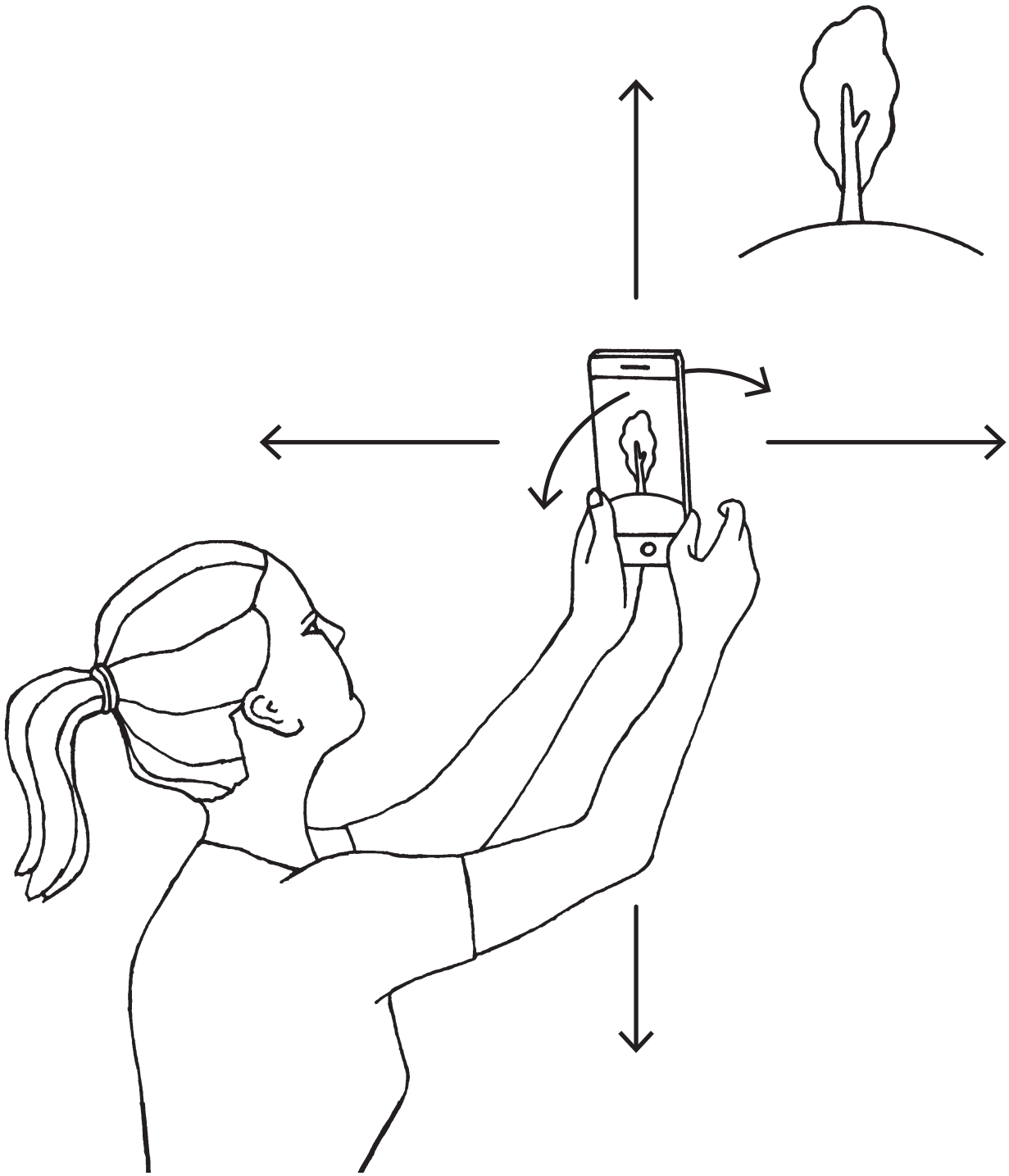


Figure 3a

UI for native mobile device camera

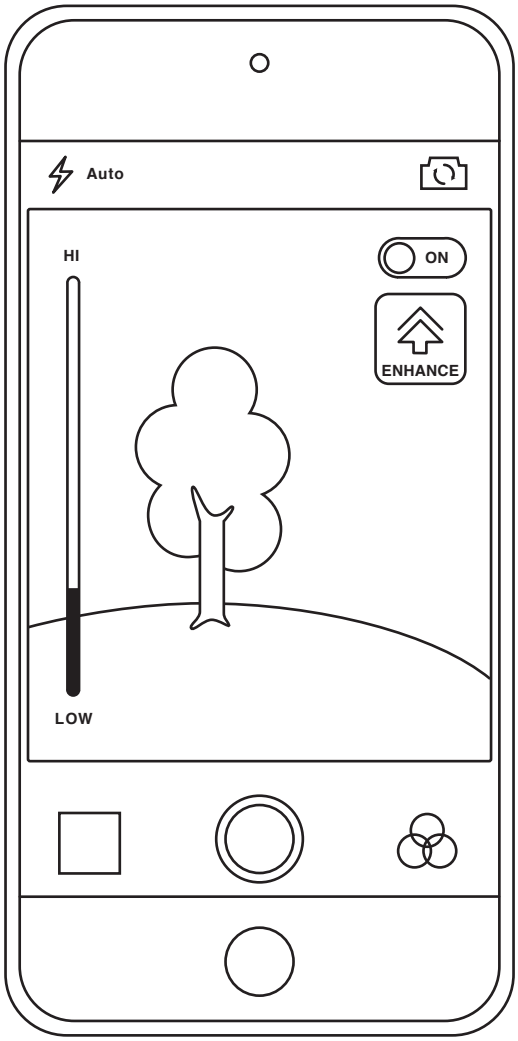


Figure 3b

UI for third-party app camera

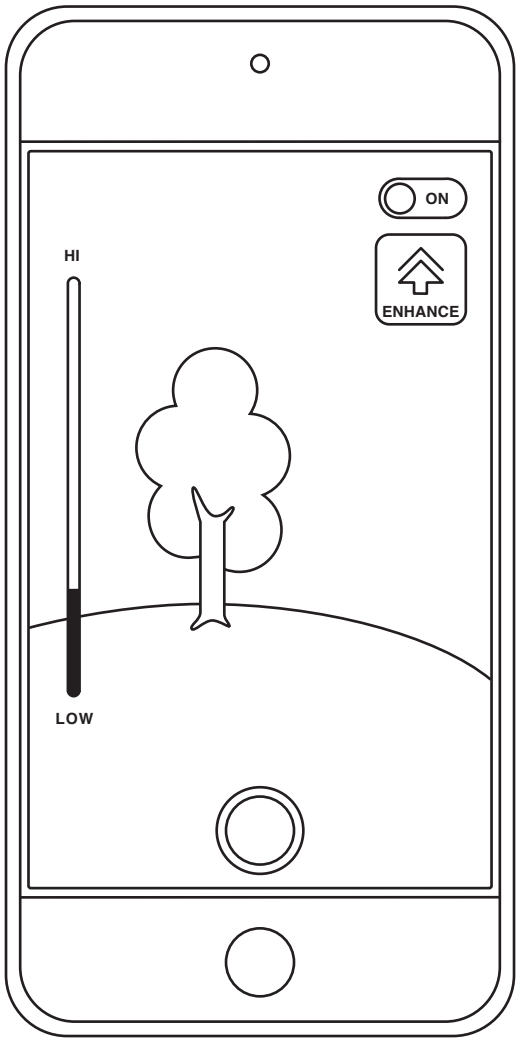


Figure 5

Non-human agent image capture flow

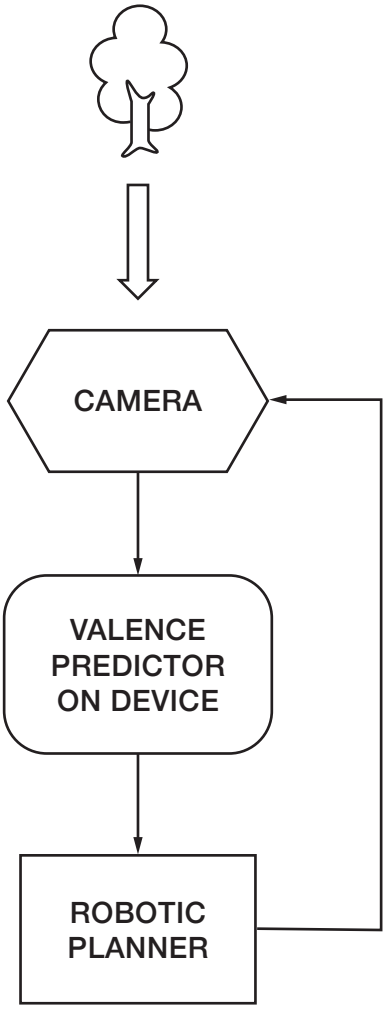


Figure 4

Human agent image capture flow

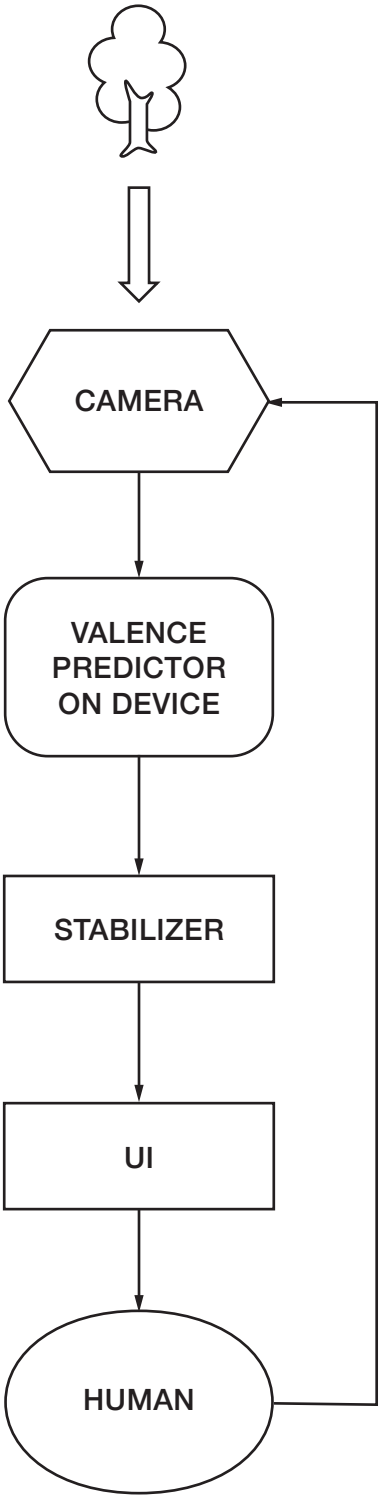


Figure 6

Flow for updating local Valence Predictor and Valence Predictor in the cloud

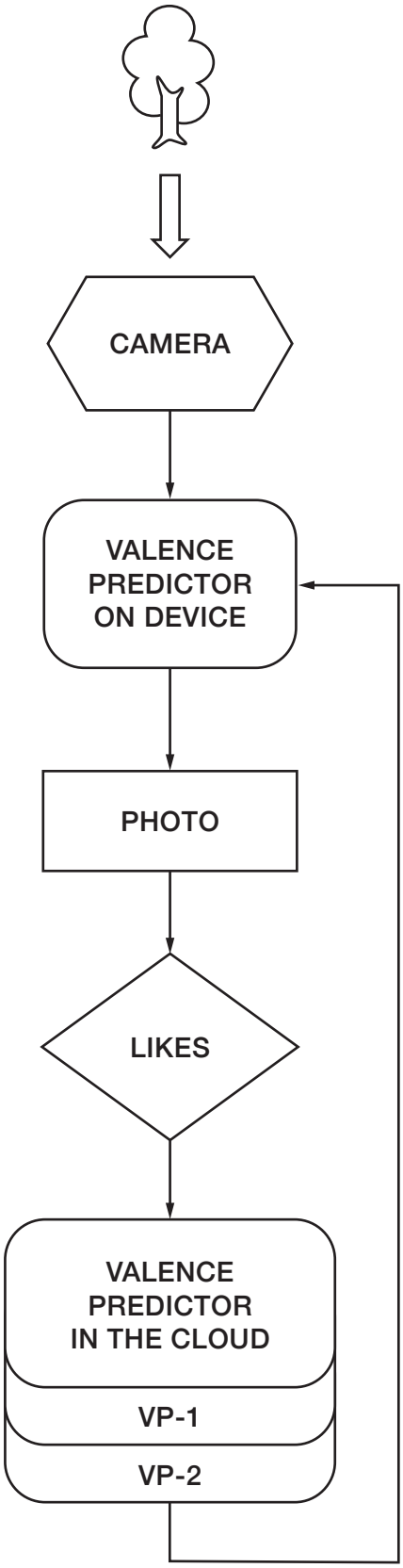


Figure 7

Valence Predictor DCNN optimized
to run on a mobile device

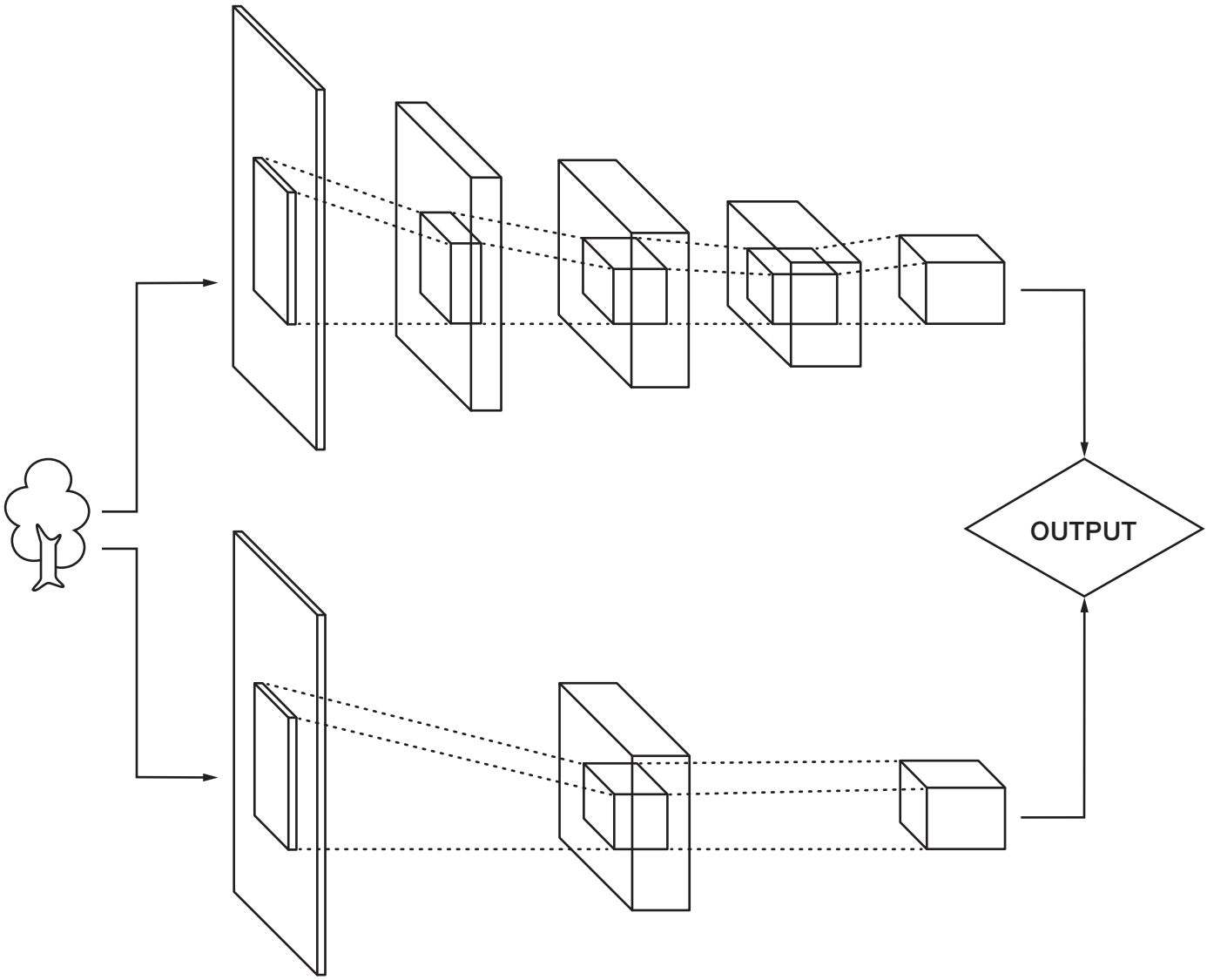


FIG. 9A

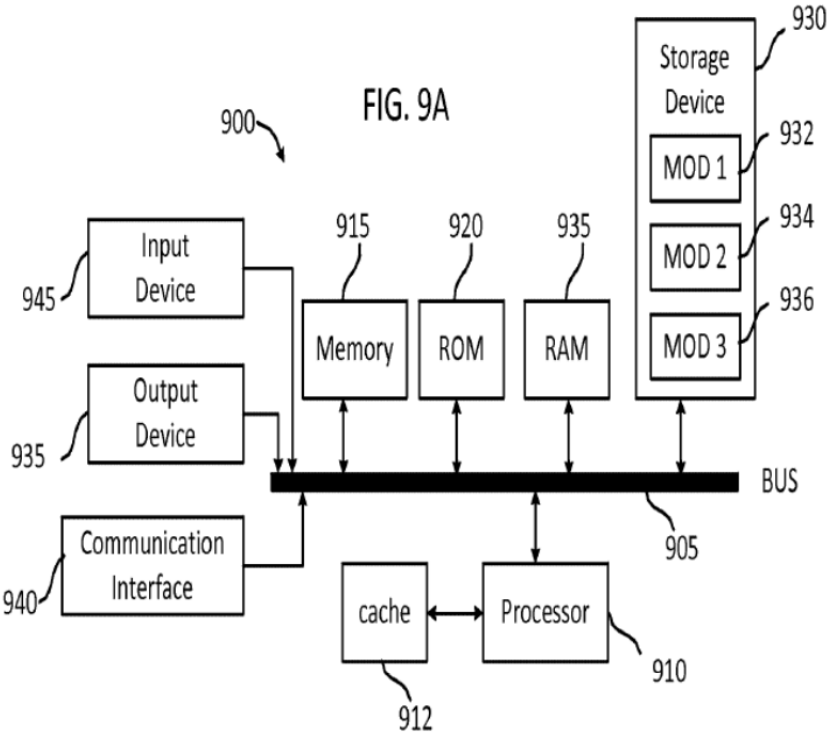
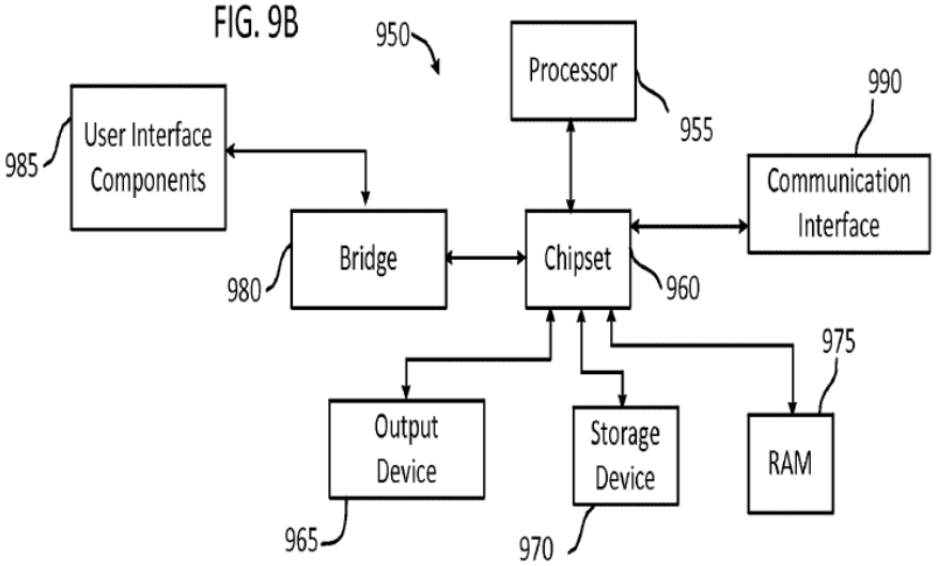

900
Input Device
945
915
Memory
920
ROM
935
RAM
Storage Device
930
MOD 1
932
MOD 2
934
MOD 3
936
Output Device
935
BUS
905
Communication Interface
940
cache
Processor
910
912

FIG. 9B


950
Processor
955
990
User Interface Components
985
Communication Interface
Bridge
980
Chipset
960
975
Output Device
965
Storage Device
970
RAM

Electronic Acknowledgement Receipt

EFS ID:	26348780
Application Number:	62362202
International Application Number:	
Confirmation Number:	8673
Title of Invention:	Automatic Optimization of Image Capture on Mobile Devices by Human and Non-Human Agents
First Named Inventor/Applicant Name:	Sophie Lebrecht
Customer Number:	5073
Filer:	Hyun Kyu Christopher Han/Doreen Hutton
Filer Authorized By:	Hyun Kyu Christopher Han
Attorney Docket Number:	084024.0108
Receipt Date:	14-JUL-2016
Filing Date:	
Time Stamp:	12:54:25
Application Type:	Provisional

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$ 130
RAM confirmation Number	22656
Deposit Account	020384
Authorized User	BAKER & BOTTS, LLP

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 CFR 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 CFR 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 CFR 1.19 (Document supply fees)

Charge any Additional Fees required under 37 CFR 1.20 (Post Issuance fees)

Charge any Additional Fees required under 37 CFR 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Drawings-only black and white line drawings	084024-0108-Drawings.pdf	632123 8f17cc8f97bb07b0eedb5881efe369ca82c9c800	no	7
Warnings:					
Information:					
2	Specification	084024-0108-Provisional-Application-14JULY2016.pdf	140133 eb81747a4e120822ff7eba38e1fc7599ccc2ae64	no	12
Warnings:					
Information:					
3	Application Data Sheet	084024-0108-ADS-14JULY2016.pdf	1823171 92b7fcac78615d6a57f13733f00f44f3051693cd	no	10
Warnings:					
Information:					
4	Fee Worksheet (SB06)	fee-info.pdf	29812 7afbfc4bca1a65ee9bbbc88c42634b34e2374c82	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	2625239		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.