US 20250259405A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2025/0259405 A1**

**Korolev et al.** (43) **Pub. Date: Aug. 14, 2025**

(54) **3D MASK GENERATION**

(71) Applicant: **Snap Inc.**, Santa Monica, CA (US)

(72) Inventors: **Sergei Korolev**, Marina del Rey, CA (US); **Aleksei Stoliar**, Marina del Rey, CA (US); **Mikhail Vasilkovskii**, Playa Vista, CA (US)

(21) Appl. No.: **18/436,199**

(22) Filed: **Feb. 8, 2024**

**Publication Classification**

(51) **Int. Cl.**

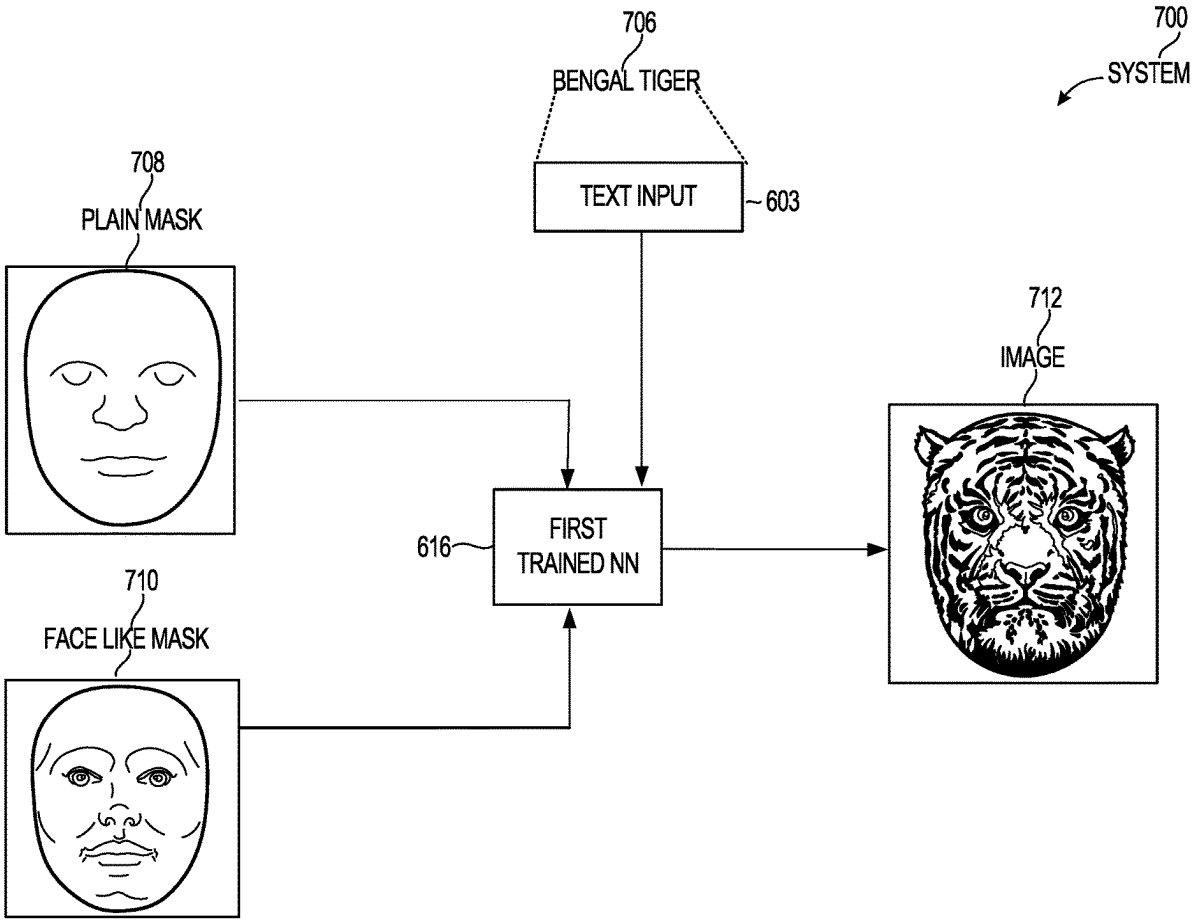
| | |
|---|---|
| ***G06T 19/20*** | (2011.01) |
| ***G06T 7/50*** | (2017.01) |
| ***G06T 15/50*** | (2011.01) |

(52) **U.S. Cl.**

CPC ..... ***G06T 19/20*** (2013.01); ***G06T 7/50*** (2017.01); ***G06T 15/506*** (2013.01); *G06T 2207/10016* (2013.01); *G06T 2207/20081* (2013.01); *G06T 2207/20084* (2013.01); *G06T 2207/20092* (2013.01); *G06T 2207/30196* (2013.01); *G06T 2219/2012* (2013.01); *G06T 2219/2021* (2013.01)

(57) **ABSTRACT**

A three-dimensional (3D) trainable mask is geometrically adjusted based on two-dimensional (2D) images of a target image. The 3D trainable mask is applied to a face of a user within an image. Example methods include rendering a trainable three-dimensional (3D) mask to generate a two-dimensional (2D) image, adding noise to the 2D image to generate a 2D image with added noise, inputting the 2D image with added noise and input text into a trained neural network to generate a predicted noise of the 2D image with added noise, determining a loss between the 2D image with added noise and the predicted noise, and updating the trainable 3D mask based on the loss.



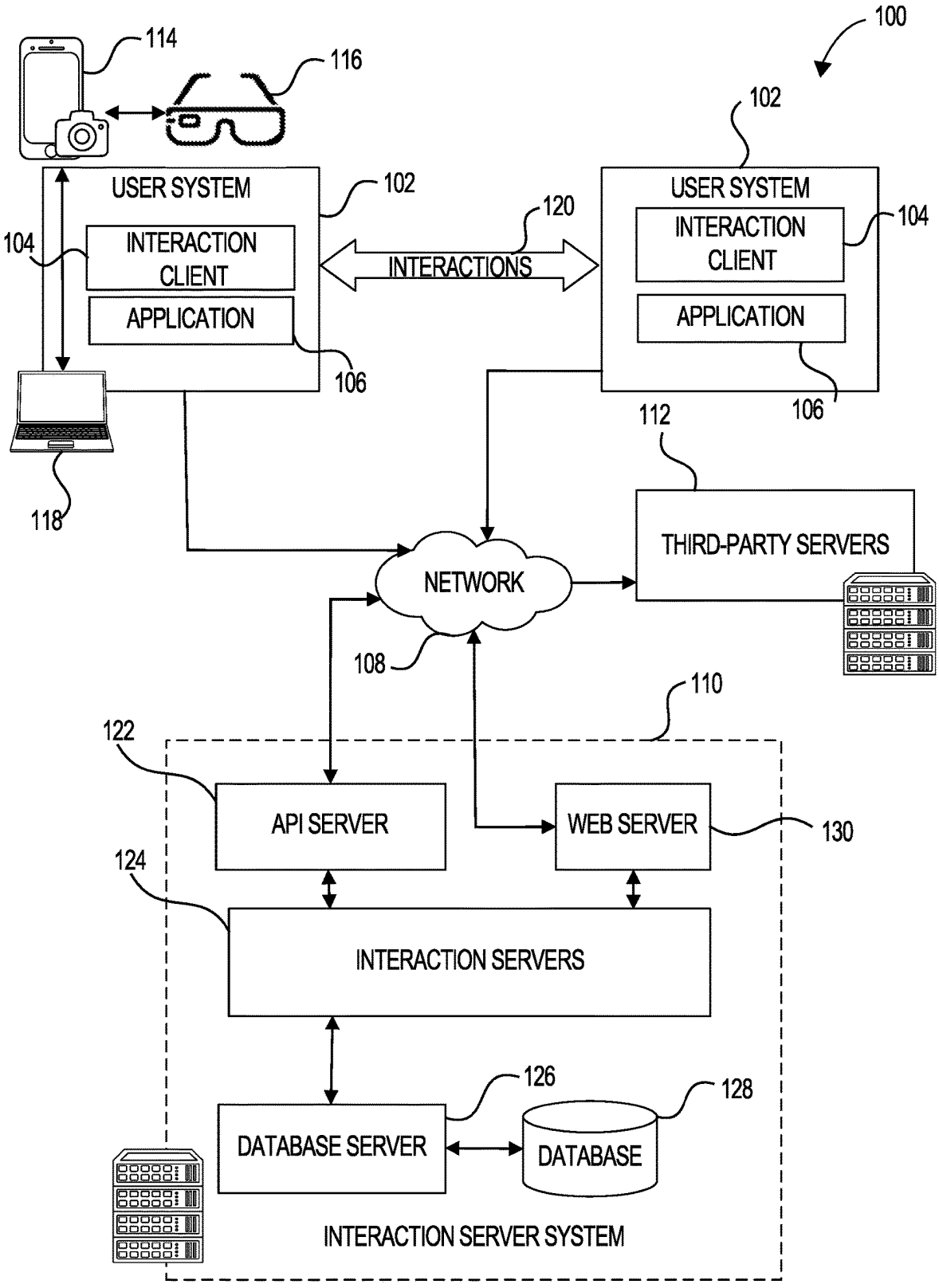

100
114
116
102
USER SYSTEM
102
104
INTERACTION CLIENT
APPLICATION
106
120
INTERACTIONS
USER SYSTEM
INTERACTION CLIENT
104
APPLICATION
106
118
112
THIRD-PARTY SERVERS
NETWORK
108
110
122
API SERVER
WEB SERVER
130
124
INTERACTION SERVERS
126
128
DATABASE SERVER
DATABASE
INTERACTION SERVER SYSTEM

FIG. 1

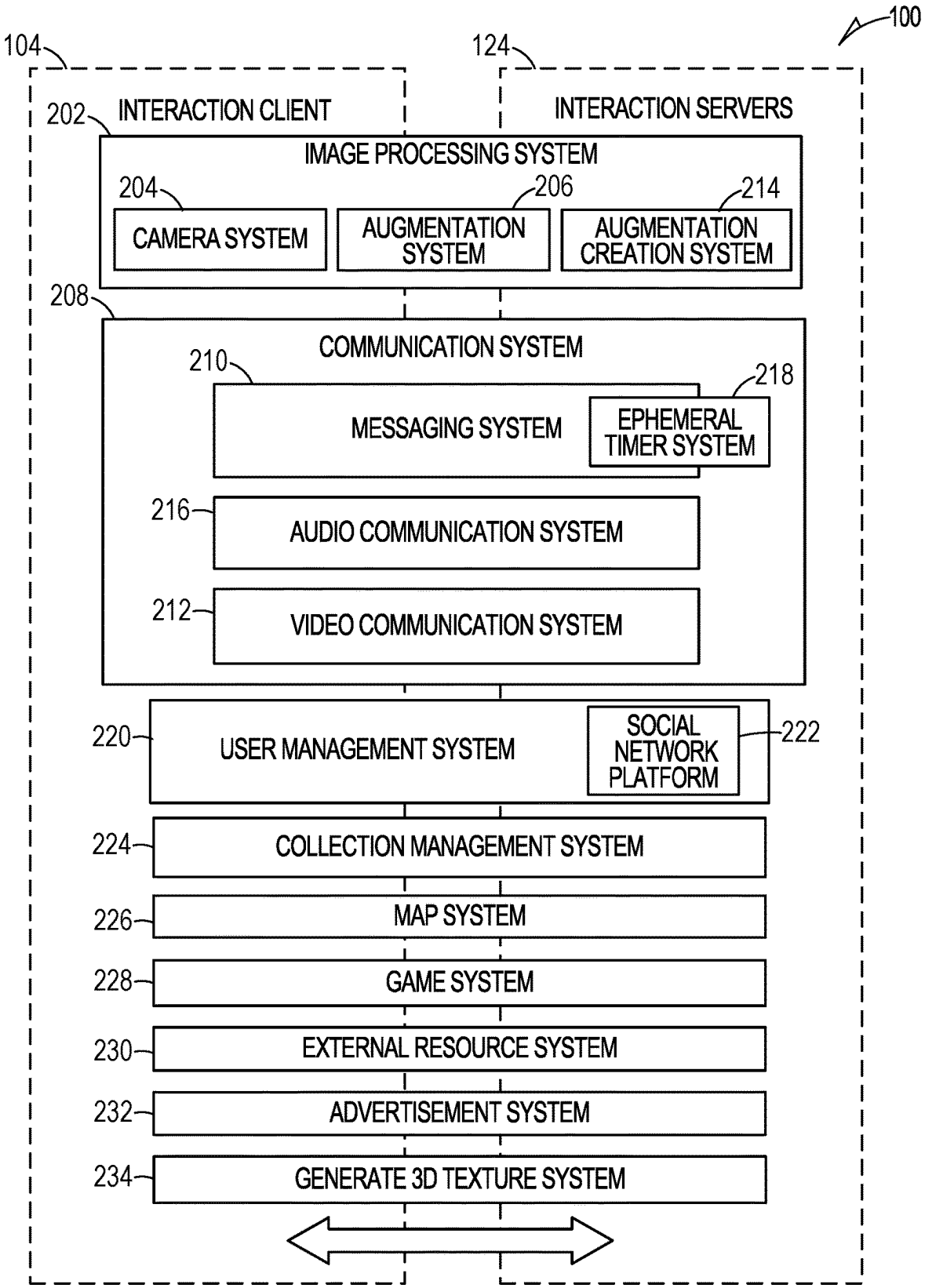

100
104
124
202
INTERACTION CLIENT
INTERACTION SERVERS
IMAGE PROCESSING SYSTEM
204
206
214
CAMERA SYSTEM
AUGMENTATION SYSTEM
AUGMENTATION CREATION SYSTEM
208
COMMUNICATION SYSTEM
210
218
MESSAGING SYSTEM
EPHEMERAL TIMER SYSTEM
216
AUDIO COMMUNICATION SYSTEM
212
VIDEO COMMUNICATION SYSTEM
220
USER MANAGEMENT SYSTEM
SOCIAL NETWORK PLATFORM
222
224
COLLECTION MANAGEMENT SYSTEM
226
MAP SYSTEM
228
GAME SYSTEM
230
EXTERNAL RESOURCE SYSTEM
232
ADVERTISEMENT SYSTEM
234
GENERATE 3D TEXTURE SYSTEM

FIG. 2

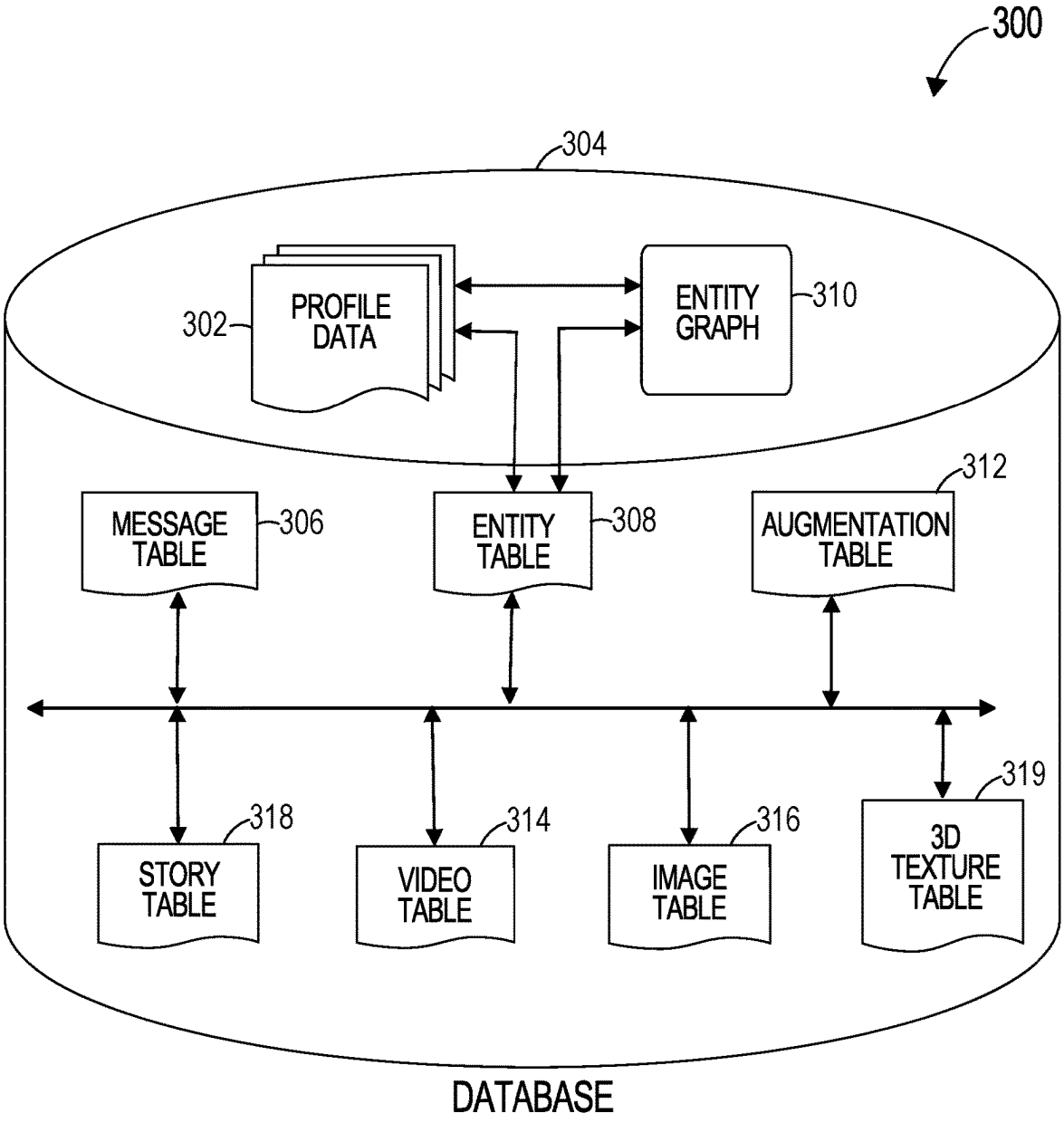

300
304
PROFILE DATA
302
ENTITY GRAPH
310
312
MESSAGE TABLE
306
ENTITY TABLE
308
AUGMENTATION TABLE
319
318
314
316
STORY TABLE
VIDEO TABLE
IMAGE TABLE
3D TEXTURE TABLE
DATABASE

FIG. 3

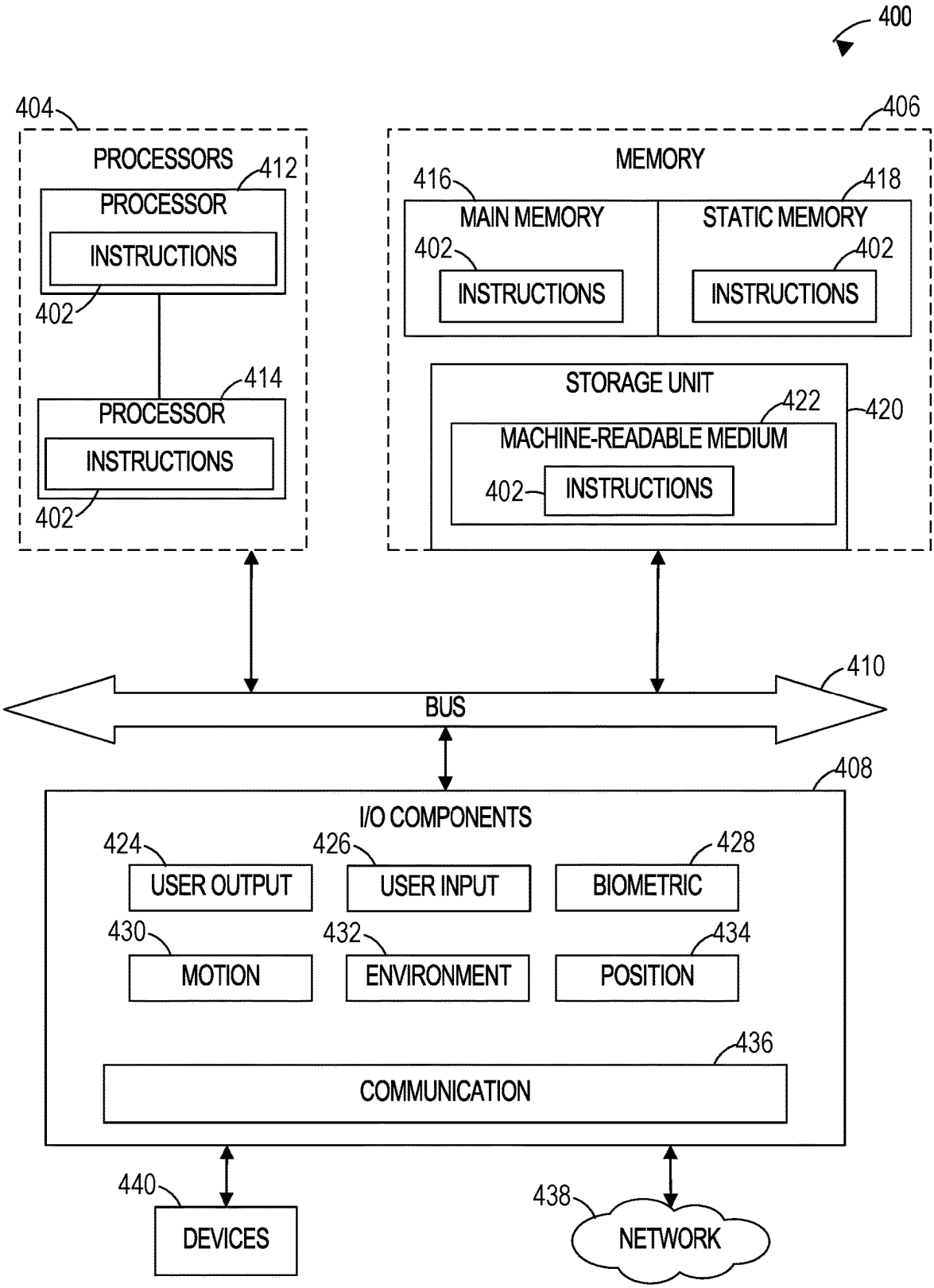

400
404
PROCESSORS
412
PROCESSOR
INSTRUCTIONS
402
414
PROCESSOR
INSTRUCTIONS
402
406
MEMORY
416
MAIN MEMORY
402
INSTRUCTIONS
418
STATIC MEMORY
402
INSTRUCTIONS
STORAGE UNIT
422
MACHINE-READABLE MEDIUM
402
INSTRUCTIONS
420
410
BUS
408
I/O COMPONENTS
424
USER OUTPUT
426
USER INPUT
428
BIOMETRIC
430
MOTION
432
ENVIRONMENT
434
POSITION
436
COMMUNICATION
440
DEVICES
438
NETWORK

FIG. 4

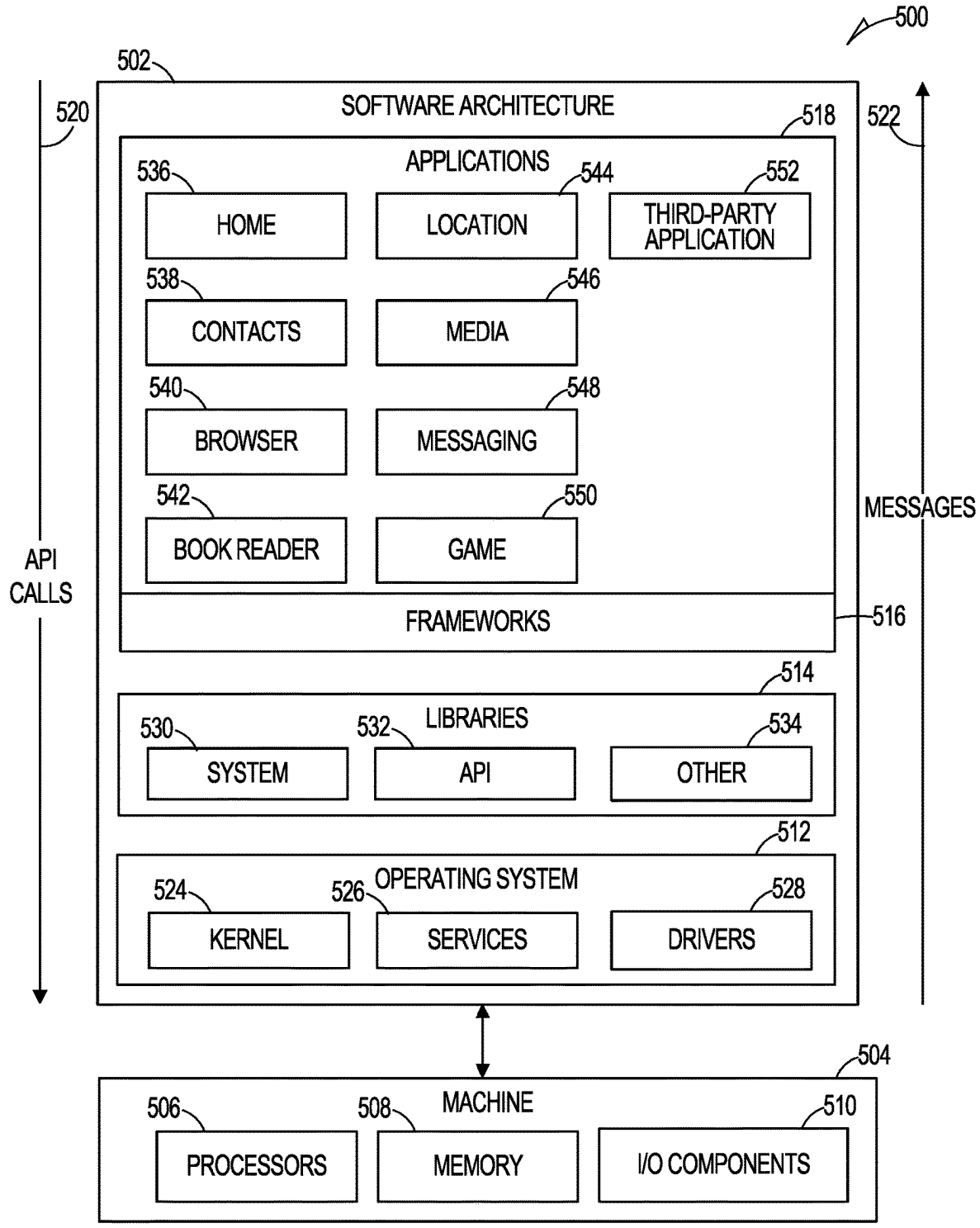

500
502
SOFTWARE ARCHITECTURE
520
522
518
APPLICATIONS
536
544
552
HOME
LOCATION
THIRD-PARTY APPLICATION
538
546
CONTACTS
MEDIA
540
548
BROWSER
MESSAGING
542
550
BOOK READER
GAME
API CALLS
MESSAGES
FRAMEWORKS
516
514
LIBRARIES
530
532
534
SYSTEM
API
OTHER
512
OPERATING SYSTEM
524
526
528
KERNEL
SERVICES
DRIVERS
504
MACHINE
506
508
510
PROCESSORS
MEMORY
I/O COMPONENTS

FIG. 5

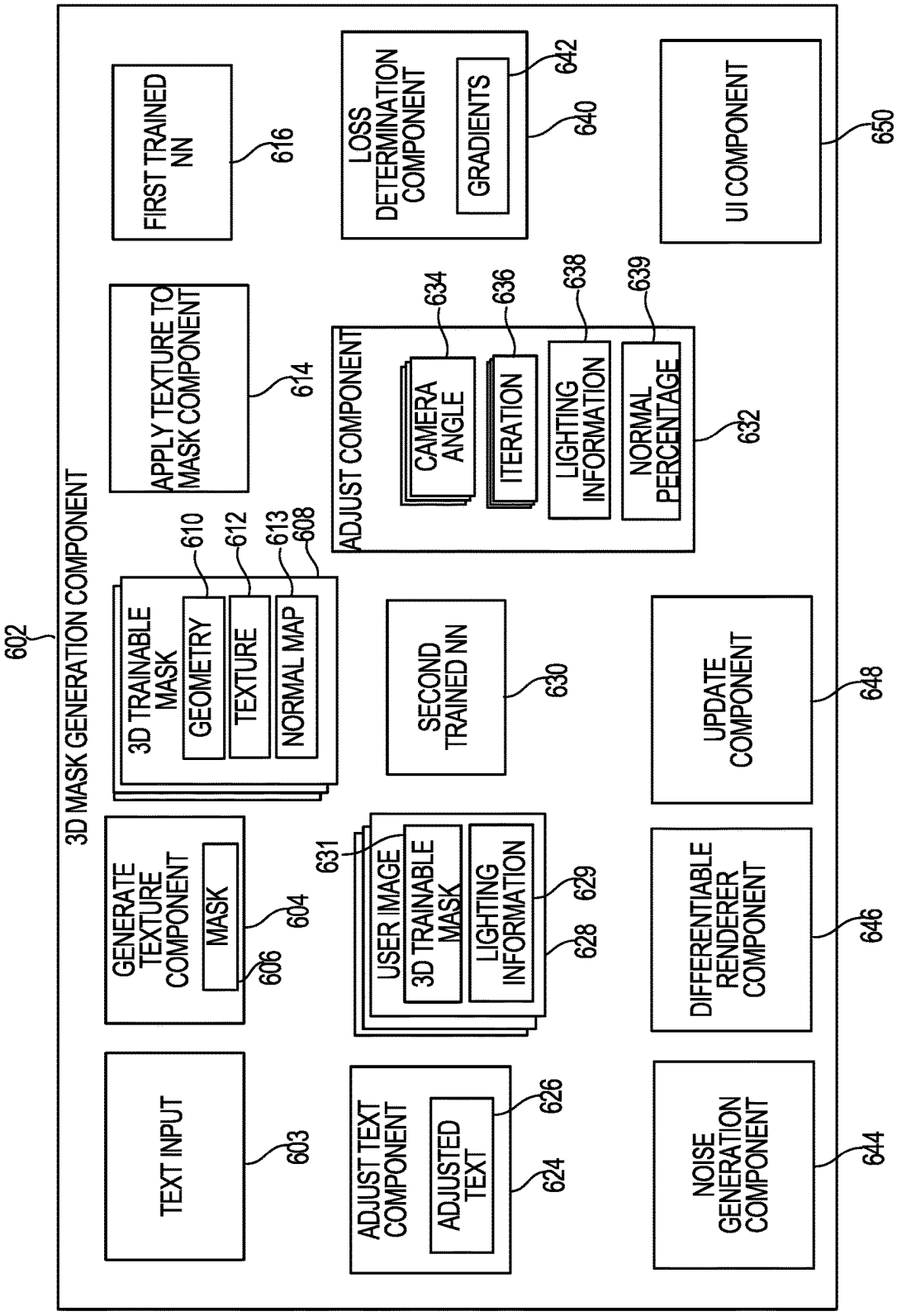

602
3D MASK GENERATION COMPONENT
TEXT INPUT
603
GENERATE TEXTURE COMPONENT
MASK
606
604
3D TRAINABLE MASK
GEOMETRY
TEXTURE
NORMAL MAP
610
612
613
608
APPLY TEXTURE TO MASK COMPONENT
614
FIRST TRAINED NN
616
ADJUST TEXT COMPONENT
ADJUSTED TEXT
626
624
USER IMAGE
3D TRAINABLE MASK
LIGHTING INFORMATION
631
629
628
SECOND TRAINED NN
630
ADJUST COMPONENT
CAMERA ANGLE
ITERATION
LIGHTING INFORMATION
NORMAL PERCENTAGE
634
636
638
639
632
LOSS DETERMINATION COMPONENT
GRADIENTS
642
640
DIFFERENTIABLE RENDERER COMPONENT
646
UPDATE COMPONENT
648
NOISE GENERATION COMPONENT
644
UI COMPONENT
650

FIG. 6

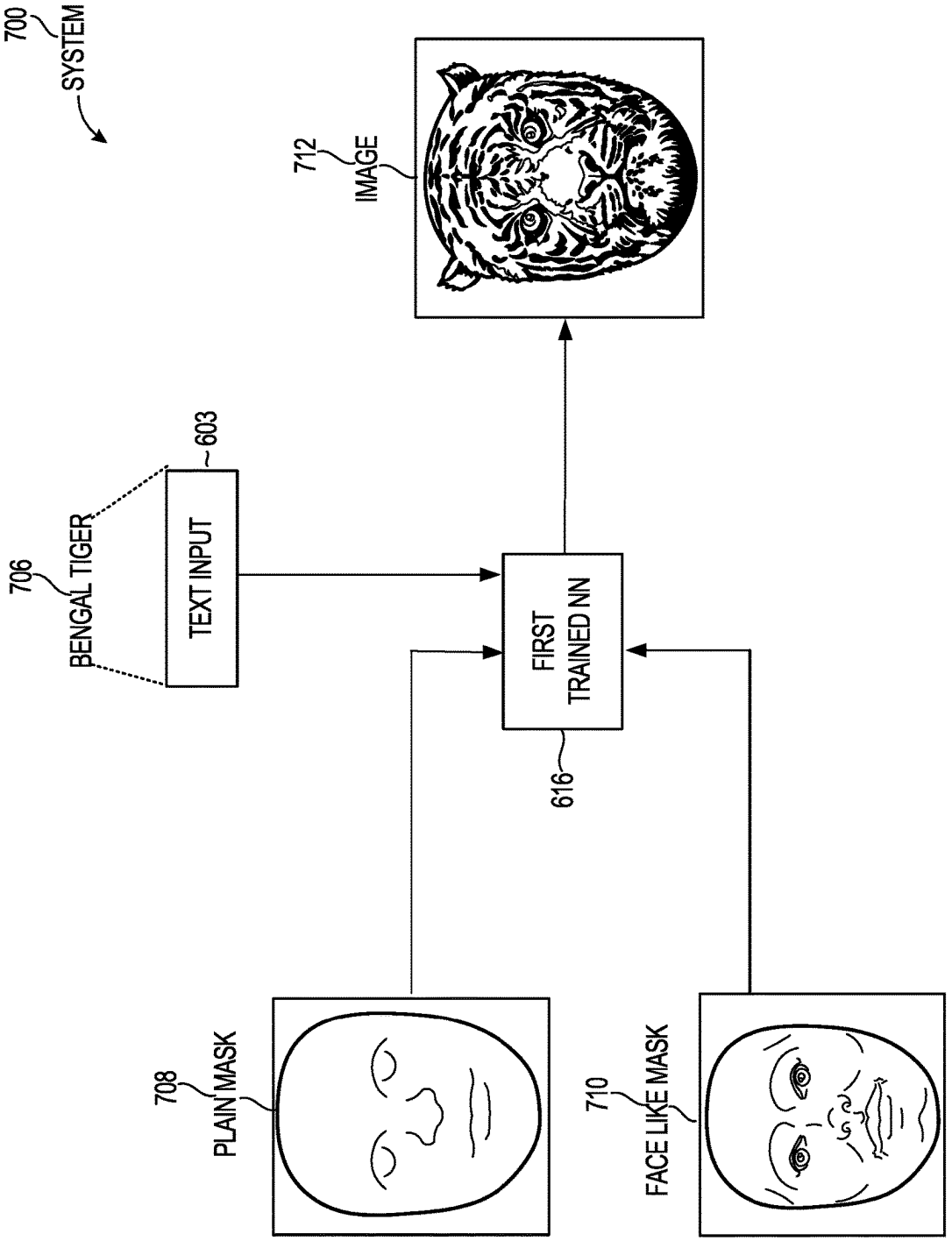

700
SYSTEM
712
IMAGE
603
TEXT INPUT
706
BENGAL TIGER
FIRST TRAINED NN
616
708
PLAIN MASK
710
FACE LIKE MASK

FIG. 7

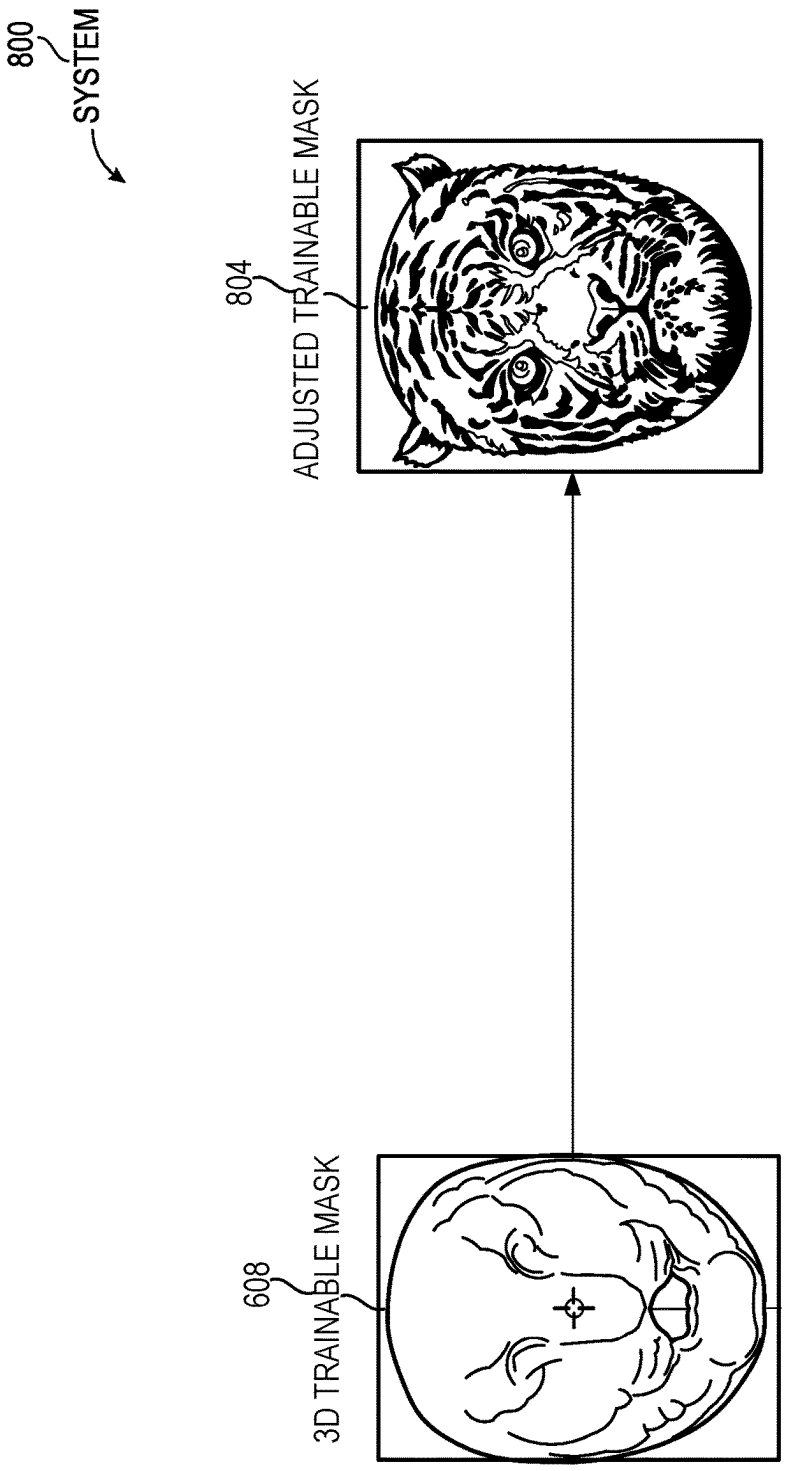

800
SYSTEM
804
ADJUSTED TRAINABLE MASK
608
3D TRAINABLE MASK

FIG. 8

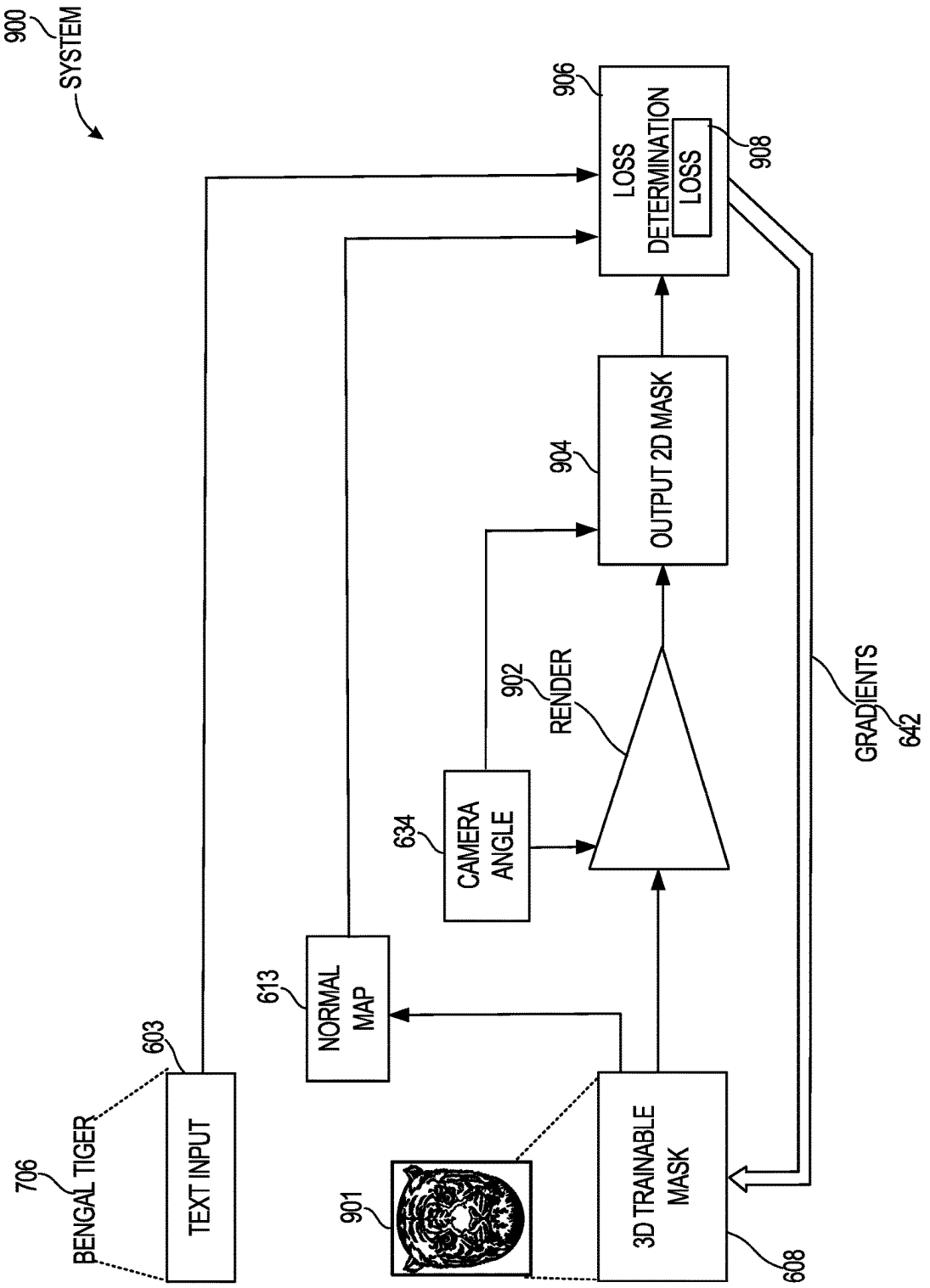

900
SYSTEM
906
LOSS DETERMINATION
LOSS
908
OUTPUT 2D MASK
904
RENDER
902
CAMERA ANGLE
634
NORMAL MAP
613
TEXT INPUT
603
BENGAL TIGER
706
3D TRAINABLE MASK
608
901
GRADIENTS
642

FIG. 9

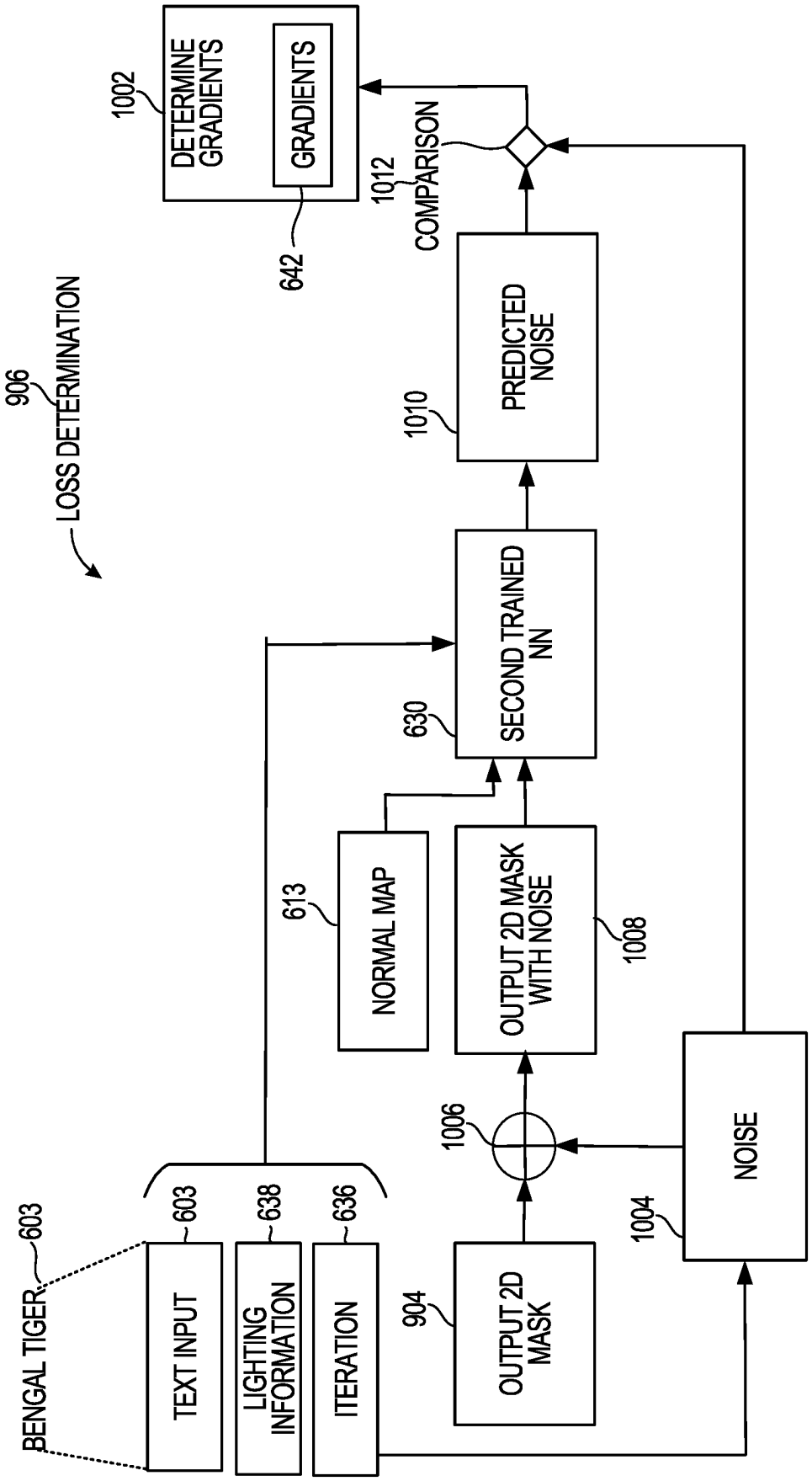

906
LOSS DETERMINATION
1002
DETERMINE GRADIENTS
GRADIENTS
642
1012
COMPARISON
1010
PREDICTED NOISE
630
SECOND TRAINED NN
613
NORMAL MAP
OUTPUT 2D MASK WITH NOISE
1008
1006
NOISE
1004
904
OUTPUT 2D MASK
603
638
636
TEXT INPUT
LIGHTING INFORMATION
ITERATION
603
BENGAL TIGER

FIG. 10

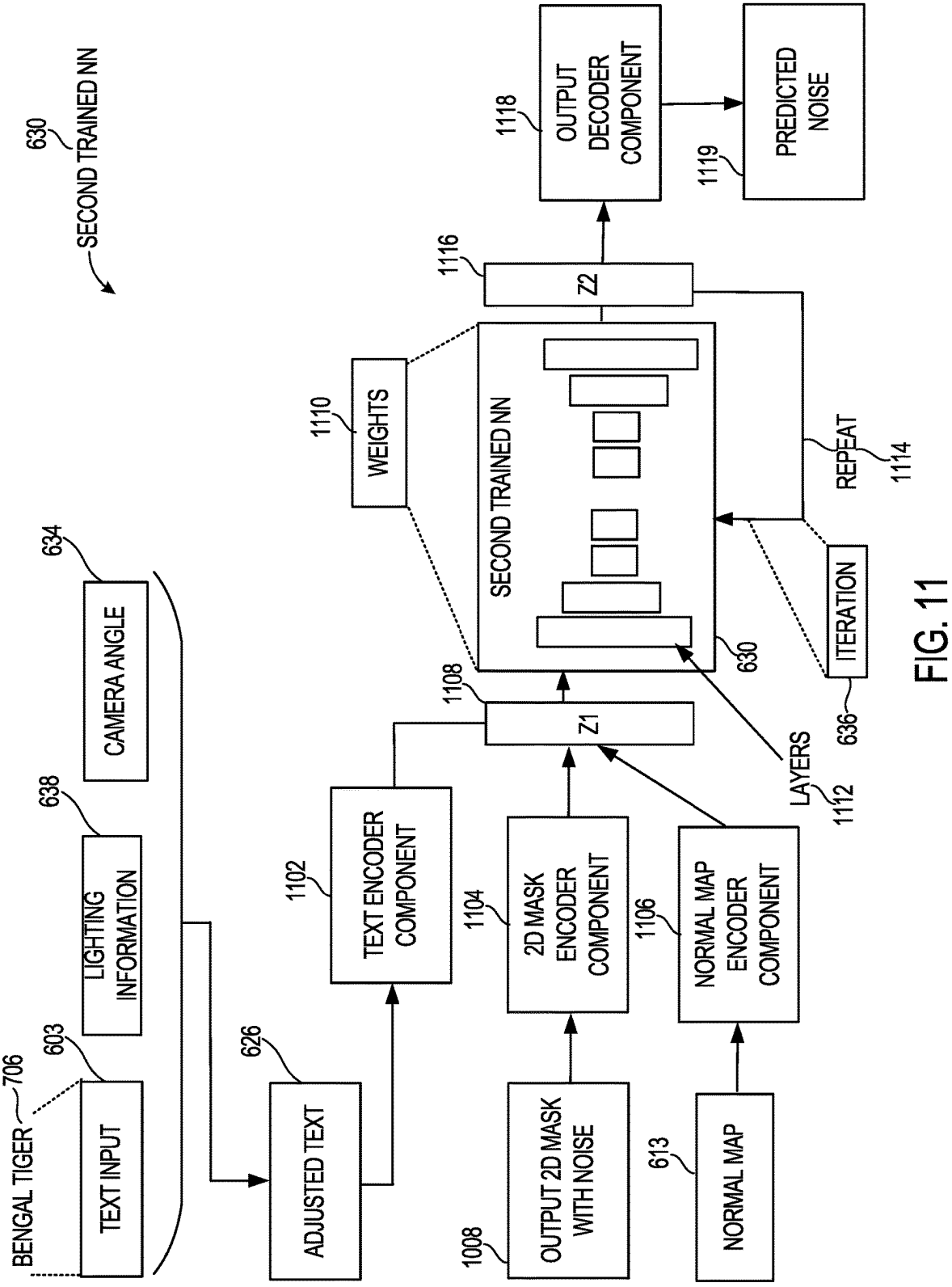

630
SECOND TRAINED NN
1118
OUTPUT DECODER COMPONENT
1119
PREDICTED NOISE
1116
Z2
1110
WEIGHTS
SECOND TRAINED NN
REPEAT
1114
634
CAMERA ANGLE
638
LIGHTING INFORMATION
603
706
BENGAL TIGER
TEXT INPUT
626
ADJUSTED TEXT
1102
TEXT ENCODER COMPONENT
1104
2D MASK ENCODER COMPONENT
1106
NORMAL MAP ENCODER COMPONENT
1108
Z1
LAYERS
1112
ITERATION
636
1008
OUTPUT 2D MASK WITH NOISE
613
NORMAL MAP

FIG. 11

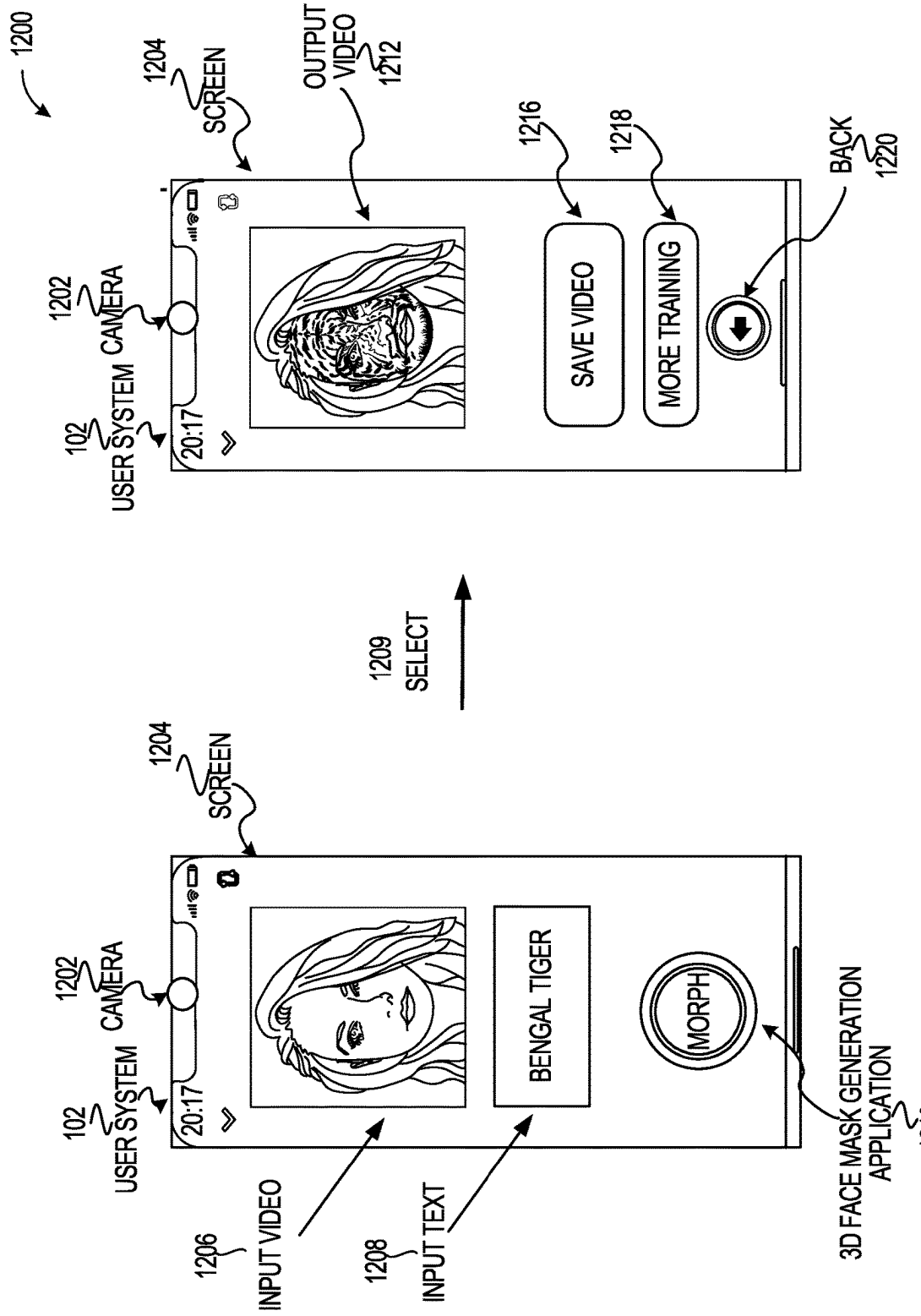

1200
1204
SCREEN
OUTPUT VIDEO
1212
1216
1218
BACK
1220
1202
CAMERA
102
USER SYSTEM
20:17
SAVE VIDEO
MORE TRAINING
1209
SELECT
1204
SCREEN
1202
CAMERA
102
USER SYSTEM
20:17
BENGAL TIGER
MORPH
1206
INPUT VIDEO
1208
INPUT TEXT
3D FACE MASK GENERATION APPLICATION
1210

FIG. 12

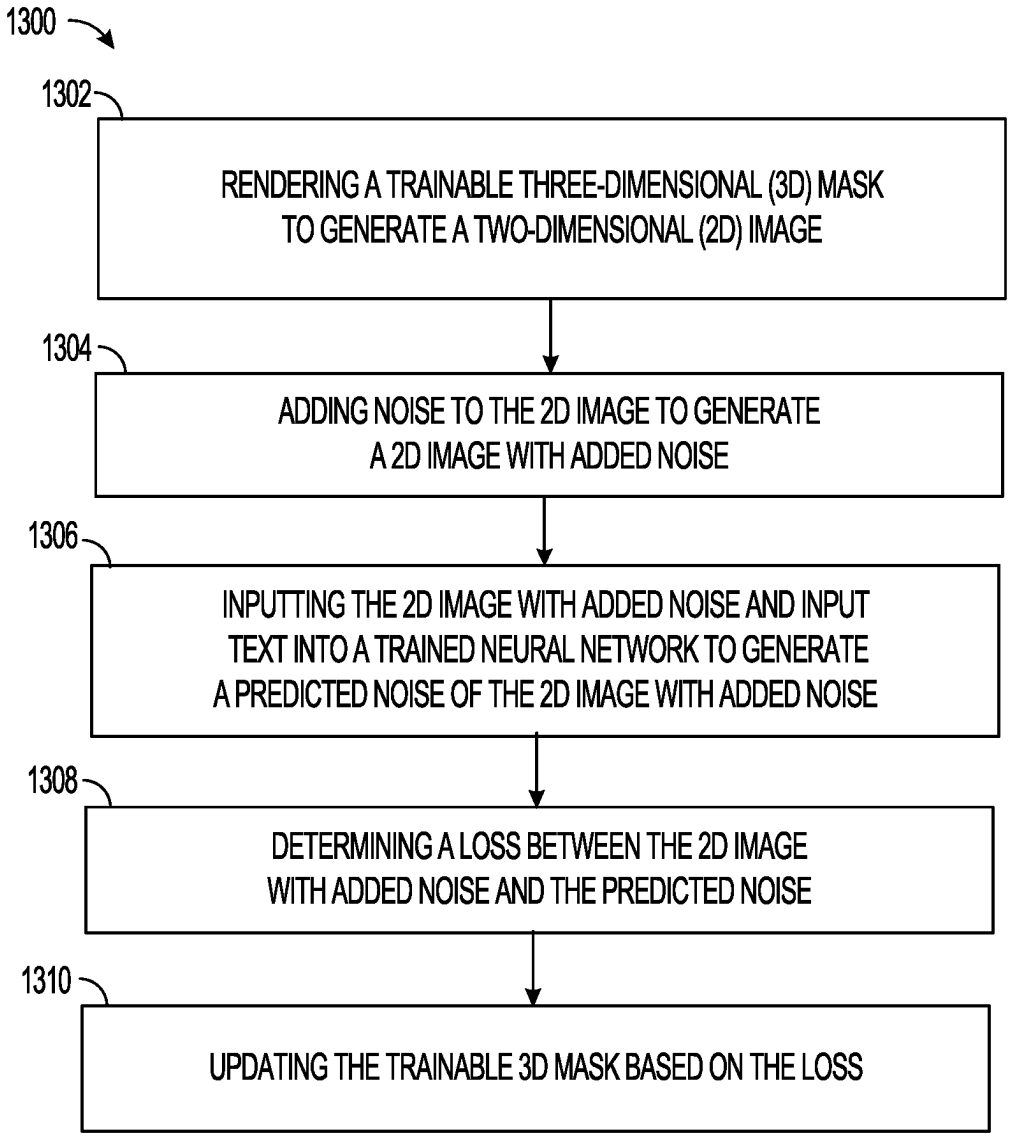

1300
1302
RENDERING A TRAINABLE THREE-DIMENSIONAL (3D) MASK TO GENERATE A TWO-DIMENSIONAL (2D) IMAGE
1304
ADDING NOISE TO THE 2D IMAGE TO GENERATE A 2D IMAGE WITH ADDED NOISE
1306
INPUTTING THE 2D IMAGE WITH ADDED NOISE AND INPUT TEXT INTO A TRAINED NEURAL NETWORK TO GENERATE A PREDICTED NOISE OF THE 2D IMAGE WITH ADDED NOISE
1308
DETERMINING A LOSS BETWEEN THE 2D IMAGE WITH ADDED NOISE AND THE PREDICTED NOISE
1310
UPDATING THE TRAINABLE 3D MASK BASED ON THE LOSS

FIG. 13

# 3D MASK GENERATION

## TECHNICAL FIELD

**[0001]** Examples of the present disclosure relate generally to generative artificial intelligence (AI) applications or generative neural networks (NN), where a three-dimensional (3D) mask is generated or iteratively improved based on input text describing the 3D mask.

## BACKGROUND

**[0002]** AI applications have become ubiquitous. Additionally, users increasingly want more and more functionality from the AI applications and for the functionality to be easier to use. But often it is difficult to add the functionality.

## BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

**[0003]** In the drawings, which are not necessarily drawn to scale, like numerals may describe similar components in different views. To easily identify the discussion of any particular element or act, the most significant digit or digits in a reference number refer to the figure number in which that element is first introduced. Some non-limiting examples are illustrated in the figures of the accompanying drawings in which:

**[0004]** FIG. **1** is a diagrammatic representation of a networked environment in which the present disclosure may be deployed, according to some examples.

**[0005]** FIG. **2** is a diagrammatic representation of a messaging system, according to some examples, that has both client-side and server-side functionality.

**[0006]** FIG. **3** is a diagrammatic representation of a data structure as maintained in a database, according to some examples.

**[0007]** FIG. **4** is a diagrammatic representation of a machine in the form of a computer system within which a set of instructions may be executed to cause the machine to perform any one or more of the methodologies discussed herein, according to some examples.

**[0008]** FIG. **5** is a block diagram showing a software architecture within which examples may be implemented.

**[0009]** FIG. **6** illustrates a 3D mask generation component, in accordance with some examples.

**[0010]** FIG. **7** illustrates a system to generate an image, in accordance with some examples.

**[0011]** FIG. **8** illustrates a system for 3D mask generation, in accordance with some examples.

**[0012]** FIG. **9** illustrates a system for 3D mask generation, in accordance with some examples.

**[0013]** FIG. **10** illustrates loss determination, in accordance with some examples.

**[0014]** FIG. **11** illustrates the second trained NN, in accordance with some examples.

**[0015]** FIG. **12** illustrates an example 3D mask generation application, in accordance with some examples.

**[0016]** FIG. **13** illustrates a method for 3D mask generation, in accordance with some examples.

## DETAILED DESCRIPTION

**[0017]** The description that follows includes systems, methods, techniques, instruction sequences, and computing machine program products that embody illustrative examples of the disclosure. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide an understanding of various examples of the inventive subject matter. It will be evident, however, to those skilled in the art, that examples of the inventive subject matter may be practiced without these specific details. In general, well-known instruction instances, protocols, structures, and techniques are not necessarily shown in detail.

**[0018]** Producing 3D masks, such as 3D face masks to place on the face of a user taking a video is often difficult and requires familiarity with applications that can be difficult to learn. For example, for a user to produce a 3D mask of a Bengal tiger, the user may have to use sophisticated 3D modeling software. However, the demand for 3D masks continues to grow as users of social network platforms continue to demand more sophisticated applications. Additionally, the users often would like the 3D mask to be generated from a textual description of the 3D mask. Moreover, it may be extraordinarily difficult to create 3D masks to be available on a social network platform **222** because the number of possible 3D masks is prohibitively large.

**[0019]** For example, a user may be using a video application such as a video call with another user and would like the image of their head to include a 3D mask of a Bengal tiger. The goal is to generate a 3D mask so that when the 3D mask is rendered from an arbitrary angle that the result is a 3D mask that fits the face of the user.

**[0020]** A technical challenge is that the computational requirements may be prohibitive for training a generative AI system to generate 3D masks from a textural description. The technical challenge is addressed by using a generative NN trained to generate 2D images from input text and updating an initial 3D mask based on the output of the generative NN compared with a 2D image of the 3D mask rendered with the initial 3D mask.

**[0021]** More particularly, but not by way of limitations, examples of the present disclosure relate to a method that begins with accessing input text describing a 3D mask. Then, a first NN is used to generate a 2D image of the input text, which is used to generate texture for an initial 3D mask.

**[0022]** The method continues where a differentiable renderer takes as input the initial 3D mask to generate an output 2D image. Gaussian noise is then added to the output 2D image to generate an output 2D image with noise. The output 2D image with noise and the input text are input into the trained NN, which generates a predicted noise. In some examples, the system determines a normal map and the normal map is also input into the trained NN. The trained NN is a stable diffusion model or a diffusion model for generating 2D images from a textual description. The trained NN model generates a predicted noise.

**[0023]** The predicted noise is an indication of how far the output 2D image with noise is, within the latent space of the trained NN, from being a 2D image in accordance with the input text. Gradients are then determined based on a comparison of the predicted noise with the output 2D image with noise. The gradients may also be determined based on the predicted noise being subtracted from the 2D image with noise.

**[0024]** The gradients are propagated or backpropagated through the output 2D image, the differentiable renderer, the normal map, and the trainable 3D mask. The trainable 3D mask is improved by the backpropagation. This method is

repeated a number of times or until the difference between the predicted loss and the added noise is below a threshold or transgresses a threshold.

**[0025]** Additionally, in some examples, the method is repeated for different camera angles used by the renderer and the input text is modified to agree with the camera angle. For example, the camera angle may be looking down at the Bengal tiger, so the input text is modified to "looking down on a Bengal tiger." In some examples, different lighting information may be used by the renderer. For example, a single bright light source at a position of x=0, y=0, and z=10 meters from the 3D mask. The input text may be modified to reflect the light source such as "looking down on a Bengal tiger with a light source shining down on the Bengal tiger." Many different camera angles and light sources may be used so that the 3D mask may be used from any angle of the 3D mask. Additionally, for an application such as adding the 3D mask to a face of a person making a video, the application may determine the lighting of the face of the person and change the 3D mask to agree with the lighting of the face of the person in the video to give a more realistic appearance.

**[0026]** In some examples, the 3D mask may be generated quickly enough to enable a user of a video application to request any 3D mask based on text input if the NN can generate a 2D image for the text input.

## Networked Computing Environment

**[0027]** FIG. **1** is a block diagram showing an example interaction system **100** for facilitating interactions (e.g., exchanging text messages, conducting text audio and video calls, or playing games) over a network. The interaction system **100** includes multiple client systems, each of which hosts multiple applications, including an interaction client **104** and other applications **106**. Each interaction client **104** is communicatively coupled, via one or more communication networks including a network **108** (e.g., the Internet), to other instances of the interaction client **104** (e.g., hosted on respective other user systems), an interaction platform **110** and third-party servers **112**). An interaction client **104** can also communicate with locally hosted applications **106** using Applications Program Interfaces (APIs).

**[0028]** Each user system **102** may include multiple user devices, such as a computing device **114**, head-wearable apparatus **116**, and a computer client device **118** that are communicatively connected to exchange data and messages. In some examples, the head-wearable apparatus **116** device is termed augmented reality (AR), mixed reality (MR), virtual reality (VR), and/or extended reality (XR) glasses.

**[0029]** An interaction client **104** interacts with other interaction clients **104** and with the interaction platform **110** via the network **108**. The data exchanged between the interaction clients **104** (e.g., interactions **120**) and between the interaction clients **104** and the interaction platform **110** includes functions (e.g., commands to invoke functions) and payload data (e.g., text, audio, video, or other multimedia data).

**[0030]** The interaction platform **110** provides server-side functionality via the network **108** to the interaction clients **104**. While certain functions of the interaction system **100** are described herein as being performed by either an interaction client **104** or by the interaction platform **110**, the location of certain functionality either within the interaction client **104** or the interaction platform **110** may be a design choice. For example, it may be technically preferable to initially deploy particular technology and functionality within the interaction platform **110** but to later migrate this technology and functionality to the interaction client **104** where a user system **102** has sufficient processing capacity.

**[0031]** The interaction platform **110** supports various services and operations that are provided to the interaction clients **104**. Such operations include transmitting data to, receiving data from, and processing data generated by the interaction clients **104**. This data may include message content, client device information, geolocation information, media augmentation and overlays, message content persistence conditions, social network information, and live event information. Data exchanges within the interaction system **100** are invoked and controlled through functions available via user interfaces (UIs) of the interaction clients **104**.

**[0032]** Turning now specifically to the interaction platform **110**, an Application Program Interface (API) server **122** is coupled to and provides programmatic interfaces to interaction servers **124**, making the functions of the interaction servers **124** accessible to interaction clients **104**, other applications **106** and third-party server **112**. The interaction servers **124** are communicatively coupled to a database server **126**, facilitating access to a database **128** that stores data associated with interactions processed by the interaction servers **124**. Similarly, a web server **130** is coupled to the interaction servers **124** and provides web-based interfaces to the interaction servers **124**. To this end, the web server **130** processes incoming network requests over the Hypertext Transfer Protocol (HTTP) and several other related protocols.

**[0033]** The Application Program Interface (API) server **122** receives and transmits interaction data (e.g., commands and message payloads) between the interaction servers **124** and the client systems (and, for example, interaction clients **104** and other application **106**) and the third-party server **112**. Specifically, the Application Program Interface (API) server **122** provides a set of interfaces (e.g., routines and protocols) that can be called or queried by the interaction client **104** and other applications **106** to invoke functionality of the interaction servers **124**. The Application Program Interface (API) server **122** exposes various functions supported by the interaction servers **124**, including account registration; login functionality; the sending of interaction data, via the interaction servers **124**, from a particular interaction client **104** to another interaction client **104**; the communication of media files (e.g., images or video) from an interaction client **104** to the interaction servers **124**; the settings of a collection of media data (e.g., a story); the retrieval of a list of friends of a user of a user system **102**; the retrieval of messages and content; the addition and deletion of entities (e.g., friends) to an entity graph (e.g., a social graph); the location of friends within a social graph; and opening an application event (e.g., relating to the interaction client **104**). The interaction servers **124** host multiple systems and subsystems, described below with reference to FIG. **2**.

## Linked Applications

**[0034]** Returning to the interaction client **104**, features and functions of an external resource (e.g., a linked application **106** or applet) are made available to a user via an interface of the interaction client **104**. In this context, "external" refers to the fact that the application **106** or applet is external to the interaction client **104**. The external resource is often pro-

vided by a third party but may also be provided by the creator or provider of the interaction client **104**. The interaction client **104** receives a user selection of an option to launch or access features of such an external resource. The external resource may be the application **106** installed on the user system **102** (e.g., a "native app"), or a small-scale version of the application (e.g., an "applet") that is hosted on the user system **102** or remote of the user system **102** (e.g., on third-party servers **112**). The small-scale version of the application includes a subset of features and functions of the application (e.g., the full-scale, native version of the application) and is implemented using a markup-language document. In some examples, the small-scale version of the application (e.g., an "applet") is a web-based, markup-language version of the application and is embedded in the interaction client **104**. In addition to using markup-language documents (e.g., a .*ml file), an applet may incorporate a scripting language (e.g., a .*js file or a .json file) and a style sheet (e.g., a .*ss file).

**[0035]** In response to receiving a user selection of the option to launch or access features of the external resource, the interaction client **104** determines whether the selected external resource is a web-based external resource or a locally-installed application **106**. In some cases, applications **106** that are locally installed on the user system **102** can be launched independently of and separately from the interaction client **104**, such as by selecting an icon corresponding to the application **106** on a home screen of the user system **102**. Small-scale versions of such applications can be launched or accessed via the interaction client **104** and, in some examples, no or limited portions of the small-scale application can be accessed outside of the interaction client **104**. The small-scale application can be launched by the interaction client **104** receiving, from a third-party server **112** for example, a markup-language document associated with the small-scale application and processing such a document.

**[0036]** In response to determining that the external resource is a locally-installed application **106**, the interaction client **104** instructs the user system **102** to launch the external resource by executing locally-stored code corresponding to the external resource. In response to determining that the external resource is a web-based resource, the interaction client **104** communicates with the third-party servers **112** (for example) to obtain a markup-language document corresponding to the selected external resource. The interaction client **104** then processes the obtained markup-language document to present the web-based external resource within a user interface of the interaction client **104**.

**[0037]** The interaction client **104** can notify a user of the user system **102**, or other users related to such a user (e.g., "friends"), of activity taking place in one or more external resources. For example, the interaction client **104** can provide participants in a conversation (e.g., a chat session) in the interaction client **104** with notifications relating to the current or recent use of an external resource by one or more members of a group of users. One or more users can be invited to join in an active external resource or to launch a recently-used but currently inactive (in the group of friends) external resource. The external resource can provide participants in a conversation, each using respective interaction clients **104**, with the ability to share an item, status, state, or location in an external resource in a chat session with one or more members of a group of users. The shared item may be an interactive chat card with which members of the chat can interact, for example, to launch the corresponding external resource, view specific information within the external resource, or take the member of the chat to a specific location or state within the external resource. Within a given external resource, response messages can be sent to users on the interaction client **104**. The external resource can selectively include different media items in the responses, based on a current context of the external resource.

**[0038]** The interaction client **104** can present a list of the available external resources (e.g., applications **106** or applets) to a user to launch or access a given external resource. This list can be presented in a context-sensitive menu. For example, the icons representing different ones of the application **106** (or applets) can vary based on how the menu is launched by the user (e.g., from a conversation interface or from a non-conversation interface).

System Architecture

**[0039]** FIG. **2** is a block diagram illustrating further details regarding the interaction system **100**, according to some examples. Specifically, the interaction system **100** is shown to comprise the interaction client **104** and the interaction servers **124**. The interaction system **100** embodies multiple subsystems, which are supported on the client-side by the interaction client **104** and on the server-side by the interaction servers **124**. Example subsystems are discussed below.

**[0040]** An image processing system **202** provides various functions that enable a user to capture and augment (e.g., annotate or otherwise modify or edit) media content associated with a message.

**[0041]** A camera system **204** includes control software (e.g., in a camera application) that interacts with and controls hardware camera hardware (e.g., directly or via operating system controls) of the user system **102** to modify and augment real-time images captured and displayed via the interaction client **104**.

**[0042]** The augmentation system **206** provides functions related to the generation and publishing of augmentations (e.g., media overlays) for images captured in real-time by cameras of the user system **102** or retrieved from memory of the user system **102**. For example, the augmentation system **206** operatively selects, presents, and displays media overlays (e.g., an image filter or an image lens) to the interaction client **104** for the augmentation of real-time images received via the camera system **204** or stored images retrieved from memory **406** of a user system **102**. These augmentations are selected by the augmentation system **206** and presented to a user of an interaction client **104**, based on a number of inputs and data, such as for example:

**[0043]** Geolocation of the user system **102**; and

**[0044]** Social network information of the user of the user system **102**.

**[0045]** An augmentation may include audio and visual content and visual effects. Examples of audio and visual content include pictures, texts, logos, animations, and sound effects. An example of a visual effect includes color overlaying. The audio and visual content or the visual effects can be applied to a media content item (e.g., a photo or video) at user system **102** for communication in a message, or applied to video content, such as a video content stream or feed transmitted from an interaction client **104**. As such, the image processing system **202** may interact with, and sup-

port, the various subsystems of the communication system **208**, such as the messaging system **210** and the video communication system **212**.

**[0046]** A media overlay may include text or image data that can be overlaid on top of a photograph taken by the user system **102** or a video stream produced by the user system **102**. In some examples, the media overlay may be a location overlay (e.g., Venice beach), a name of a live event, or a name of a merchant overlay (e.g., Beach Coffee House). In further examples, the image processing system **202** uses the geolocation of the user system **102** to identify a media overlay that includes the name of a merchant at the geolocation of the user system **102**. The media overlay may include other indicia associated with the merchant. The media overlays may be stored in the databases **128** and accessed through the database server **126**.

**[0047]** The image processing system **202** provides a user-based publication platform that enables users to select a geolocation on a map and upload content associated with the selected geolocation. The user may also specify circumstances under which a particular media overlay should be offered to other users. The image processing system **202** generates a media overlay that includes the uploaded content and associates the uploaded content with the selected geolocation.

**[0048]** The augmentation creation system **214** supports augmented reality developer platforms and includes an application for content creators (e.g., artists and developers) to create and publish augmentations (e.g., augmented reality experiences) of the interaction client **104**. The augmentation creation system **214** provides a library of built-in features and tools to content creators including, for example custom shaders, tracking technology, and templates.

**[0049]** In some examples, the augmentation creation system **214** provides a merchant-based publication platform that enables merchants to select a particular augmentation associated with a geolocation via a bidding process. For example, the augmentation creation system **214** associates a media overlay of the highest bidding merchant with a corresponding geolocation for a predefined amount of time.

**[0050]** A communication system **208** is responsible for enabling and processing multiple forms of communication and interaction within the interaction system **100** and includes a messaging system **210**, an audio communication system **216**, and a video communication system **212**. The messaging system **210** is responsible for enforcing the temporary or time-limited access to content by the interaction clients **104**. The messaging system **210** incorporates multiple timers (e.g., within an ephemeral timer system **218**) that, based on duration and display parameters associated with a message or collection of messages (e.g., a story), selectively enable access (e.g., for presentation and display) to messages and associated content via the interaction client **104**. Further details regarding the operation of the ephemeral timer system **218** are provided below. The audio communication system **216** enables and supports audio communications (e.g., real-time audio chat) between multiple interaction clients **104**. Similarly, the video communication system **212** enables and supports video communications (e.g., real-time video chat) between multiple interaction clients **104**.

**[0051]** A user management system **220** is operationally responsible for the management of user data and profiles, and includes an interaction platform that maintains information regarding relationships between users of the interaction system **100**.

**[0052]** A collection management system **224** is operationally responsible for managing sets or collections of media (e.g., collections of text, image video, and audio data). A collection of content (e.g., messages, including images, video, text, and audio) may be organized into an "event gallery" or an "event story." Such a collection may be made available for a specified time period, such as the duration of an event to which the content relates. For example, content relating to a music concert may be made available as a "story" for the duration of that music concert. The collection management system **224** may also be responsible for publishing an icon that provides notification of a particular collection to the user interface of the interaction client **104**. The collection management system **224** includes a curation function that allows a collection manager to manage and curate a particular collection of content. For example, the curation interface enables an event organizer to curate a collection of content relating to a specific event (e.g., delete inappropriate content or redundant messages). Additionally, the collection management system **224** employs machine vision (or image recognition technology) and content rules to curate a content collection automatically. In certain examples, compensation may be paid to a user to include user-generated content into a collection. In such cases, the collection management system **224** operates to automatically make payments to such users to use their content.

**[0053]** A map system **226** provides various geographic location functions and supports the presentation of map-based media content and messages by the interaction client **104**. For example, the map system **226** enables the display of user icons or avatars (e.g., stored in profile data **302**) on a map to indicate a current or past location of "friends" of a user, as well as media content (e.g., collections of messages including photographs and videos) generated by such friends, within the context of a map. For example, a message posted by a user to the interaction system **100** from a specific geographic location may be displayed within the context of a map at that particular location to "friends" of a specific user on a map interface of the interaction client **104**. A user can furthermore share his or her location and status information (e.g., using an appropriate status avatar) with other users of the interaction system **100** via the interaction client **104**, with this location and status information being similarly displayed within the context of a map interface of the interaction client **104** to selected users.

**[0054]** A game system **228** provides various gaming functions within the context of the interaction client **104**. The interaction client **104** provides a game interface providing a list of available games that can be launched by a user within the context of the interaction client **104** and played with other users of the interaction system **100**. The interaction system **100** further enables a particular user to invite other users to participate in the play of a specific game by issuing invitations to such other users from the interaction client **104**. The interaction client **104** also supports audio, video, and text messaging (e.g., chats) within the context of gameplay, provides a leaderboard for the games, and also supports the provision of in-game rewards (e.g., coins and items).

**[0055]** An external resource system **230** provides an interface for the interaction client **104** to communicate with remote servers (e.g., third-party servers **112**) to launch or

access external resources, i.e., applications or applets. Each third-party server **112** hosts, for example, a markup language (e.g., HTML5) based application or a small-scale version of an application (e.g., game, utility, payment, or ride-sharing application). The interaction client **104** may launch a web-based resource (e.g., application) by accessing the HTML5 file from the third-party servers **112** associated with the web-based resource. Applications hosted by third-party servers **112** are programmed in JavaScript leveraging a Software Development Kit (SDK) provided by the interaction servers **124**. The SDK includes Application Programming Interfaces (APIs) with functions that can be called or invoked by the web-based application. The interaction servers **124** host a JavaScript library that provides a given external resource access to specific user data of the interaction client **104**. HTML5 is an example of technology for programming games, but applications and resources programmed based on other technologies can be used.

**[0056]** To integrate the functions of the SDK into the web-based resource, the SDK is downloaded by the third-party server **112** from the interaction servers **124** or is otherwise received by the third-party server **112**. Once downloaded or received, the SDK is included as part of the application code of a web-based external resource. The code of the web-based resource can then call or invoke certain functions of the SDK to integrate features of the interaction client **104** into the web-based resource.

**[0057]** The SDK stored on the interaction platform **110** effectively provides the bridge between an external resource (e.g., applications **106** or applets) and the interaction client **104**. This gives the user a seamless experience of communicating with other users on the interaction client **104** while also preserving the look and feel of the interaction client **104**. To bridge communications between an external resource and an interaction client **104**, the SDK facilitates communication between third-party servers **112** and the interaction client **104**. A Web ViewJavaScriptBridge running on a user system **102** establishes two one-way communication channels between an external resource and the interaction client **104**. Messages are sent between the external resource and the interaction client **104** via these communication channels asynchronously. Each SDK function invocation is sent as a message and callback. Each SDK function is implemented by constructing a unique callback identifier and sending a message with that callback identifier.

**[0058]** By using the SDK, not all information from the interaction client **104** is shared with third-party servers **112**. The SDK limits which information is shared based on the needs of the external resource. Each third-party server **112** provides an HTML5 file corresponding to the web-based external resource to interaction servers **124**. The interaction servers **124** can add a visual representation (such as a box art or other graphic) of the web-based external resource in the interaction client **104**. Once the user selects the visual representation or instructs the interaction client **104** through a GUI of the interaction client **104** to access features of the web-based external resource, the interaction client **104** obtains the HTML5 file and instantiates the resources to access the features of the web-based external resource.

**[0059]** The interaction client **104** presents a graphical user interface (e.g., a landing page or title screen) for an external resource. During, before, or after presenting the landing page or title screen, the interaction client **104** determines whether the launched external resource has been previously authorized to access user data of the interaction client **104**. In response to determining that the launched external resource has been previously authorized to access user data of the interaction client **104**, the interaction client **104** presents another graphical user interface of the external resource that includes functions and features of the external resource. In response to determining that the launched external resource has not been previously authorized to access user data of the interaction client **104**, after a threshold period of time (e.g., 3 seconds) of displaying the landing page or title screen of the external resource, the interaction client **104** slides up (e.g., animates a menu as surfacing from a bottom of the screen to a middle or other portion of the screen) a menu for authorizing the external resource to access the user data. The menu identifies the type of user data that the external resource will be authorized to use. In response to receiving a user selection of an accept option, the interaction client **104** adds the external resource to a list of authorized external resources and allows the external resource to access user data from the interaction client **104**. The external resource is authorized by the interaction client **104** to access the user data under an OAuth **2** framework.

**[0060]** The interaction client **104** controls the type of user data that is shared with external resources based on the type of external resource being authorized. For example, external resources that include full-scale applications (e.g., an application **106**) are provided with access to a first type of user data (e.g., two-dimensional avatars of users with or without different avatar characteristics). As another example, external resources that include small-scale versions of applications (e.g., web-based versions of applications) are provided with access to a second type of user data (e.g., payment information, two-dimensional avatars of users, three-dimensional avatars of users, and avatars with various avatar characteristics). Avatar characteristics include different ways to customize a look and feel of an avatar, such as different poses, facial features, clothing, and so forth.

**[0061]** An advertisement system **232** operationally enables the purchasing of advertisements by third parties for presentation to end-users via the interaction clients **104** and also handles the delivery and presentation of these advertisements.

**[0062]** The generating 3D mask system **234** supports or is the same as the, referring to FIGS. **6** and **12**, 3D mask generation component **602** and/or the 3D face mask generation application **1210**. In some examples, the generate 3D mask system **234** performs one or more functions for the 3D mask generation component **602** and/or the 3D face mask generation application **1210**.

**[0063]** Additionally, the generate 3D mask system **234** can provide the 3D face mask generation application **1210** to user systems **102** and/or head-wearable apparatuses **116**. The 3D mask system **234** interacts with the social network platform **222** by providing services such as lists of available 3D masks, text inputs **603**, and by delivering data to user systems **102** and/or head-wearable apparatuses **116**.

Data Architecture

**[0064]** FIG. **3** is a schematic diagram illustrating data structures **300**, which may be stored in the database **304** of the interaction platform **110**, according to certain examples. While the content of the database **304** is shown to comprise

multiple tables, it will be appreciated that the data could be stored in other types of data structures (e.g., as an object-oriented database).

**[0065]** The database **304** includes message data stored within a message table **306**. This message data includes, for any particular message, at least message sender data, message recipient (or receiver) data, and a payload. Further details regarding information that may be included in a message and included within the message data stored in the message table **306**, are described below with reference to FIG. **3**.

**[0066]** An entity table **308** stores entity data, and is linked (e.g., referentially) to an entity graph **310** and profile data **302**. Entities for which records are maintained within the entity table **308** may include individuals, corporate entities, organizations, objects, places, events, and so forth. Regardless of entity type, any entity regarding which the interaction platform **110** stores data may be a recognized entity. Each entity is provided with a unique identifier, as well as an entity type identifier (not shown).

**[0067]** The entity graph **310** stores information regarding relationships and associations between entities. Such relationships may be social, professional (e.g., work at a common corporation or organization), interest-based, or activity-based, merely for example. Certain relationships between entities may be unidirectional, such as a subscription by an individual user to digital content of a commercial or publishing user (e.g., a newspaper or other digital media outlet, or a brand). Other relationships may be bidirectional, such as a "friend" relationship between individual users of the interaction system **100**.

**[0068]** Certain permissions and relationships may be attached to each relationship, and also to each direction of a relationship. For example, a bidirectional relationship (e.g., a friend relationship between individual users) may include authorization for the publication of digital content items between the individual users, but may impose certain restrictions or filters on the publication of such digital content items (e.g., based on content characteristics, location data or time of day data). Similarly, a subscription relationship between an individual user and a commercial user may impose different degrees of restrictions on the publication of digital content from the commercial user to the individual user, and may significantly restrict or block the publication of digital content from the individual user to the commercial user. A particular user, as an example of an entity, may record certain restrictions (e.g., by way of privacy settings) in a record for that entity within the entity table **308**. Such privacy settings may be applied to all types of relationships within the context of the interaction system **100**, or may selectively be applied to certain types of relationships.

**[0069]** The profile data **302** stores multiple types of profile data about a particular entity. The profile data **302** may be selectively used and presented to other users of the interaction system **100** based on privacy settings specified by a particular entity. Where the entity is an individual, the profile data **302** includes, for example, a user name, telephone number, address, settings (e.g., notification and privacy settings), as well as a user-selected avatar representation (or collection of such avatar representations). A particular user may then selectively include one or more of these avatar representations within the content of messages communicated via the interaction system **100**, and on map interfaces displayed by interaction clients **104** to other users. The collection of avatar representations may include "status avatars," which present a graphical representation of a status or activity that the user may select to communicate at a particular time.

**[0070]** Where the entity is a group, the profile data **302** for the group may similarly include one or more avatar representations associated with the group, in addition to the group name, members, and various settings (e.g., notifications) for the relevant group.

**[0071]** The database **304** also stores augmentation data, such as overlays or filters, in an augmentation table **312**. The augmentation data is associated with and applied to videos (for which data is stored in a video table **314**) and images (for which data is stored in an image table **316**).

**[0072]** Filters, in some examples, are overlays that are displayed as overlaid on an image or video during presentation to a recipient user. Filters may be of various types, including user-selected filters from a set of filters presented to a sending user by the interaction client **104** when the sending user is composing a message. Other types of filters include geolocation filters (also known as geo-filters), which may be presented to a sending user based on geographic location. For example, geolocation filters specific to a neighborhood or special location may be presented within a user interface by the interaction client **104**, based on geolocation information determined by a Global Positioning System (GPS) unit of the user system **102**.

**[0073]** Another type of filter is a data filter, which may be selectively presented to a sending user by the interaction client **104** based on other inputs or information gathered by the user system **102** during the message creation process. Examples of data filters include current temperature at a specific location, a current speed at which a sending user is traveling, battery life for a user system **102**, or the current time.

**[0074]** Other augmentation data that may be stored within the image table **316** includes augmented reality content items (e.g., corresponding to applying "lenses" or augmented reality experiences). An augmented reality content item may be a real-time special effect and sound that may be added to an image or a video.

**[0075]** A story table **318** stores data regarding collections of messages and associated image, video, or audio data, which are compiled into a collection (e.g., a story or a gallery). The creation of a particular collection may be initiated by a particular user (e.g., each user for which a record is maintained in the entity table **308**). A user may create a "personal story" in the form of a collection of content that has been created and sent/broadcast by that user. To this end, the user interface of the interaction client **104** may include an icon that is user-selectable to enable a sending user to add specific content to his or her personal story.

**[0076]** A collection may also constitute a "live story," which is a collection of content from multiple users that is created manually, automatically, or using a combination of manual and automatic techniques. For example, a "live story" may constitute a curated stream of user-submitted content from various locations and events. Users whose client devices have location services enabled and are at a common location event at a particular time may, for example, be presented with an option, via a user interface of the interaction client **104**, to contribute content to a particular live story. The live story may be identified to the user by

the interaction client **104**, based on his or her location. The end result is a "live story" told from a community perspective.

**[0077]** A further type of content collection is known as a "location story," which enables a user whose user system **102** is located within a specific geographic location (e.g., on a college or university campus) to contribute to a particular collection. In some examples, a contribution to a location story may employ a second degree of authentication to verify that the end-user belongs to a specific organization or other entity (e.g., is a student on the university campus).

**[0078]** As mentioned above, the video table **314** stores video data that, in some examples, is associated with messages for which records are maintained within the message table **306**. Similarly, the image table **316** stores image data associated with messages for which message data is stored in the entity table **308**. The entity table **308** may associate various augmentations from the augmentation table **312** with various images and videos stored in the image table **316** and the video table **314**.

**[0079]** The databases **304** also includes 3D mask table **319**. The 3D mask table **319** includes, referring to FIGS. **6** and **7**, first trained NN **616**, 3D trainable mask **608**, 3D face mask application **1210**, and so forth.

Machine Architecture

**[0080]** FIG. **4** is a diagrammatic representation of the machine **400** within which instructions **402** (e.g., software, a program, an application, an applet, an app, or other executable code) for causing the machine **400** to perform any one or more of the methodologies discussed herein may be executed. For example, the instructions **402** may cause the machine **400** to execute any one or more of the methods described herein. The instructions **402** transform the general, non-programmed machine **400** into a particular machine **400** programmed to carry out the described and illustrated functions in the manner described. The machine **400** may operate as a standalone device or may be coupled (e.g., networked) to other machines. In a networked deployment, the machine **400** may operate in the capacity of a server machine or a client machine in a server-client network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine **400** may comprise, but not be limited to, a server computer, a client computer, a personal computer (PC), a tablet computer, a laptop computer, a netbook, a set-top box (STB), a personal digital assistant (PDA), an entertainment media system, a cellular telephone, a smartphone, a mobile device, a wearable device (e.g., a smartwatch), a smart home device (e.g., a smart appliance), other smart devices, a web appliance, a network router, a network switch, a network bridge, or any machine capable of executing the instructions **402**, sequentially or otherwise, that specify actions to be taken by the machine **400**. Further, while a single machine **400** is illustrated, the term "machine" shall also be taken to include a collection of machines that individually or jointly execute the instructions **402** to perform any one or more of the methodologies discussed herein. The machine **400**, for example, may comprise the user system **102** or any one of multiple server devices forming part of the interaction platform **110**. In some examples, the machine **400** may also comprise both client and server systems, with certain operations of a particular method or algorithm being performed on the server-side and with certain operations of the particular method or algorithm being performed on the client-side.

**[0081]** The machine **400** may include processors **404**, memory **406**, and input/output I/O components **408**, which may be configured to communicate with each other via a bus **410**. In an example, the processors **404** (e.g., a Central Processing Unit (CPU), a Reduced Instruction Set Computing (RISC) Processor, a Complex Instruction Set Computing (CISC) Processor, a Graphics Processing Unit (GPU), a Digital Signal Processor (DSP), an Application Specific Integrated Circuit (ASIC), a Radio-Frequency Integrated Circuit (RFIC), another processor, or any suitable combination thereof) may include, for example, a processor **412** and a processor **414** that execute the instructions **402**. The term "processor" is intended to include multi-core processors that may comprise two or more independent processors (sometimes referred to as "cores") that may execute instructions contemporaneously. Although FIG. **4** shows multiple processors **404**, the machine **400** may include a single processor with a single-core, a single processor with multiple cores (e.g., a multi-core processor), multiple processors with a single core, multiple processors with multiples cores, or any combination thereof.

**[0082]** The memory **406** includes a main memory **416**, a static memory **418**, and a storage unit **420**, both accessible to the processors **404** via the bus **410**. The main memory **406**, the static memory **418**, and storage unit **420** store the instructions **402** embodying any one or more of the methodologies or functions described herein. The instructions **402** may also reside, completely or partially, within the main memory **416**, within the static memory **418**, within machine-readable medium **422** within the storage unit **420**, within at least one of the processors **404** (e.g., within the processor's cache memory), or any suitable combination thereof, during execution thereof by the machine **400**.

**[0083]** The I/O components **408** may include a wide variety of components to receive input, provide output, produce output, transmit information, exchange information, capture measurements, and so on. The specific I/O components **408** that are included in a particular machine will depend on the type of machine. For example, portable machines such as mobile phones may include a touch input device or other such input mechanisms, while a headless server machine will likely not include such a touch input device. It will be appreciated that the I/O components **408** may include many other components that are not shown in FIG. **4**. In various examples, the I/O components **408** may include user output components **424** and user input components **426**. The user output components **424** may include visual components (e.g., a display such as a plasma display panel (PDP), a light-emitting diode (LED) display, a liquid crystal display (LCD), a projector, or a cathode ray tube (CRT)), acoustic components (e.g., speakers), haptic components (e.g., a vibratory motor, resistance mechanisms), other signal generators, and so forth. The user input components **426** may include alphanumeric input components (e.g., a keyboard, a touch screen configured to receive alphanumeric input, a photo-optical keyboard, or other alphanumeric input components), point-based input components (e.g., a mouse, a touchpad, a trackball, a joystick, a motion sensor, or another pointing instrument), tactile input components (e.g., a physical button, a touch screen that provides location and force of touches or touch gestures, or

other tactile input components), audio input components (e.g., a microphone), and the like.

**[0084]** In further examples, the I/O components **408** may include biometric components **428**, motion components **430**, environmental components **432**, or position components **434**, among a wide array of other components. For example, the biometric components **428** include components to detect expressions (e.g., hand expressions, facial expressions, vocal expressions, body gestures, or eye-tracking), measure biosignals (e.g., blood pressure, heart rate, body temperature, perspiration, or brain waves), identify a person (e.g., voice identification, retinal identification, facial identification, fingerprint identification, or electroencephalogram-based identification), and the like. The motion components **430** include acceleration sensor components (e.g., accelerometer), gravitation sensor components, rotation sensor components (e.g., gyroscope).

**[0085]** The environmental components **432** include, for example, one or cameras (with still image/photograph and video capabilities), illumination sensor components (e.g., photometer), temperature sensor components (e.g., one or more thermometers that detect ambient temperature), humidity sensor components, pressure sensor components (e.g., barometer), acoustic sensor components (e.g., one or more microphones that detect background noise), proximity sensor components (e.g., infrared sensors that detect nearby objects), gas sensors (e.g., gas detection sensors to detection concentrations of hazardous gases for safety or to measure pollutants in the atmosphere), or other components that may provide indications, measurements, or signals corresponding to a surrounding physical environment.

**[0086]** With respect to cameras, the user system **102** may have a camera system comprising, for example, front cameras on a front surface of the user system **102** and rear cameras on a rear surface of the user system **102**. The front cameras may, for example, be used to capture still images and video of a user of the user system **102** (e.g., "selfies"), which may then be augmented with augmentation data (e.g., filters) described above. The rear cameras may, for example, be used to capture still images and videos in a more traditional camera mode, with these images similarly being augmented with augmentation data. In addition to front and rear cameras, the user system **102** may also include a 360° camera for capturing 360° photographs and videos.

**[0087]** Further, the camera system of the user system **102** may include dual rear cameras (e.g., a primary camera as well as a depth-sensing camera), or even triple, quad or penta rear camera configurations on the front and rear sides of the user system **102**. These multiple cameras systems may include a wide camera, an ultra-wide camera, a telephoto camera, a macro camera, and a depth sensor, for example.

**[0088]** The position components **434** include location sensor components (e.g., a GPS receiver component), altitude sensor components (e.g., altimeters or barometers that detect air pressure from which altitude may be derived), orientation sensor components (e.g., magnetometers), and the like.

**[0089]** Communication may be implemented using a wide variety of technologies. The I/O components **408** further include communication components **436** operable to couple the machine **400** to a network **438** or devices **440** via respective coupling or connections. For example, the communication components **436** may include a network interface component or another suitable device to interface with the network **438**. In further examples, the communication components **436** may include wired communication components, wireless communication components, cellular communication components, Near Field Communication (NFC) components, Bluetooth® components (e.g., Bluetooth® Low Energy), Wi-Fi® components, and other communication components to provide communication via other modalities. The devices **440** may be another machine or any of a wide variety of peripheral devices (e.g., a peripheral device coupled via a USB).

**[0090]** Moreover, the communication components **436** may detect identifiers or include components operable to detect identifiers. For example, the communication components **436** may include Radio Frequency Identification (RFID) tag reader components, NFC smart tag detection components, optical reader components (e.g., an optical sensor to detect one-dimensional bar codes such as Universal Product Code (UPC) bar code, multi-dimensional bar codes such as Quick Response (QR) code, Aztec code, Data Matrix, Dataglyph, MaxiCode, PDF417, Ultra Code, UCC RSS-2D bar code, and other optical codes), or acoustic detection components (e.g., microphones to identify tagged audio signals). In addition, a variety of information may be derived via the communication components **436**, such as location via Internet Protocol (IP) geolocation, location via Wi-Fi® signal triangulation, location via detecting an NFC beacon signal that may indicate a particular location, and so forth.

**[0091]** The various memories (e.g., main memory **416**, static memory **418**, and memory of the processors **404**) and storage unit **420** may store one or more sets of instructions and data structures (e.g., software) embodying or used by any one or more of the methodologies or functions described herein. These instructions (e.g., the instructions **402**), when executed by processors **404**, cause various operations to implement the disclosed examples.

**[0092]** The instructions **402** may be transmitted or received over the network **438**, using a transmission medium, via a network interface device (e.g., a network interface component included in the communication components **436**) and using any one of several well-known transfer protocols (e.g., hypertext transfer protocol (HTTP)). Similarly, the instructions **402** may be transmitted or received using a transmission medium via a coupling (e.g., a peer-to-peer coupling) to the devices **440**.

Software Architecture

**[0093]** FIG. **5** is a block diagram **500** illustrating a software architecture **502**, which can be installed on any one or more of the devices described herein. The software architecture **502** is supported by hardware such as a machine **504** that includes processors **506**, memory **508**, and I/O components **510**. In this example, the software architecture **502** can be conceptualized as a stack of layers, where each layer provides a particular functionality. The software architecture **502** includes layers such as an operating system **512**, libraries **514**, frameworks **516**, and applications **518**. Operationally, the applications **518** invoke API calls **520** through the software stack and receive messages **522** in response to the API calls **520**.

**[0094]** The operating system **512** manages hardware resources and provides common services. The operating system **512** includes, for example, a kernel **524**, services **526**, and drivers **528**. The kernel **524** acts as an abstraction layer between the hardware and the other software layers.

For example, the kernel **524** provides memory management, processor management (e.g., scheduling), component management, networking, and security settings, among other functionalities. The services **526** can provide other common services for the other software layers. The drivers **528** are responsible for controlling or interfacing with the underlying hardware. For instance, the drivers **528** can include display drivers, camera drivers, BLUETOOTH® or BLUETOOTH® Low Energy drivers, flash memory drivers, serial communication drivers (e.g., USB drivers), WI-FI® drivers, audio drivers, power management drivers, and so forth.

**[0095]** The libraries **514** provide a common low-level infrastructure used by the applications **518**. The libraries **514** can include system libraries **530** (e.g., C standard library) that provide functions such as memory allocation functions, string manipulation functions, mathematic functions, and the like. In addition, the libraries **514** can include API libraries **532** such as media libraries (e.g., libraries to support presentation and manipulation of various media formats such as Moving Picture Experts Group-4 (MPEG4), Advanced Video Coding (H.264 or AVC), Moving Picture Experts Group Layer-3 (MP3), Advanced Audio Coding (AAC), Adaptive Multi-Rate (AMR) audio codec, Joint Photographic Experts Group (JPEG or JPG), or Portable Network Graphics (PNG)), graphics libraries (e.g., an OpenGL framework used to render in two dimensions (2D) and three dimensions (3D) in a graphic content on a display), database libraries (e.g., SQLite to provide various relational database functions), web libraries (e.g., WebKit to provide web browsing functionality), and the like. The libraries **514** can also include a wide variety of other libraries **534** to provide many other APIs to the applications **518**.

**[0096]** The frameworks **516** provide a common high-level infrastructure that is used by the applications **518**. For example, the frameworks **516** provide various graphical user interface (GUI) functions, high-level resource management, and high-level location services. The frameworks **516** can provide a broad spectrum of other APIs that can be used by the applications **518**, some of which may be specific to a particular operating system or platform.

**[0097]** In an example, the applications **518** may include a home application **536**, a contacts application **538**, a browser application **540**, a book reader application **542**, a location application **544**, a media application **546**, a messaging application **548**, a game application **550**, and a broad assortment of other applications such as a third-party application **552**. The applications **518** are programs that execute functions defined in the programs. Various programming languages can be employed to create one or more of the applications **518**, structured in a variety of manners, such as object-oriented programming languages (e.g., Objective-C, Java, or C++) or procedural programming languages (e.g., C or assembly language). In a specific example, the third-party application **552** (e.g., an application developed using the ANDROID™ or IOS™ software development kit (SDK) by an entity other than the vendor of the particular platform) may be mobile software running on a mobile operating system such as IOS™, ANDROID™, WINDOWS® Phone, or another mobile operating system. In this example, the third-party application **552** can invoke the API calls **520** provided by the operating system **512** to facilitate functionalities described herein.

3D Mask Generation

**[0098]** FIG. **6** illustrates a 3D mask generation component **602**, in accordance with some examples. The text input **603** is text that describes a desired face mask for a face of user within an image of the user. Example text input **603** includes "Bengal tiger," "lion," "gorilla," "cartoon character," "elephant," and so forth. In some examples, the mask may be of a different part of the body of the user or may be a mask for a different object other than a human.

**[0099]** The generate texture component **604** generates the information for the apply texture component **614** to generate the texture **612** for the 3D trainable mask **608**. In some examples, the generate texture component **604** inputs the mask **606** and the text input **603** into a first trained NN **616**, which generates a 2D image such as image **712** of FIG. **7**. The 2D image is then used by the apply texture component **614** to generate the texture **612**. The generate texture component **604** selects a mask **606** in accordance with body part of portion of an object such as a head or face of a user within a user image **628**.

**[0100]** The 3D trainable mask **608** is a 3D representation of a mask. In some examples, the geometry **610** of the 3D trainable mask **608** is mesh vertices having x, y, and z coordinates. The 3D trainable mask **608** may be a different 3D representation of a mask such as signed distance functions (SDFs), 3D Manufacturing Format, Blender® format, irregular voxel format, and so forth. The texture **612** is a texture representation where a surface color can be determined from the 3D trainable mask **608** based on a 3D coordinate. In some examples, the texture **612** is an existing UV map, a UV map generated by the 3D mask generation component **602**, an implicit 3D representation using hash-grid positional encodings, or another representation. The texture **612** may be colors assigned to the mesh vertices. The normal map **613** is the distances of the vertices from a normal plane at the back of the 3D trainable mask **608**.

**[0101]** The apply texture component **614** adds the texture **612** to the 3D trainable mask **608**. For example, the apply texture component **614** takes a 2D image and determines a color to assign to vertices of the geometry **610**. The first trained neural network (NN) **616** is trained to take text input **603** and generate an image based on the text input **603**. In some examples, the first trained NN **616** takes as input the text input **603** and the mask **606** and generates a 2D image in accordance with the shape of the mask **606** and in accordance with the text input **603**.

**[0102]** The user image **628** is an image of a user such as a head or upper body of a user. The user image **628** may be another object or an animal. For example, the user image **628** could be the head of a dog or a refrigerator. The user image **628** may be a video or series of images of the user. The user image **628** may be captured by a camera of the user system **102**. The 3D mask generation component **602** may operate in real time to provide a 3D trainable mask **631** in accordance with the text input **603** to accommodate an application such as the 3D face mask generation application **1210** where images of the user wearing the 3D trainable mask **631** may be provided in real time to support a video chat application. The lighting information **629** may be determined by adjust text component **624**. The lighting information **629** is the lighting of the user image **628** and/or random lighting conditions to adjust the geometry **610** of the 3D trainable mask **608**. The adjust text component **624** may determine, for example, one or more light sources that are

causing the brightness or intensity settings in the user image **628** to generate the lighting information **629**.

**[0103]** The adjust text component **624** adjusts the text input **603** in accordance with the camera angle **634** and/or lighting information **638** for input to the second trained NN **630**. For example, if the camera angle **634** is from a top perspective and the text input **603** is "Bengal tiger", then the adjust text component **624** adjusts the text input **603** to generate the adjusted text **626** of "a Bengal tiger looking from above." Other examples of the adjusted text **626** include "in a front view", "in a side view", "looking up at", "looking down at", and so forth. In some examples, the adjust text component **624** indicates in the text the camera angle **634** such as indicating x, y, and z coordinates of the camera relative to the 3D trainable mask **608**.

**[0104]** Additionally, in some examples, lighting information **638** is used which includes lighting sources for the differentiable renderer component **646** to use in rendering the 3D trainable mask **608**. The adjust text component **624** analyzes the lighting information **638** and adjusts the text input **603** accordingly. For example, the lighting information **638** may be for the lighting to be a uniform white light. The adjust text component **624** includes text to indicate that the "Bengal tiger" should have "uniform white light" such as "a Bengal tiger looking from above with uniform white light." The adjust text component **624** may adjust the text input **603** in accordance with the lighting information **638** by adding text such as "in twilight", "in the dark", "during a setting sun", "in moon light", "during the day", "during the evening", "in bright noon sunlight," and so forth. In some examples, the adjust text component **624** indicates in the adjusted text **626** the lighting information **638** such as a number of lighting sources, intensities of the light sources, and the type of light for the light sources. For example, the adjusted text **626** may be "a Bengal tiger with one lighting source with red light with the lighting source at position x, y, and z." The adjust text component **624** determines the lighting information **629** from the user image **628** and the lighting information **638** is then determined based on the lighting information **629**. The adjust text component **624** uses a NN to determine the lighting information **629**, in accordance with some examples.

**[0105]** The second trained NN **630** is a trained neural network, which takes text and produces a 2D image based on the text. For example, a text input **603** of "astronaut on a horse" produces a 2D image of an astronaut riding a white horse on the moon. In some examples, the second trained NN **630** is a diffusion model where noise is input with the text input **603** and the output is the predicted noise **1010** of FIG. **11**. The second trained NN **630** has an architecture similar to U-Net with convolutional layers, down sampling, up sampling, cross layers, fully connected layers, and so forth, in accordance with some examples. The weights **1110** of the second trained NN **630** enable the second trained NN **630** to generate a 2D image from text input **603**. The first trained NN **616** and the second trained NN **616** may be the same NN.

**[0106]** In some examples, the second trained NN **630** is trained to take input 2D noise, such as Gaussian noise, and text input **603**, and output predicted noise **1010** of FIG. **11**. The predicted noise **1010** is subtracted from the input 2D noise and the process is repeated a predetermined number of times or a number of times until the difference between the output and the input 2D noise is below or transgresses a threshold. The input 2D noise in this way is transformed into the 2D image representing the text input **603**. In each iteration, the text input **603** is encoded and input into one or more of the layers **1112** of the second trained NN **630**. In some examples, in each iteration, an iteration **636** value is encoded and feed into one or more layers **1112** of the second trained NN **630**. Additionally, in some examples, the normal map **613** is encoded and input into one or more layers **1112** of the second trained NN **630**. The iteration **636** may indicate a level of Gaussian noise that was added during training to a ground truth input, which the second trained NN **630** is learning to denoise.

**[0107]** The differentiable renderer component **646** is a differential renderer that permits the gradients **642** to be backpropagated through the differential renderer component **646** to the 3D trainable mask **608**, so the geometry **610** of the 3D trainable mask **608** can be updated based on the gradients **642**. The noise generation component **644** generates noise such as noise **1004** of FIG. **10** that is added to rendered 2D output image such as output 2D mask **904** of FIG. **9**.

**[0108]** The loss determination component **640** determines gradient **642** based on a difference between the predicted noise such as predicted noise **1010** of FIG. **10** and noise **1004** generated by the noise generation component **644**. The update component **648** backpropagates the gradients **642** through one or more of: the output 2D mask with noise **1008** of FIG. **10**, the output 2D mask **904**, the normal map **613**, and the differential renderer component **646** to update the 3D trainable mask **608**.

**[0109]** The text encoder component **1102** of FIG. **11** takes the adjusted text **626** and encodes it into latent space Z1 **1103** and inputs the encoded text into one or more of the layers **1112** of the second trained NN **630**. In some examples, an encoding such as Word2Vec is used. The latent space Z1 **1108**, in some examples, is a lower dimensional latent space to reduce the complexity of training the second trained NN **630**.

**[0110]** The normal map encoder component **1106** of FIG. **11** takes the normal map **613** and encodes it into latent space Z1 **1103** and inputs the encoded normal map **613** into one or more of the layers **1112** of the second trained NN **630**. In some examples, an encoding such as Word2Vec is used. The latent space Z1 **1108**, in some examples, is a lower dimensional latent space to reduce the complexity of training the second trained NN **630**. The normal percentage **639** determines a percentage for the encoded normal map **613** relative to the encoded output 2D mask with noise **1008** that is input into the latent space Z1 **1108** and the layers **1112** by the adjust component **632**. The adjust component **632** adjusts performs the overall method to adjust the geometry **610** of the 3D trainable mask **608**.

**[0111]** The user interface (UI) component **650** accesses text input **603** and a 3D trainable mask **608**. In some examples, the text input **603** is input from a user and the UI component **650** provides a UI such as the one illustrated in FIG. **12** for the user to provide the text input **603**. The 3D trainable mask **608** is the output of the adjust component **632** in accordance with the text input **603**. For example, if the text input **603** is "Bengal tiger," then the output is the 3D trainable mask **608**, which is a 3D trainable mask **608** that can be used by a renderer to place the 3D trainable mask **608** on a face of the user as illustrated in FIG. **12**.

**[0112]** FIG. **7** illustrates a system **700** to generate an image **712**, in accordance with some examples. The generate

texture component **604** inputs the plain mask **708** and/or the face like mask **710**, which are examples of a mask **608**, into the first trained NN **616** with the text input **603**, which here is "Bengal tiger" **706**, to generate the image **712**. The image **712** is used to generate the texture **612** for the 3D trainable mask **608**.

**[0113]** FIG. **8** illustrates a system **800** for 3D mask generation, in accordance with some examples. The 3D mask generation component **602** takes a 3D trainable mask **608** and adjusts the geometry **610** to generate the adjusted trainable mask **804**, which is in the shape of what is described in the text input **603**. The 3D trainable mask **608** includes the texture **612**, which is not illustrated, as described above.

**[0114]** FIG. **9** illustrates a system **900** for 3D mask generation, in accordance with some examples. The adjust component **632** adjusts the geometry **610** of the 3D trainable mask **608** by iterating the following operations one or more times. Referring to FIGS. **9** and **10**, the differential renderer component **646** renders **902** output 2D mask **904** from the 3D trainable mask **608**, a camera angle **634**, and, optionally, lighting information **638**. The example 3D trainable mask **901** is a plain mask with the Bengal tiger texture. The 3D mask generation component **602** determines the normal map **613** from the 3D trainable mask **608**. The normal map **613** is composed of distances from the surface of the 3D trainable mask **608** to a normal plain defined by the geometry **610** of the 3D trainable mask. The adjust component **632** selects different camera angles **634** and/or lighting information **638** during different iterations of training to include a variety of camera angles **634** to train the 3D trainable mask **608**.

**[0115]** After rendering the output 2D mask **904**, the loss determination component **640** determines the gradients **642** based on the loss **908**. The operations for the loss determination **906** that determines the loss **908** are disclosed in conjunction with FIGS. **10** and **11**.

**[0116]** The update component **648** backpropagates or propagates the gradients **642** to update the geometry **610** of the 3D trainable mask **608**. The adjust component **632** repeats these operations until the loss **908** transgresses or is below a threshold value, in accordance with some examples. In some examples, the adjust component **632** performs these operations a predetermined number of times. In some examples, the adjust component **632** performs these operations for a fixed amount of time to support a real-time video application that places the 3D trainable mask **608** on a face of a user.

**[0117]** FIG. **10** illustrates loss determination **906**, in accordance with some examples. The noise generation component **644** generates noise **1004**. The adjust component **632** combines **1006** the noise **1004** with the output 2D mask **904** to generate the output 2D mask with noise **1008**. The adjust component **632** inputs the output 2D mask with noise **1008**, the normal map **613**, the text input **603**, which here is "Bengal tiger" **603**, the lighting information **638**, and the iteration **636**, into the second trained NN **630** to generate the predicted noise **1010**. The lighting information **638** and iteration **636** are optional. The loss determination component **640** determines the gradients **642** based on a comparison **1012** between the predicted noise **1010** and the noise **1004**. The comparison **1012** is a difference, in accordance with some examples. In some examples, the loss determination component **640** determines the gradients **642** based on a comparison **1012** between the predicted noise **1010** and the output 2D mask with noise **1008**. In some examples, the loss determination component **640** uses the lighting information **638** and/or the iteration **636** in determining the gradients **642**. Additionally, in some examples, the noise generation component **644** generates the noise **1004** based on an iteration **636**. In some examples, the amount of noise **1004** is greater in each subsequent iteration **636**.

**[0118]** FIG. **11** illustrates the second trained NN **630**, in accordance with some examples. The adjust text component **624** generates the adjusted text **626** from text input **603** and, optionally, the lighting information **638** and/or the camera angle **634**. In some examples, the text encoder component **1102** takes as input the adjusted text **626** generated by the adjust text component **624** and encodes the adjusted text **626** into the latent space Z1 **1108** as disclosed herein. The encoded text is input to one or more layers **1112** of the second trained NN **630**. The normal map encoder component **1106** encodes the normal map **613** into the latent space Z1 **1108**. The 2D mask encoder component **1104** encodes the output 2D mask with noise **1008** into the latent space Z1 **1108**. The adjust component **632** inputs the encoded output 2D mask with noise **1008**, the encoded adjusted text **626**, the encoded normal map **613**, and, optionally, the iteration **636**, into the input layers of the second trained NN **630** and, optionally, into one or more layers **1112** of the second trained NN **630**, to produce a result in the latent space Z2 **1116**. The output decoder component **1118** decodes the result to generate the predicted noise **1119**. The iteration **636** is an indication of the amount of noise added to the output 2D mask with noise **1008**. The predicted noise **1119** is used to adjust the geometry **610** of the 3D trainable mask **608** as disclosed herein. The repeat **1114** indicates that the method to generate the predicted noise **1119** may repeat a number of times for each iteration **636**.

**[0119]** FIG. **12** illustrates an example 3D mask generation application, in accordance with some examples. The user system **102** includes a camera **1202** and screen **1204**. The user of the user system **102** provides input text **1208**, which here is "Bengal tiger". The user then selects **1209** the user interface item 3D face mask generation application **1210**, which generates the output video **1212** from the input video **1206** and input text **1208**. The user can select save video **1216** to save the output video **1212**. The user can select "more training" **1218**, which will cause the 3D face mask generation application **1210** to perform more iterations of training the 3D trainable mask **608**. The user can select back **1220** to return to the previous screen **1204** illustrating the input video **1206**. The 3D face mask generation application **1210** is an application that uses the 3D mask generation component **602** to perform the functions disclosed herein.

**[0120]** FIG. **13** illustrates a method **1300** for 3D mask generation, in accordance with some examples. The method **1300** begins at operation **1302** with rendering a trainable 3D mask to generate a 2D image. For example, referring to FIG. **9**, the differential renderer component **646** renders the 3D trainable mask **608** to generate the output 2D mask **904**. The method **1300** continues at operation **1304** with adding noise to the 2D image to generate a 2D image with added noise. For example, the adjust component **632**, referring to FIG. **10**, adds the noise **1004** to the output 2D mask **904** to generate the output 2D mask with noise **1008**.

**[0121]** The method **1300** continues at operation **1306** with inputting the 2D image with added noise and input text into a trained neural network to generate a predicted noise of the

2D image with added noise. For example, the adjust component **632**, referring to FIG. **10**, inputs the text input **603** and the output 2D mask with noise **1008** to generate the predicted noise **1010**.

**[0122]** The method **1300** continues at operation **1308** with determining a loss between the 2D image with added noise and the predicted noise. For example, the loss determination component **640**, referring to FIG. **10**, determines gradients **1002** and the loss to generate gradients **642**. The method **1300** continues at operation **1310** with updating the trainable 3D mask based on the loss. For example, the update component **648**, referring to FIG. **9**, updates the geometry **610** of the 3D trainable mask **608** based on the gradients **642**.

**[0123]** The method **1300** optionally includes one or more additional operations, the operations of method **1300** can be performed in a different order, or one or more of the operations of the method **1300** can be optional. The method **1300** may be performed in whole or in part by one or more computing devices, or an apparatus of one or more computing devices disclosed herein. The functions of a component, such as the 3D mask generation component **602**, are performed or executed by one or more computing devices configured to perform or execute the functions of the component.

## Glossary

**[0124]** "Carrier signal" refers, for example, to any intangible medium that is capable of storing, encoding, or carrying instructions for execution by the machine and includes digital or analog communications signals or other intangible media to facilitate communication of such instructions. Instructions may be transmitted or received over a network using a transmission medium via a network interface device.

**[0125]** "Client device" refers, for example, to any machine that interfaces to a communications network to obtain resources from one or more server systems or other client devices. A client device may be, but is not limited to, a mobile phone, desktop computer, laptop, portable digital assistants (PDAs), smartphones, tablets, ultrabooks, netbooks, laptops, multi-processor systems, microprocessor-based or programmable consumer electronics, game consoles, set-top boxes, or any other communication device that a user may use to access a network.

**[0126]** "Communication network" refers, for example, to one or more portions of a network that may be an ad hoc network, an intranet, an extranet, a virtual private network (VPN), a local area network (LAN), a wireless LAN (WLAN), a wide area network (WAN), a wireless WAN (WWAN), a metropolitan area network (MAN), the Internet, a portion of the Internet, a portion of the Public Switched Telephone Network (PSTN), a plain old telephone service (POTS) network, a cellular telephone network, a wireless network, a Wi-Fi® network, another type of network, or a combination of two or more such networks. For example, a network or a portion of a network may include a wireless or cellular network, and the coupling may be a Code Division Multiple Access (CDMA) connection, a Global System for Mobile communications (GSM) connection, or other types of cellular or wireless coupling. In this example, the coupling may implement any of a variety of types of data transfer technology, such as Single Carrier Radio Transmission Technology (1×RTT), Evolution-Data Optimized (EVDO) technology, General Packet Radio Service (GPRS) technology, Enhanced Data rates for GSM Evolution (EDGE) technology, third Generation Partnership Project (3GPP) including 3G, fourth-generation wireless (4G) networks, Universal Mobile Telecommunications System (UMTS), High Speed Packet Access (HSPA), Worldwide Interoperability for Microwave Access (WiMAX), Long Term Evolution (LTE) standard, others defined by various standard-setting organizations, other long-range protocols, or other data transfer technology.

**[0127]** "Component" or "module" refers, for example, to a device, physical entity, or logic having boundaries defined by function or subroutine calls, branch points, APIs, or other technologies that provide for the partitioning or modularization of particular processing or control functions. Components or modules may be combined via their interfaces with other components to carry out a machine process. A component or module may be a packaged functional hardware unit designed for use with other components and a part of a program that usually performs a particular function of related functions. Components or modules may constitute either software components (e.g., code embodied on a machine-readable medium) or hardware components. A "hardware component" or "hardware module" is a tangible unit capable of performing certain operations and may be configured or arranged in a certain physical manner. In various examples, one or more computer systems (e.g., a standalone computer system, a client computer system, or a server computer system) or one or more hardware components or software components of a computer system (e.g., a processor or a group of processors) may be configured by software (e.g., an application or application portion) as a hardware component or software component that operates to perform certain operations as described herein. A hardware component or hardware module may also be implemented mechanically, electronically, or any suitable combination thereof. For example, a hardware component hardware module may include dedicated circuitry or logic that is permanently configured to perform certain operations. A hardware component or hardware module may be a special-purpose processor, such as a field-programmable gate array (FPGA) or an application-specific integrated circuit (ASIC). A hardware component or hardware module may also include programmable logic or circuitry that is temporarily configured by software to perform certain operations. For example, a hardware component or hardware module may include software executed by a general-purpose processor or other programmable processors. Once configured by such software, hardware components become specific machines (or specific components of a machine) uniquely tailored to perform the configured functions and are no longer general-purpose processors. It will be appreciated that the decision to implement a hardware component or hardware module mechanically, in dedicated and permanently configured circuitry, or in temporarily configured circuitry (e.g., configured by software), may be driven by cost and time considerations. Accordingly, the phrase "hardware component" (or "hardware-implemented component") should be understood to encompass a tangible entity, be that an entity that is physically constructed, permanently configured (e.g., hardwired), or temporarily configured (e.g., programmed) to operate in a certain manner or to perform certain operations described herein. Considering examples in which hardware components are temporarily configured (e.g., programmed), each of the hardware components need not be configured or instantiated at any one instance in time. For example, where

a hardware component comprises a general-purpose processor configured by software to become a special-purpose processor, the general-purpose processor may be configured as respectively different special-purpose processors (e.g., comprising different hardware components) at different times. Software accordingly configures a particular processor or processors, for example, to constitute a particular hardware component at one instance of time and to constitute a different hardware component at a different instance of time. Hardware components or hardware modules can provide information to, and receive information from, other hardware components. Accordingly, the described hardware components or hardware modules may be regarded as being communicatively coupled. Where multiple hardware components exist contemporaneously, communications may be achieved through signal transmission (e.g., over appropriate circuits and buses) between or among two or more of the hardware components. In examples in which multiple hardware components are configured or instantiated at different times, communications between such hardware components may be achieved, for example, through the storage and retrieval of information in memory structures to which the multiple hardware components have access. For example, one hardware component may perform an operation and store the output of that operation in a memory device to which it is communicatively coupled. A further hardware component or hardware module may then, at a later time, access the memory device to retrieve and process the stored output. Hardware components may also initiate communications with input or output devices, and can operate on a resource (e.g., a collection of information). The various operations of example methods described herein may be performed, at least partially, by one or more processors that are temporarily configured (e.g., by software) or permanently configured to perform the relevant operations. Whether temporarily or permanently configured, such processors may constitute processor-implemented components that operate to perform one or more operations or functions described herein. As used herein, "processor-implemented component" refers to a hardware component implemented using one or more processors. Similarly, the methods described herein may be at least partially processor-implemented, with a particular processor or processors being an example of hardware. For example, at least some of the operations of a method may be performed by one or more processors or processor-implemented components. Moreover, the one or more processors may also operate to support performance of the relevant operations in a "cloud computing" environment or as a "software as a service" (SaaS). For example, at least some of the operations may be performed by a group of computers (as examples of machines including processors), with these operations being accessible via a network (e.g., the Internet) and via one or more appropriate interfaces (e.g., an API). The performance of certain of the operations may be distributed among the processors, not only residing within a single machine, but deployed across a number of machines. In some examples, the processors or processor-implemented components may be located in a single geographic location (e.g., within a home environment, an office environment, or a server farm). In other examples, the processors or processor-implemented components may be distributed across a number of geographic locations.

**[0128]** "Computer-readable storage medium" refers, for example, to both machine-storage media and transmission media. Thus, the terms include both storage devices/media and carrier waves/modulated data signals. The terms "machine-readable medium," "computer-readable medium" and "device-readable medium" mean the same thing and may be used interchangeably in this disclosure.

**[0129]** "Ephemeral message" refers, for example, to a message that is accessible for a time-limited duration. An ephemeral message may be a text, an image, a video and the like. The access time for the ephemeral message may be set by the message sender. Alternatively, the access time may be a default setting or a setting specified by the recipient. Regardless of the setting technique, the message is transitory.

**[0130]** "Machine storage medium" refers, for example, to a single or multiple storage devices and media (e.g., a centralized or distributed database, and associated caches and servers) that store executable instructions, routines and data. The term shall accordingly be taken to include, but not be limited to, solid-state memories, and optical and magnetic media, including memory internal or external to processors. Specific examples of machine-storage media, computer-storage media and device-storage media include non-volatile memory, including by way of example semiconductor memory devices, e.g., erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), FPGA, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks The terms "machine-storage medium," "device-storage medium," "computer-storage medium" mean the same thing and may be used interchangeably in this disclosure. The terms "machine-storage media," "computer-storage media," and "device-storage media" specifically exclude carrier waves, modulated data signals, and other such media, at least some of which are covered under the term "signal medium."

**[0131]** "Non-transitory computer-readable storage medium" refers, for example, to a tangible medium that is capable of storing, encoding, or carrying the instructions for execution by a machine.

**[0132]** "Signal medium" refers, for example, to any intangible medium that is capable of storing, encoding, or carrying the instructions for execution by a machine and includes digital or analog communications signals or other intangible media to facilitate communication of software or data. The term "signal medium" shall be taken to include any form of a modulated data signal, carrier wave, and so forth. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a matter as to encode information in the signal. The terms "transmission medium" and "signal medium" mean the same thing and may be used interchangeably in this disclosure.

**[0133]** "User device" refers, for example, to a device accessed, controlled or owned by a user and with which the user interacts perform an action, or an interaction with other users or computer systems. Additional claimable subject matter further includes the following:

**[0134]** Example 1 is a computing device comprising: one or more processors; and one or more memories storing instructions that, when executed by the one or more processors, configure the one or more processors to perform operations comprising: rendering a trainable three-dimensional (3D) mask to generate a two-dimensional (2D) image;

adding noise to the 2D image to generate a 2D image with added noise; inputting the 2D image with added noise and input text into a trained neural network to generate a predicted noise of the 2D image with added noise; determining a loss between the 2D image with added noise and the predicted noise; and updating the trainable 3D mask based on the loss.

**[0135]** In Example 2, the subject matter of Example 1 includes, wherein the operations further comprise: determining a normal map for the trainable 3D mask, wherein inputting further comprises: inputting the 2D image with added noise, the input text, and the normal map into the trained neural network to generate a predicted noise of the 2D image with added noise.

**[0136]** In Example 3, the subject matter of Examples 1-2 includes, wherein determining the loss further comprises: determining the loss based on a difference between the 2D image with added noise and the predicted noise.

**[0137]** In Example 4, the subject matter of Example 3 includes, wherein the operations further comprise: determining a gradient based on the loss; backpropagating the gradient through the 2D image with added noise; backpropagating the gradient through the 3D trainable mask; and updating the trainable 3D mask based on the gradient.

**[0138]** In Example 5, the subject matter of Example 4 includes, D image with added noise further comprises: subtracting the predicted noise from the 2D image with added noise.

**[0139]** In Example 6, the subject matter of any of Examples 1-5 includes, wherein rendering further comprises: rendering, using a differential renderer component, the trainable 3D mask to generate the 2D image.

**[0140]** In Example 7, the subject matter of Example 6 includes, wherein the operations further comprise: determining a gradient based on the loss; and propagating the gradient through the differential renderer component.

**[0141]** In Example 8, the subject matter of any of Examples 1-7 includes, D images based on input texts and masks, and the trained neural network comprises one or more of: convolutional layers, one or more up sampling layers, one or more down sampling layers, and one or more fully connected layers.

**[0142]** In Example 9, the subject matter of any of Examples 1-8 includes, wherein the operations further comprise: receiving the input text from a user; accessing an image of the user; and determining a shape of the trainable 3D mask based on a head of the user within the image.

**[0143]** In Example 10, the subject matter of any of Examples 1-9 includes, D mask further comprises: selecting a camera angle; and rendering the trainable 3D mask based on the camera angle to generate the 2D image.

**[0144]** In Example 11, the subject matter of Example 10 includes, D image further comprises: modifying the input text in accordance with the camera angle; and inputting the 2D image with added noise and the modified input text into the trained neural network to generate the predicted noise.

**[0145]** In Example 12, the subject matter of Example 10 or Example 11 includes, wherein the operations further comprise: selecting one or more lighting sources, wherein the rendering is further based on the one or more lighting sources, wherein the one or more lighting sources are selected based on an image of a user.

**[0146]** In Example 13, the subject matter of Example 12 includes, D image further comprises: modifying the input text in accordance with the one or more lighting sources; and inputting the 2D image with added noise and the modified input text into the trained neural network to generate the predicted noise.

**[0147]** In Example 14, the subject matter of any of Examples 1-13 includes, D mask comprises: adjusting position of vertices of a plurality of vertices, the trainable 3D mask comprising the plurality of vertices.

**[0148]** In Example 15, the subject matter of any of Examples 1-14 includes, wherein adding noise further comprises: sampling a Gaussian distribution to determine the noise, wherein an amount of the noise is based on a number of iterations of updating the trainable 3D mask; and adding the Gaussian noise to the 2D image to generate the 2D image with the added noise.

**[0149]** In Example 16, the subject matter of any of Examples 1-15 includes, D image further comprises: inputting the 2D image with added noise, the input text, and a number of iterations into the trained neural network to generate the predicted noise of the 2D image with added noise.

**[0150]** In Example 17, the subject matter of any of Examples 1-16 includes, wherein the operations further comprise: repeating the rendering, the adding, the inputting, the determining, and the updating, until the loss transgresses a threshold value.

**[0151]** In Example 18, the subject matter of any of Examples 1-17 includes, D image, and wherein the operations further comprise: inputting a 2D mask and the text into a second trained neural network to generate a second 2D image in a shape of the 2D mask representing the text; and determining, based on colors of the second 2D image, colors for a plurality of vertices, the 3D trainable mask comprising the plurality of vertices.

**[0152]** Example 19 is a non-transitory computer-readable storage medium including instructions that, when processed by one or more processors, configure the one or more processors to perform operations comprising: rendering a trainable three-dimensional (3D) mask to generate a two-dimensional (2D) image; adding noise to the 2D image to generate a 2D image with added noise; inputting the 2D image with added noise and input text into a trained neural network to generate a predicted noise of the 2D image with added noise; determining a loss between the 2D image with added noise and the predicted noise; and updating the trainable 3D mask based on the loss.

**[0153]** Example 20 is a method performed by one or more processors comprising: rendering a trainable three-dimensional (3D) mask to generate a two-dimensional (2D) image; adding noise to the 2D image to generate a 2D image with added noise; inputting the 2D image with added noise and input text into a trained neural network to generate a predicted noise of the 2D image with added noise; determining a loss between the 2D image with added noise and the predicted noise; and updating the trainable 3D mask based on the loss.

**[0154]** Example 21 is at least one machine-readable medium including instructions that, when executed by processing circuitry, cause the processing circuitry to perform operations to implement of any of Examples 1-20.

**[0155]** Example 22 is an apparatus comprising means to implement of any of Examples 1-20.

**[0156]** Example 23 is a system to implement of any of Examples 1-20.

[0157] Example 24 is a method to implement of any of Examples 1-20.

What is claimed is:

1. A computing device comprising:
one or more processors; and
one or more memories storing instructions that, when executed by the one or more processors, configure the computing device to perform operations comprising:
rendering a trainable three-dimensional (3D) mask to generate a two-dimensional (2D) image;
adding noise to the 2D image to generate a 2D image with added noise;
inputting the 2D image with added noise and input text into a trained neural network to generate a predicted noise of the 2D image with added noise;
determining a loss between the 2D image with added noise and the predicted noise; and
updating the trainable 3D mask based on the loss.

2. The computing device of claim 1, wherein the operations further comprise:
determining a normal map for the trainable 3D mask, wherein inputting further comprises:
inputting the 2D image with added noise, the input text, and the normal map into the trained neural network to generate a predicted noise of the 2D image with added noise.

3. The computing device of claim 1, wherein determining the loss further comprises:
determining the loss based on a difference between the 2D image with added noise and the predicted noise.

4. The computing device of claim 3, wherein the operations further comprise:
determining a gradient based on the loss;
backpropagating the gradient through the 2D image with added noise;
backpropagating the gradient through the 3D trainable mask; and
updating the trainable 3D mask based on the gradient.

5. The computing device of claim 4, wherein the backpropagating the gradient through the 2D image with added noise further comprises:
subtracting the predicted noise from the 2D image with added noise.

6. The computing device of claim 1, wherein rendering further comprises:
rendering, using a differential renderer component, the trainable 3D mask to generate the 2D image.

7. The computing device of claim 6, wherein the operations further comprise:
determining a gradient based on the loss; and
propagating the gradient through the differential renderer component.

8. The computing device of claim 1, wherein the trained neural network is trained, using a diffusion model, to generate 2D images based on input texts and masks, and the trained neural network comprises one or more of: convolutional layers, one or more up sampling layers, one or more down sampling layers, and one or more fully connected layers.

9. The computing device of claim 1, wherein the operations further comprise:
receiving the input text from a user;
accessing an image of the user; and
determining a shape of the trainable 3D mask based on a head of the user within the image.

10. The computing device of claim 1, wherein rendering the trainable 3D mask further comprises:
selecting a camera angle; and
rendering the trainable 3D mask based on the camera angle to generate the 2D image.

11. The computing device of claim 10, wherein the inputting the 2D image further comprises:
modifying the input text in accordance with the camera angle; and
inputting the 2D image with added noise and the modified input text into the trained neural network to generate the predicted noise.

12. The computing device of claim 10, wherein the operations further comprise:
selecting one or more lighting sources, wherein the rendering is further based on the one or more lighting sources, wherein the one or more lighting sources are selected based on an image of a user.

13. The computing device of claim 12, wherein the inputting the 2D image further comprises:
modifying the input text in accordance with the one or more lighting sources; and
inputting the 2D image with added noise and the modified input text into the trained neural network to generate the predicted noise.

14. The computing device of claim 1, wherein updating the trainable 3D mask comprises:
adjusting position of vertices of a plurality of vertices, the trainable 3D mask comprising the plurality of vertices.

15. The computing device of claim 1, wherein adding noise further comprises:
sampling a Gaussian distribution to determine the noise, wherein an amount of the noise is based on a number of iterations of updating the trainable 3D mask; and
adding the Gaussian noise to the 2D image to generate the 2D image with the added noise.

16. The computing device of claim 1, wherein the inputting the 2D image further comprises:
inputting the 2D image with added noise, the input text, and a number of iterations into the trained neural network to generate the predicted noise of the 2D image with added noise.

17. The computing device of claim 1, wherein the operations further comprise:
repeating the rendering, the adding, the inputting, the determining, and the updating, until the loss transgresses a threshold value.

18. The computing device of claim 1, wherein the trained neural network is a first trained neural network, the 2D image is a first 2D image, and wherein the operations further comprise:
inputting a 2D mask and the text into a second trained neural network to generate a second 2D image in a shape of the 2D mask representing the text; and
determining, based on colors of the second 2D image, colors for a plurality of vertices, the 3D trainable mask comprising the plurality of vertices.

19. A non-transitory computer-readable storage medium including instructions that, when processed by one or more processors of a computing device, configure the computing device to perform operations comprising:

rendering a trainable three-dimensional (3D) mask to generate a two-dimensional (2D) image;

adding noise to the 2D image to generate a 2D image with added noise;

inputting the 2D image with added noise and input text into a trained neural network to generate a predicted noise of the 2D image with added noise;

determining a loss between the 2D image with added noise and the predicted noise; and

updating the trainable 3D mask based on the loss.

**20**. A method performed on a computing device, the method comprising:

rendering a trainable three-dimensional (3D) mask to generate a two-dimensional (2D) image;

adding noise to the 2D image to generate a 2D image with added noise;

inputting the 2D image with added noise and input text into a trained neural network to generate a predicted noise of the 2D image with added noise;

determining a loss between the 2D image with added noise and the predicted noise; and

updating the trainable 3D mask based on the loss.

* * * * *