(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 December 2020 (17.12.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/252352 A1

(51) International Patent Classification:
G06T 11/00 (2006.01) *G06T 19/00* (2011.01)

(21) International Application Number:
PCT/US2020/037548

(22) International Filing Date:
12 June 2020 (12.06.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/860,461 12 June 2019 (12.06.2019) US

(71) Applicant: **WOLF VENTURES, LLC** [US/US]; 7616 S. 2700 E., Cottonwood Heights, UT 84121 (US).

(72) Inventor: **BARKER, Jeremiah, Timberline**; 7616 S. 2700 E., Cottonwood Heights, UT 84121 (US).

(74) Agent: **EDWARDS, Terrence, J.**; TechLaw Ventures, PLLC, 3290 West Mayflower Ave, Lehi, UT 84043 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

(54) Title: DATA SERIALIZATION EXTRUSION FOR CONVERTING TWO-DIMENSIONAL IMAGES TO THREE-DIMENSIONAL GEOMETRY

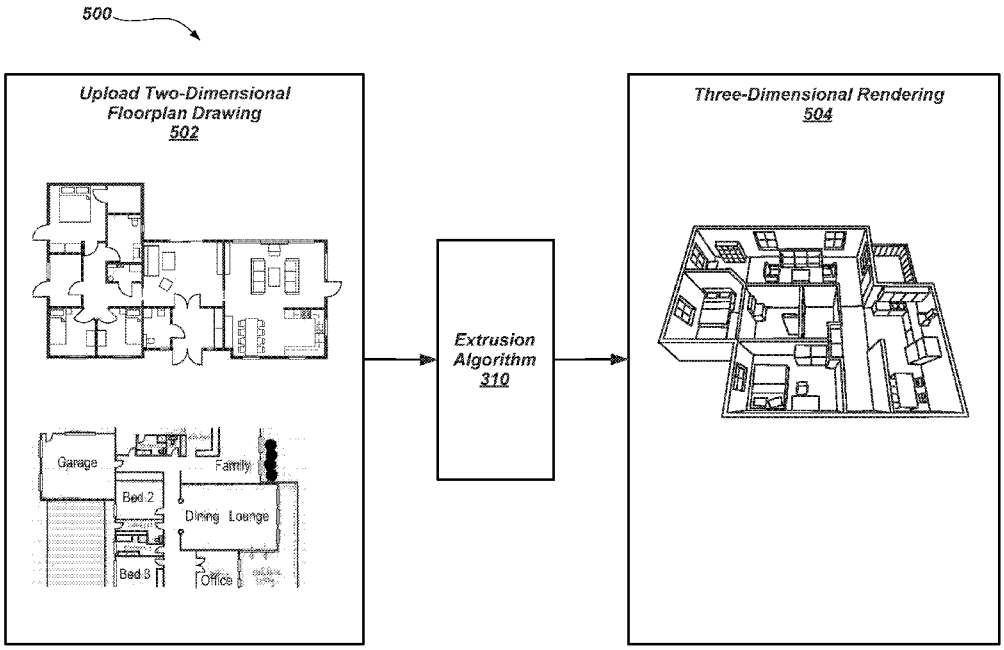

FIG. 5

(57) Abstract: Systems, methods, and devices for data serialization extrusion of input images and raw input data. A method includes determining an input image comprising one or more lines and generating a bitmap based on the input image. The method includes performing edge detection on the bitmap to identify the one or more lines in the input image by applying a growth algorithm to pixels of the bitmap. The method includes calculating a pixel width of each of the one or more lines in the input image.

WO 2020/252352 A1

[Continued on next page]

Published:
— *with international search report (Art. 21(3))*

DATA SERIALIZATION EXTRUSION FOR CONVERTING TWO-DIMENSIONAL IMAGES TO THREE-DIMENSIONAL GEOMETRY

TECHNICAL FIELD

[0001] The present disclosure relates to image processing and data storage.

BACKGROUND

[0002] Residential and commercial construction and remodeling projects can be extraordinarily complex. A relatively simple residential remodeling project, for example, can require multiple parties to make numerous decisions regarding floorplans, constructions materials, design materials, furnishings, and so forth. It is challenging to visualize how different construction and design materials will look together in a specific space. Further, and particularly for new-build construction or extensive remodeling, it can be challenging to visualize how a floorplan or cabinetry layout will look and feel.

[0003] Construction and design rendering programs currently known in the art are difficult to use and primarily directed for use by professionals who are trained to use a specific program. Additionally, these construction and design rendering programs can be extraordinarily expensive and are cost-prohibitive for average consumers. There is a need for a simple-to-use design rendering program that can generate three-dimensional renderings of a space based on two-dimensional drawings. Such a program would be useful for numerous entities in the construction and remodeling industries and would enable consumers to visualize a space with specific constructions materials prior to beginning construction.

[0004] In light of the foregoing, disclosed herein are systems, methods, and devices for improved design rendering programs. Specifically disclosed herein are systems, methods, and devices for data serialization extrusion of a two-dimensional drawing to generate a geometric three-dimensional rendering.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Non-limiting and non-exhaustive implementations of the present disclosure are described with reference to the following figures, wherein like reference numerals refer to like or similar parts throughout the various views unless otherwise specified. Advantages of the present disclosure will become better understood with regard to the following description and accompanying drawings where:

[0006] FIG. 1 is a schematic diagram of a design rendering system comprising a rendering platform operated by a rendering server in communication with a network;

[0007] FIG. 2 is a schematic block diagram comprising illustrative components of a rendering platform;

[0008] FIG. 3 is a schematic block diagram of a process flow for data serialization of a two-dimensional drawing to generate a three-dimensional geometric vector diagram;

[0009] FIG. 4 is a schematic block diagram illustrating process components of an extrusion algorithm as disclosed herein;

[0010] FIG. 5 is a process flow illustrating a two-dimensional floorplan input that can be processed by an extrusion algorithm to generate a three-dimensional rendering;

[0011] FIG. 6A is an exemplary input image comprising a two-dimensional computer-generated drawing of a residential floorplan;

[0012] FIG. 6B is a bitmap corresponding with the input image illustrated in FIG. 6A;

[0013] FIG. 7 illustrates an exemplary pixel grid comprising assignments generated pursuant to a growth and/or decay algorithm;

[0014] FIG. 8 illustrates an exemplary pixel grid comprising assignments generated pursuant to a growth and/or decay algorithm;

[0015] FIG. 9 is a schematic diagram of a process flow generating a color and texture placement that can be applied to a rendered model of a space;

[0016] FIG. 10 is a schematic diagram of a process flow for generating and applying a digital sticky note to a media file or rendering of a space;

[0017] FIG. 11 illustrate two possible embodiments of a digital sticky note;

[0018] FIG. 12 is a screen shot of an exemplary information box associated with a digital sticky note;

[0019] FIGS. 13A-13D illustrate applications of adhering a digital sticky note to an image or rendering of a space; and

[0020] FIG. 14 is a schematic block diagram of an example computing system according to an example embodiment of the systems and methods described herein.

DETAILED DESCRIPTION

[0021] Disclosed herein are systems, methods, and devices for data serialization extrusion and generating a three-dimensional rendering of a space. An embodiment of the disclosure comprises an extrusion algorithm to be executed by one or more processors for analyzing an input image. In embodiments of the disclosure, the input image may comprise a two-dimensional drawing of a floorplan, landscape design, commercial construction project, and so forth. The extrusion algorithm disclosed herein can be executed to perform data serialization extrusion on the two-dimensional drawing to generate a geometric vector diagram. In an embodiment, the geometric vector diagram is used to generate an interactive three-dimensional rendering corresponding with the two-dimensional drawing.

[0022] Before the methods, systems, and devices for image extrusion and rendering are disclosed and described, it is to be understood that this disclosure is not limited to the configurations, process steps, and materials disclosed herein as such configurations, process steps, and materials may vary somewhat. It is also to be understood that the terminology employed herein is used for describing implementations only and is not intended to be limiting since the scope of the disclosure will be limited only by the appended claims and equivalents thereof.

[0023] In describing and claiming the disclosure, the following terminology will be used in accordance with the definitions set out below.

[0024] It must be noted that, as used in this specification and the appended claims, the singular forms “a,” “an,” and “the” include plural referents unless the context clearly dictates otherwise.

[0025] As used herein, the terms “comprising,” “including,” “containing,” “characterized by,” and grammatical equivalents thereof are inclusive or open-ended terms that do not exclude additional, unrecited elements or method steps.

[0026] A detailed description of systems, methods, and devices consistent with embodiments of the present disclosure is provided below. While several embodiments are described, it should be understood that this disclosure is not limited to any one embodiment, but instead encompasses numerous alternatives, modifications, and equivalents. In addition, while numerous specific details are set forth in the following description in order to provide a thorough understanding of the embodiments disclosed herein, some embodiments may be practiced without some or all of these details. Moreover, for clarity, certain technical material that is known in the related art has not been described in detail to avoid unnecessarily obscuring the disclosure.

[0027] Referring now to the figures, FIG. 1 is a schematic diagram of a system 100 for image rendering and visualization. The system 100 includes a rendering platform 102 in communication with a rendering server 110 and a

network 120. The network 120 is in communication with the rendering server 110, and access to the network 120 may be achieved by way of a network connection 118 that may be connected with the rendering server and/or individual devices such as a personal device 114.

[0028] The rendering platform 102 includes one or more of a media component 104, an extrusion component 106, and a color and texture component 108. The rendering platform 102 may include further components and may be configured to perform additional instructions, for example according to the rendering platform 102 as discussed in FIG. 2. The rendering platform 102 can be accessed by way of a personal device 114 such as a smart phone, a tablet, a laptop, a personal computer, and so forth.

[0029] The media component 104 of the rendering platform 102 is configured to analyze and assess input media in furtherance of generating a three-dimensional rendering of a space. The media component 104 is configured to perform image recognition to identify objects, colors, shapes, textures, constructions materials, and so forth within media data. In an embodiment, the media component 104 comprises a neural network trained for image recognition and identification of objects, colors, shapes, textures, construction materials, and so forth.

[0030] In an embodiment, a user uploads media to the rendering server 110 by way of a user interface for the rendering platform 102. This media may be stored in one or more of the user database 124 and/or the texture database 126. The media may comprise images, videos, hyperlinks, products available in the marketplace, product Stock Keeping Units (SKUs), user-specified parameters, and so forth. The media component 104 is configured to analyze and assess the media to identify pertinent construction materials, design materials, floorplan preferences, textures, colors, and so forth. These identified materials and preferences may then be applied to a three-dimensional rendering of an interactive floorplan that may be customized to the user's own space.

[0031] For example, in an embodiment, the media comprises an image of an example kitchen that has been uploaded by the user. The user may indicate that the countertop shown in the image should be replicated and rendered in the user's own customizable three-dimensional rendering of the user's own kitchen floorplan. The media component 104 may be configured to analyze the image with a neural network to identify the countertop in the image and further to identify the color, texture, and/or type of countertop shown in the image. The media component 104 may, for example, determine that the example image comprises a marble countertop. The media component 104 may indicate that a marble countertop should be selected from the texture database 126, and then the marble countertop should be applied to the user's personalized and customizable rendering of the user's own kitchen floorplan.

[0032] The extrusion component 106 of the rendering platform 102 converts an input two-dimensional drawing into a geometric vector diagram. The geometric vector diagram can be used by the rendering server 110 or some other service to generate an interactive three-dimensional rendering of the objects illustrated in the input two-dimensional drawing. In an embodiment, the input two-dimensional drawing illustrates the floorplan of a residential or commercial construction project, a residential or commercial remodeling project, a landscaping construction project, and/or a landscaping remodeling project. The input two-dimensional drawing may be a computer-generated line drawing that follow traditional blueprint conventions for a construction or remodeling project. The input two-dimensional drawing may be a computer-generated line drawing that does not follow traditional blueprint conventions, and may alternatively be a hand-drawn floorplan, a photograph taken of a printed computer-generated or hand-drawn floorplan, and so forth. The extrusion component 106 analyzes and assess the input two-dimensional drawing and generates serialized data comprising a vector diagram equivalent to the floorplan illustrated in the input two-dimensional drawing. The vector

diagram can be used to generate an interactive, customizable, three-dimensional rendering of the floorplan illustrated in the input two-dimensional drawing.

[0033] The color and texture component 108 of the rendering platform 102 applies colors, textures, constructions materials, design materials, user-defined preferences, and so forth to a three-dimensional rendering of a space. In an embodiment, the color and texture component 108 receives one or more indications from a user comprising parameters and preferences for the three-dimensional rendering. These parameters and preferences may comprise desired design materials such as tile, countertop, flooring, paint, and so forth, along with desired construction materials such as windows, doors, cabinetry, and so forth, along with desired floorplans and configurations. The color and texture component 108 implements these parameters and preferences into a three-dimensional rendering of the user's own space.

[0034] In an embodiment, the color and texture component 108 comprises a neural network trained to generate realistic renderings of countertops, flooring, tile, floorplans, windows, doors, and so forth. In an embodiment, the neural network comprises a variational autoencoder (VAE), a generative adversarial network (GAN), and/or a variational autoencoder generative adversarial network (VAE-GAN). The color and texture component 108 may be trained to combine a desired color with a desired texture. The texture and/or the color may be stored in the texture database 126. The color and texture component 108 may create a new never-before-created color-texture combination according to a user's preferences.

[0035] The personal device 114 is any personal computing device that can communicate with the rendering server 110. The personal device 114 may include a smart phone, a tablet, a laptop, a personal computer, virtual or augmented reality device, and so forth. Personal devices 114 may communicate with the rendering server 110 by way of a local area network (LAN) connection, a wide area network (WAN) connection, or another network connection. In an embodiment, personal devices 114 can connection to a network 120, such as a cloud computing network or the Internet, by way of a network connection 118.

[0036] The user database 124 is in communication with the rendering server 110. The user database 124 stores information about user accounts that are associated with the rendering platform 102. The user database 124 stores information about each user that has created an account with the rendering platform 102. The user database 124 stores, for example, personal user information, user preferences, data uploaded to the user's account, data saved to the user's account, aesthetic preferences designated by the user account, and so forth.

[0037] The texture database 126 is in communication with the rendering server 110. The texture database 124 stores media data comprising textures, colors, materials, and so forth that may be used for creating a rendering of a space. For example, the texture database 126 may comprise texture renderings and texture data for different mediums that may be used in architectural design. In an example implementation, the texture database 126 comprises texture information for a variety of a countertop options, for example, numerous colors of granite countertops, numerous colors of marble countertops, numerous colors of solid surface countertops, numerous colors of laminate countertops, and so forth. Further in the example implementation, the texture database 126 may comprise textures, shapes, and configurations for different types of tile, carpet flooring, hardwood flooring, laminate flooring, vinyl flooring, hardware, and so forth. The data stored in the texture database 126 is non-limiting, and any suitable data that may be used for generating a rendering of a space may be stored in and/or associated with the texture database 126.

[0038] FIG. 2 is a block diagram of the rendering platform 102. The rendering platform 102 includes the media component 104, the extrusion component 106, and the color and texture component 108 as discussed in FIG. 1. The

rendering platform 102 may further include one or more of a digital sticky note component 212, a texture identification component 204, a color identification component 206, a three-dimensional rendering component 208, and a machine learning component 210.

[0039] For purposes of illustration, programs and other executable program components are shown herein as discrete blocks, although it is understood that such programs and components may reside at various times in different storage components of a computing device and are executed by one or more processors. Alternatively, the systems and procedures described herein can be implemented in hardware, or a combination of hardware, software, and/or firmware. For example, one or more application specific integrated circuits (ASICs) can be programmed to carry out one or more of the systems and procedures described herein. As used herein, the term "component" is intended to convey the implementation apparatus for accomplishing a process, such as by hardware, or a combination of hardware, software, and/or firmware, for the purposes of performing all or parts of operations disclosed herein. The terms "component" is intended to convey independence in how the modules, components, or their functionality or hardware may be implemented in different embodiments.

[0040] The digital sticky note component 202 stores and represents digital stick notes on two-dimensional media, three-dimensional media, a live editor or rendering of a scene, and so forth. The digital sticky note component 202 need not apply to only residential or commercial construction and/or landscaping projects and has application in numerous other industries. The digital sticky note component 202 generates a "digital sticky note" that serves as a digital representation of a graphical shape or object that is attached to a certain location on an image, photograph, video stream, real-time three-dimensional editor, and so forth. The digital sticky note comprising a location indicator relative to pertinent media, such as the image, photograph, video stream, and/or real-time three-dimensional editor. The digital sticky is movable by a user by way of a user interface. Additionally, in an embodiment, the rendering server and/or a neural network associated with the rendering server 110 generates a suggestion to place the digital sticky note at a certain location. The digital sticky note comprises information and data received from a user. This information and data may comprise manually-input parameters and preferences, media such as images, hyperlinks, video streams, and so forth, and/or information applicable to the location of the digital sticky note. In an embodiment, the digital sticky note comprises real-world location data such as global positioning system (GPS) coordinates.

[0041] In an example, a digital sticky note is applied to a three-dimensional rendering of a user's kitchen. The user manually creates the digital sticky note and places the digital sticky note on the countertop of the countertop in the three-dimensional rendering. The user uploads an image of an exemplary kitchen with the same countertop desired by user. The user may indicate that the countertop in the uploaded image should be replicated and applied to the countertop in the three-dimensional rendering. The user may upload a plurality of media comprising different possible ideas, exemplary designs, products, and so forth. The user may attach notes or thoughts to the digital sticky note to remind the user in the future of the thought process or progress in designing the countertop. In an embodiment, multiple users may view and edit the digital sticky note to coordinate with one another.

[0042] The texture identification component 204 is configured to identify a texture or type of material shown in media. In an embodiment, the texture identification component 204 comprises a neural network trained for image recognition to identify textures, colors, material types, and so forth illustrated within media.

[0043] In an example implementation, a user uploads an image of an exemplary kitchen and indicates that the backsplash tile in the exemplary kitchen should be applied to the user's personalized three-dimensional rendering of the user's own kitchen. The texture identification component 204 may provide the image to a neural network trained to

locate the backsplash tile in the image. The same neural network or an additional neural network may be trained to identify the backsplash tile or identify a close approximation to the backsplash tile. After the backsplash tile has been identified or approximated, the texture identification component 204 may retrieve from memory the same tile texture and/or an approximated tile texture that is stored in the texture database 126. In an example, the tile in the exemplary kitchen is a white subway tile of a certain size. The texture identification component 204 is configured to locate the tile in the image of the exemplary kitchen, determine the tile is a white subway tile, and determine a size of the subway tile. The texture identification component 204 may additionally be configured to scan the user database 124 to retrieve a texture file corresponding with subway tile of the same size or a similar size. The texture file may be applied to the user's personalized three-dimensional rendering of the user's own kitchen.

[0044] The color identification component 206 is configured to identify a color shown in media. In an embodiment, the color identification component 206 comprises a neural network trained for image recognition to identify textures, colors, material types, and so forth illustrated within media. The color identification component 206 may be configured to locate a certain material within media uploaded by the user. The uploaded media may include images, photographs, video streams, hyperlinks, available products, product SKUs, and so forth. The color identification component 206 may comprise a neural network trained to locate a certain material or object within the media. In an example implementation, the color identification component 206 is trained to locate the cabinets in an image of an exemplary kitchen. The color identification component 206 is additionally configured to identify the color of a certain object in the media. In the example implementation, the color identification component 206 may be configured to identify the color of the cabinets in the image of the exemplary kitchen.

[0045] The color identification component 206 may additionally be configured to approximate the color of a certain structure or object within media. Further to the example implementation discussed above, the color identification component 206 may be tasked with identifying the color of cabinets within an image of an exemplary kitchen. However, based on the quality of the image and the lighting of the cabinets, the color of the cabinets might not be consistent throughout the image. The color identification component 206 may be configured to take an "average" color of the cabinets that most likely represents the color of the cabinets in real-life.

[0046] The three-dimensional rendering component 208 is configured to create a three-dimensional rendering of a space. The three-dimensional rendering component 208 may create the three-dimensional rendering using the serialized data (see 314 at FIG. 3) output by the extrusion algorithm (see 310 at FIG. 3). The serialized data 314 may include a geometric vector diagram indicating the (x,y,z) coordinates of walls, windows, doors, and other structures in a floorplan. In an alternative embodiment, the geometric vector diagram may include the (x,y,z) coordinates of hardscaping, trees, plants, retaining walls, lawn, and so forth in a landscaping design. The three-dimensional rendering component 208 is configured to create an interactive and personalized three-dimensional rendering of a space based on an input two-dimensional drawing of the space.

[0047] In an embodiment, the three-dimensional rendering component 208 provides the serialized data 314 output from the extrusion algorithm 310 to an external service configured to create the three-dimensional rendering. In such an embodiment, the three-dimensional rendering component 208 may be configured to convert the serialized data 314 into a certain file-type that can be used by an external rendering program.

[0048] The three-dimensional rendering is a three-dimensional model comprising individual geometric elements. The three-dimensional model is interactive in real-time and can be amended by a user by way of the rendering platform 102. The three-dimensional rendering may additionally include rendered images at different angles to illustrate

additional view and details of the space. The three-dimensional rendering may be viewed with virtual reality or augmented reality modes.

[0049] The machine learning component 210 is configured to improve operations of the rendering platform 102 by analyzing past inputs, outputs, and issues experienced by the rendering server 110 during operation of the rendering platform 102. Additionally, the machine learning component 210 is configured to analyze media provided by a user to locate certain objects within the media, identify certain objects, materials, colors, textures, or surfaces within the media, and so forth. The machine learning component 210 may comprise one or more neural networks configured for different tasks and trained with different sets. The machine learning component 210 may comprise one or more of a radial basis forward (RBF) network, a deep feed forward (DFF) network, a recurrent neural network (RNN), a long/short term memory (LTSM) network, a gated recurrent unit (GRU) network, an autoencoder (AE) network, a variational autoencoder (VAE) network, a denoising autoencoder (DAE) network, a sparse autoencoder (SAE) network, a deep belief network (DBN), a deep convolutional network (DCN), a deconvolutional network (DN), a deep convolutional inverse graphics network (DCIGN), a generative adversarial network (GAN), a liquid state machine (LSM), an extreme learning machine (ELM), an echo state network (ESN), a deep residual network (DRN), a support vector machine (SVM), a neural Turing machine (NTM), and so forth. The type of neural network deployed by the machine learning component 210 may be selected based on the type of task to be executed by the machine learning component 210. Any suitable neural network may be used depending on the type of task to be executed and the efficiency of the neural network in executing that task.

[0050] The user account component 212 stores information pertaining to a user's account with the rendering platform 102. The user account component 212 may prompt the user to enter personal information or to create an account with the rendering platform 102. The user account component 212 may store information about data that has been uploaded, saved, or notated to the user account by the user or some other party. In an embodiment, a user creates a user account when engaging with the rendering platform 102 and that user account information is stored on the user database 124 in communication with the rendering server 110. The user may login to the user account by way of an Internet connection.

[0051] FIG. 3 is a schematic drawing of a process flow 300 and component diagram for generating a three-dimensional output image based on input data. The process flow 300 may be referred to herein as an "extrusion" process for extruding information from a two-dimensional drawing to generate a three-dimensional rendering. The extrusion process includes two primary phases. The first phase includes receiving an input 302, providing the input 302 to the service layer 308 for performing the extrusion algorithm 310, and receiving the output 312 of the extrusion algorithm 310. The second phase of the extrusion process includes generating additional instructions to be provided to a renderer 316 in combination with the output 312 of the service layer 308. The additional instructions may include, for example, Z-axis coordinates, rulesets on what data should be kept and what data should be discarded, and/or three-dimensional mesh rules that enable the renderer 316 to generate the three-dimensional rendering.

[0052] The process flow 300 includes receiving inputs 302 including one or more of an input image 304 and raw input data 306. The process flow 300 includes providing the inputs 302 to a service layer 308 where the extrusion algorithm 310 is performed. The process flow 300 includes receiving output 312 from the service layer 308, wherein the output 312 may include serialized data 314. The process flow 300 includes providing the serialized data 314 to the renderer 316 that includes a parser 318 for generating a three-dimensional rendering based on the inputs 302.

[0053] In an embodiment, the renderer 316 and parser 318 are included in a third-party application for generating the three-dimensional rendering based on the inputs 302. Example third-party applications include Unity 3D, Unreal, Blender, 3DSMax, and Maya. One or more of these third-party applications can render the serialized data and parse what is needed to generate a three-dimensional model of the scene when provided with additional instructions on how to execute the rendering. These additional instructions enable a third-party application (or internal application) to visually extrude the code and add Z-axis coordinates to generate the three-dimensional model. In an embodiment, the parser 318 includes a neural network or provides information for training a neural network. The neural network may be trained on a set of rules regarding which artifacts should be kept and which artifacts should be discarded. In an embodiment, the system 100 further includes a library storing training information regarding what artifacts should be kept or discarded.

[0054] The input image 304 may include a drawing of an architectural floorplan, landscaping plan, or other diagram. In an embodiment, the input image 304 is a two-dimensional drawing, and in alternative embodiments the input image 304 may include a capture of a three-dimensional drawing. The input image 304 may include a computer-generated drawing, a hand-drawn drawing, a photograph of a printed computer-generated drawing, a photograph of a hand-drawn drawing, and so forth. The input image 304 may include any suitable data type such as, for example, jpg, jpeg, png, pdf, raw data captured by a camera image sensor, and so forth.

[0055] The raw input data 306 may include raw code from an augmented reality program. In an embodiment, an augmented reality program may be leveraged to capture measurements, shapes, and dimensions of an existing space. For example, a user may leverage light detection and ranging (LIDAR) capabilities of a mobile computing device in conjunction with an augmented reality program to scan an existing space and capture the measurements, shapes, and dimensions of the space. The existing space may include, for example, the architectural structure of a building, landscaping structures, semi-permanent or permanent structures within a space such as cabinetry, doors, windows, plumbing fixtures, and so forth, non-permanent structures within a space such as furniture and other furnishings, and so forth. In an embodiment, a LIDAR system is used to measure the distance from the LIDAR transmitter to a wall, cabinet, or other structure, and to thereby calculate the shapes, measurements, and dimensions of existing of rooms and other structures. These shapes, measurements, and dimensions may be provided as raw input data 306 for use in generating a three-dimensional rendering of a space. In an embodiment, the raw input data 306 includes raw LIDAR data.

[0056] In an embodiment, the raw input data 306 generated by augmented reality and/or LiDAR can be used to generate a geometric vector diagram of a scene. The geometric vector diagram includes x,y,z serialized data that may be used to generate a three-dimensional rendering of a scene. LiDAR includes the use of lasers for exceptional accuracy and augmented reality may be based more on a user's movement through an environment.

[0057] In an embodiment, the extrusion algorithm 310 is wrapped around a service layer that can run on the rendering server 110 independent of other backend processes performed by the rendering server 110. The extrusion algorithm 310 is configured to serialize the input 302 to generate serialized data 314 that can be provided to the renderer 316. The extrusion algorithm 310 may include instructions to be performed by one or more processors for reading the input 302, tracing the input 302 data, graphing the input 302 data, and prioritizing the input 302 data to generate the serialized data 314.

[0058] The output 312 of the extrusion algorithm 310 within the service layer 302 is serialized data 314. The extrusion algorithm 310 includes instructions for serialization to translate the input 302 to a format that can be stored,

transmitted, and/or reconstructed later for other uses. When the serialized data 314 is reread according to the serialization format, the serialized data 314 can be used to create a semantically identical clone of the input 302.

[0059] The renderer 316 parses the serialized data 318 to generate a three-dimensional rendering of the input 302. In an embodiment, the renderer 316 is an application program interface (API) that may function as a real-time editor and generator of three-dimensional renderings of a space. The renderer 316 may create a three-dimensional rendering of a space that can be altered and interacted with by a user in real-time. The renderer 316 may create an image of a rendering of a space that may be saved, printed, or transmitted. In an embodiment, a user of the rendering platform 102 may interact with the renderer 316 API to amend the three-dimensional rendering by, for example, adding walls or other structures, moving wall or other structures, deleting walls or other structures, and so forth. The renderer 316 may be run on the rendering server 110 and may operate independently of the extrusion algorithm 310.

[0060] In an embodiment, the process flow 300 includes applying additional data and instructions to the output 312 of the service layer 308. This additional data and instructions can be provided to a third-party renderer 316 to enable the renderer 316 to parse the serialized data 318 and generate a three-dimensional model of the input 302. The renderer 316 may need to be instructed on how to generate the three-dimensional model. These additional instructions may include, for example, Z-axis coordinates applied to the serialized data 314. The serialized data 314 may include a geometric vector diagram representing lines and other objects within the input 302. Z-axis coordinates may additionally be applied to the geometric vector diagram to enable the renderer 316 to generate a three-dimensional model based on the two-dimensional input 302. This may be particularly useful in an implementation where the input image 304 is a two-dimensional drawing of an architectural floorplan, city design, landscaping design, and so forth, and the process flow 300 is applied to generate a three-dimensional model that represents the input image 304. The additional instructions may further include, for example, instructions on what data should be kept and what data should be discarded, and/or instructions on how to generate the three-dimensional model. These instructions may be implemented by "internal" processors for the rendering server 110 or may be provided to an external program configured to generate a three-dimensional model.

[0061] FIG. 4 is a schematic diagram illustrating a process flow 400 that may be performed by one or more processors executing the extrusion algorithm 310. Given a specific input 302, the service layer 308 may run a process including a plurality of layers to create the serialized data 314. In the process flow 400 illustrated in FIG. 4 the bottom of the extrusion algorithm 310 is the base layer and each layer builds upon the other.

[0062] Prior to performing the extrusion algorithm 310, the input 302 may be validated and normalized to a set standard. For example, the input 302 may be validated to determine whether the input 302 is of a sufficient quality for performing the extrusion algorithm 310. In an embodiment, the input 302 is validated to determine whether it satisfies certain conditions such as file size, dimensions, clarity, white balance, and so forth. After the input 302 is validated, the extrusion algorithm 310 includes creating a bitmap 402 based on the input 302. The bitmap 402 represents the input 302 as an image. In an embodiment, the bitmap 402 represents the input 302 as a Red-Green-Blue (RGB) image comprising eight bit channels. The bitmap 402 may be loaded to an image library in any standard image format such as jpg, jpeg, png, pdf, and so forth.

[0063] The extrusion algorithm 310 includes performing one or more of edge detection 404, line weight detection 406, and region detection 408 on the bitmap 402. The edge, line weight, and region detections 404-408 may be performed by executing value 412, decay 414, and follow peak 416 steps.

[0064] The value 412 step includes generating a binary map indicating the distance each pixel in the bitmap 402 is from the nearest edge. In an embodiment, the value 412 includes analyzing the color of each pixel and returning a binary integer indicating whether the pixel is "dark enough" to be considered as being "on" and further to be considered a portion of a line or other object within the input image.

[0065] The decay 414 step may include generating a depth map based on the binary map and indicating the distance each pixel is from the nearest edge. The decay 414 step may include receiving information from the value 412 step indicating whether pixels are "on" or "off" and then generating "mountains" from the lines. The "center" of a line is the peak of the mountain and the edge of the line is the lowest point in the mountain. The decay 414 layer may further fine tune the artifacts to keep or discard.

[0066] The follow peak 416 step may include tracing lines based on the depth map. The region detection 408 includes identifying closed regions, and this may be performed with a flood fill algorithm in some embodiments. The extrusion algorithm 310 may include identifying region types, for example, whether a region comprises a wall or a bank of cabinets, based on the results of the region detection 408. The follow peak 416 may be analogized to walking on a roofline until the roofline turns or terminates. In the analogy, a change in the direction to the roofline is similar to a change in direction of a line of the bitmap 402. When the "on" pixels of a line terminate in the original direction and continue in a different direction, the line has changed direction and may include an angle, a curve, or some other geometric change.

[0067] The edge detection 404 process is performed to identifying edges of "on" or "off" color pixels within the bitmap 402. The results of the edge detection 404 process can indicate where the perimeter of a floorplan is located, where walls within the floorplan are located, where cabinets within the floorplan are located, and so forth. The edge detection 404 process includes examining the bitmap 402 to determine the values 412 of RGB pixels and then determine whether those values 412 satisfy a threshold. The threshold indicates whether the value 412 of the pixel qualifies the pixel as being "on" or "off." A pixel that is deemed as being "on" may indicate a wall, cabinet, or other structure within the floorplan of the input 302 data. A pixel that is deemed as being "off" may indicate an open space or non-wall within the floorplan.

[0068] In an embodiment, the edge detection 404 process includes applying a stencil buffer of the same resolution of the bitmap 402 to represent pixels as being "on" or "off." The results of the stencil buffer will essentially yield a black and white solid shape. After the stencil buffer has been performed, then spaces within the stencil buffer can be sampled. In a perfect implementation, empty space pixels are a perfect white and non-empty space pixels (*e.g.*, walls, cabinets, structures, and so forth) are a perfect black. In most implementations, this is not the case, and the pixels will return a range of values. In an embodiment, edge detection 404 process includes assigning a value 412 to each pixel and then grouping lighter pixel values together and grouping darker pixel values together. The lighter pixel values may be associated with pixels that are "off," *i.e.* pixels that represent an empty space in the floorplan. The darker pixel values 412 may be associated with pixels that are "on," *i.e.* pixels that represent a non-empty space within the floorplan such as a wall, cabinet, or other structure. This process may be done for each input 302 on a case-by-case basis. In an embodiment, the edge detection 404 process begins with finding a midpoint value within the image. Typically, there is a smaller range for the light values than for darker values, and therefore, it can be easier to identify what classifies as white within the bitmap 402. This process is typically more challenging when the input 302 is a photograph of a line drawing or other document rather than a direct input of a computer-generated line drawing.

[0069] In an example implementation of the edge detection 404 process, the extrusion algorithm 310 is performed on a black-and-white line drawing of an architectural floorplan. The bitmap 402 of the architectural floorplan may have a plurality of walls that need to be identified. The border pixels of the walls may be a darker color than the interior pixels of the walls. Even still, the walls in the architectural floorplan can be identified by sampling the pixels in the floorplan and determining whether the pixels are dark or light.

[0070] In an embodiment, the edge detection 404 process includes applying a growth algorithm. As discussed herein, a "growth algorithm" includes processes and algorithms associated with its inverse, which may commonly be referred to as a decay algorithm. A growth algorithm and a decay algorithm are inverses of one another and may functionally result in the same output information. Reference herein to a "growth algorithm" includes processes, steps, methods, and algorithms are associated with each of a growth algorithm and a decay algorithm. The growth algorithm may be applied to the bitmap 402 pixel-by-pixel to identify which pixels are "on" and which pixels are "off." The growth algorithm can be used to identify the edges and peaks of lines within the bitmap 402.

[0071] The line weight detection 406 process can be implemented to differentiate walls from other lines such as cabinets, doors, windows, and so forth. In typical floorplan drawings, walls will be illustrated with darker or thicker lines than other objects. In an embodiment, the stencil buffer provides valuable information for the line weight detection 406 step. It can be important to differentiate between walls, windows, cabinets, doors, and so forth, but in the stencil buffer stage, the input consists only of solid lines. After the stencil buffer is applied, the extrusion algorithm 310 includes identifying lines and assigning a weight to the lines. This may be performed by implementing a grow algorithm. The grow algorithm includes beginning from a center pixel within a line and growing from there to identify whether the pixel is part of a line, and further to identify the thickness of the line. At each layer, the width of the line will decrease.

[0072] In an embodiment, the growth algorithm includes removing a first layer of a line pixel-by-pixel. This first layer may be referred to as layer one. After the first layer of the line has been removed, the growth algorithm proceeds with removing the second layer of the line. The second layer of the line may be referred to as layer two. The growth algorithm includes counting the number of layers that are removed from the line until the center of the line has been reached. This count can be used to determine the thickness of the line.

[0073] In an embodiment, the growth algorithm is referred to as a decay 414 algorithm at the pixel level. As discussed herein, the term "growth algorithm" includes processes, steps, methods, and algorithms associated with a growth algorithm and/or a decay algorithm. The growth algorithm is built from the decay algorithm and may occur simultaneously as the growth algorithm and the decay algorithm work hand-in-hand. The growth algorithm travels around each line to add mass or weight while the decay algorithm performs the inverse of this operation. The decay algorithm may run very quickly to trim pixel edges off lines through a decay process. In an embodiment, a timestamp is recorded that represents when the decay algorithm reached a certain pixel. Termination of the decay algorithm may occur when all pixels have been removed from the bitmap 402.

[0074] After the decay 414 algorithm is performed, the peak of each line is identified and followed to identify the geometry of the line (see follow peak 416). This can be particularly useful in an implementation where lines within the floorplan are shaded differently to represent different structures. For example, interior walls and exterior walls may be shaded differently, or there may be a shading system for identifying load-bearing walls versus non-load-bearing walls, and so forth. When the peak is identified, the peak is followed continually until it ends. The peak ends when a different structure is identified. This may occur when a wall rounds a corner or ends, for example. After the end of the peak has been identified, then the algorithm returns to the first identified location of the peak, and the peak is travelled in the

opposite direction until it ends again. In an embodiment, the results of this process are analyzed to identify patterns within input 302 drawing. Such patterns may include, for example, an indication that lines with certain weight ranges are associated with interior walls, exterior walls, load-bearing walls, non-load-bearing walls, cabinets, windows, doors, and so forth. These patterns can be saved and analyzed by a machine learning algorithm to aid in assessing future input 302 drawings.

[0075] In an embodiment, the ending point of the peak indicates where a line changes direction rather than a complete termination of the line. At a corner, curve, or other point where a line changes direction, the peak will terminate and begin again in the new direction. It should be appreciated that the ending point of a line may indicate the beginning of a new direction for the line. The ending point may signify where the line takes a turn of any angle.

[0076] After interior and exterior walls, cabinets, doors, windows, and so forth have been identified based on the line weight detection 406 process, the extrusion algorithm 310 performs region detection 408 to identify closed spaces within the floorplan or landscaping design. The region detection 408 includes determining where rooms or areas are connected or not connected. The region detection 408 process identifies enclosed regions such as rooms, cabinets, islands, and so forth. In an embodiment, the region detection 408 includes a flood fill algorithm. The flood fill algorithm includes analyzing the image to locate empty values, and then filling the corresponding region with a flood fill.

[0077] In an embodiment, the flood fill algorithm begins with a single pixel and fills in that pixel. The flood fill algorithm continues by identifying neighboring pixels and filling in those neighboring pixels. The flood fill algorithm continues this process recursively until hitting an edge such as a wall, cabinet, door, window, and so forth. The termination of the flood fill algorithm occurs when an edge has been reached. In an embodiment, each closed or open region is filled with a different color value. It should be appreciated that the flood fill algorithm may apply any color value to a region, including true black or true white. The region detection 408 steps can be used to identify enclosed areas and show the geometry within a certain region, and further to identify the width and shape of the region.

[0078] In an embodiment, the extrusion algorithm 310 further includes hiding and adding to the two-dimensional geometric space. Portions of the two-dimensional geometric space may be hidden and discarded. Additionally, the extrusion algorithm 310 may add to portions of the two-dimensional geometric space to apply vertical space by way of Z coordinates or other information needed to generate a three-dimensional rendering of the space.

[0079] The extrusion algorithm 310 results in a serialized data 314 output. In some embodiments, the serialized data 314 output may be referred to as metadata. The serialized data 314 represents the two-dimensional geometry extracted from the bitmap 402 and can be used to construct three-dimensional geometry. The serialized data 314 may be automatically created code that represents defined parameters. The parameters may include specific patterns found in a line drawing. The serialized data 314 output may be configured into any three-dimensional modeling editor and may be ready for common file extensions such as STL, OBJ, FBX, COLLADA, 3DS, IGES, STEP, VRML/X3D, and so forth.

[0080] FIG. 5 illustrates a process flow 500 for inputting a floorplan drawing into the extrusion algorithm 310 and generating a three-dimensional rendering that represents the floorplan drawing. In an embodiment, a user uploads at 502 a two-dimensional floorplan drawing. The user may upload this drawing to the rendering server 110 by accessing a user interface for the rendering platform 102. The two-dimensional floorplan drawing is stored in the user database 124 and may be processed by one or more processors executing the extrusion algorithm 310. The output of the extrusion algorithm 310 is a vector map that can be used to generate a three-dimensional rendering 502 of the floorplan drawing.

[0081] FIGS. 6A and 6B illustrate an input image 304 and a corresponding bitmap 402, respectively. The input image 304 illustrated in FIG. 6A comprises a two-dimensional drawing of a residential floorplan. This exemplary input image 304 is computer-generated, although other hand-drawn input images 304 can also be processed as discussed herein. The input image 304 comprises a plurality of closed regions, including a porch, a master bedroom, a master bathroom, an additional bedroom, an additional bathroom, and a combined living space comprising a great room, a dining area, and a kitchen. These written indications may be read and stored by the rendering server 110 for possible future use. The input image 304 additionally comprises dimensions corresponding with real-world measurements. In an embodiment, the rendering server 110 reads and extracts these dimensions and stores them in the user database 124 for possible future use. The input image 304 additionally includes indications of whether lines indicate an exterior wall, an interior wall, a window, a door, a cabinet, or furniture. For example, as shown, the walls are marked with dark black lines, the windows are shown as a narrow rectangle within a wall, and the doors as shown as gaps in the walls with a triangular opening indicating which direction the door will swing. The cabinetry in the kitchen and the possible furniture are illustrated with faint black lines. This input image 304 may be processed according to the extrusion algorithm 310 to generate the bitmap 402 illustrated in FIG. 6B.

[0082] The bitmap 402 is a black-and-white drawing illustrating where the input image 304 data is "on" or "off." As shown in the exemplary bitmap 402 illustrated in FIG. 6B, the negative "off" space is a perfect black while the positive "on" space is white. The negative space represents the white areas and lighter-colored areas in the input image 304. The white lines represent the black areas and darker-colored areas in the input image 304. As shown, the white lines in the bitmap 402 correspond with the dark black lines in the input image 304 that represent the walls of the floorplan. The bitmap may be analyzed according to the steps of the extrusion algorithm to generate serialized data 314. The serialized data 314 may comprise a two-dimensional and/or three-dimensional vector diagram comprising information for generating a three-dimensional rendering of the floorplan illustrated in the input image 304.

[0083] FIG. 7 illustrates an exemplary grid of pixels in a bitmap. The pixels have been assigned a number indicating the result of performing a growth or decay algorithm on the grid of pixels. The pixels comprising a "0" (zero) designation are identified as being "off" or not comprising a line. The pixels comprising a "1" (one) or "2" (two) designation are identified as being "on" or comprising some black or darker-colored value. These pixels are likely part of a line in the original input image 304. The center peak of the line comprises the "2" designation. The neighboring pixels with a "1" designation are also "on" and comprise a black or darker-colored value in the original input image 304. These assignments indicate where the pixels are within the growth/decay algorithm process.

[0084] FIG. 8 illustrates an exemplary grid of pixels in a bitmap. Similar to the grid of pixels illustrated in FIG. 7, the pixels have been assigned a number indicating the result of performing a growth or decay algorithm. In contrast with FIG. 7, none of the pixels illustrated in FIG. 8 have a "0" designation indicating the pixel is "off" or comprising white or a lighter-colored value in the original input image 304. Each of the pixels illustrated in FIG. 8 comprises a designation of "1" (one), "2" (two), or "3" (three). These assignments indicate whether the pixels are within the growth/decay algorithm process. The center peak of the line are the pixels comprising the "3" designation. The surrounding neighboring pixels comprising a "2" designation or a "1" designation are removed on subsequent iterations of the growth/decay algorithm.

[0085] FIG. 9 is a schematic flow chart diagram of a process flow 900 for generating a three-dimensional rendering 304 based on the output of an extrusion algorithm 310. The input 902 of the process flow 900 includes a geometric vector diagram 904 and/or an exemplary input image 906. The rendering is completed by a service layer 908 that

executes the identification and application 910 algorithm. The process of identification and application 910 includes identifying a saved texture at 912 and identifying an element location at 914 for applying a texture to a three-dimensional model 932. The identification and application 910 process further includes normalizing the color at 916 for the texture and defining the texture at 918. The identification and application 910 process includes applying diffuse mapping at 920. The service layer 908 executes the identification and application 910 algorithm to generate a three-dimensional model 932 of a space that comprises personalized color and texture modeling.

[0086] The geometric vector diagram 904 is an output of the extrusion algorithm 310. In an embodiment where the process flow 900 is applied to an architectural or construction design, the geometric vector diagram 904 comprises serialized data indicating where walls, cabinets, doors, windows, and other structures in a space are located. In an embodiment where the process flow 900 is applied to a landscaping design, the geometric vector diagram 904 may comprise indications of where hardscaping, pathways, retaining walls, trees, fences, plants, and other elements are located within a landscaping design. The geometric vector diagram 904 may be saved in any suitable file format that may be used by other third-party rendering programs.

[0087] The exemplary input image 906 is an image or other media uploaded by a user comprising an exemplary design or material selection. The exemplary input image 906 may comprise one or more of an image, a photograph personally captured by a user, an available product, a product SKU, a note or preference manually input by a user, a video stream, a hyperlink to an image or other media, and/or an output from a machine learning algorithm trained to predict user preferences. The exemplary input image 906 comprises one or more elements to be applied to the three-dimensional model 932. In an example implementation, the exemplary input image 906 is an image of a kitchen, and the user has indicated that the backsplash in the example kitchen should be applied to the three-dimensional model 932. The user may further indicate that the size/scale of the backsplash, the texture of the backsplash, and/or the color of the backsplash in the exemplary input image 906 should be modified when generating the three-dimensional model 932.

[0088] The service layer 908 is a processing layer of the rendering server 110. The service layer 908 may run independently of other layers in the rendering server 110. The service layer 908 comprises one or more processors for executing an identification and application 910 algorithm or applying colors, textures, and other preferences to a three-dimensional model 932 of a space. The identification and application 910 algorithm comprises at least five steps, including identifying a saved texture at 912, identifying an element location at 914, normalizing a preferred color at 916, defining an applicable texture at 918, and applying diffuse mapping at 910.

[0089] The process of identifying a saved texture at 912 comprises locating and retrieving an applicable texture saved in the texture database 126. The process of identifying a saved texture 912 may further include analyzing the exemplary input image 906 to locate an applicable structure, construction material, furnishing, and so forth. Examples include tile, flooring, countertops, furnishings, plumbing fixtures, lighting fixtures, cabinets, wall textures, and so forth. This process may be executed by a neural network trained to locate a certain structure or object within an image. The neural network may be trained to distinguish countertops from backsplash or flooring, for example, and provide an indication of where the applicable structure or object is located within the exemplary input image 906. The process of identifying a saved texture at 912 may further include analyzing the structure or object within the exemplary input image 906 to determine what type of texture it has. This process may be executed by a neural network trained to differentiate, for example, different types of tile, different types of countertop surfaces, different types of cabinets, and so forth.

[0090] The process of defining an element location at 914 comprises identifying a location within the geometric vector diagram 904 and/or the three-dimensional model 932 where a certain color, texture, structure, or object should

be applied. This may be done at least in part based on a manual input from a user indicating where the color, texture, structure, or object should be applied.

[0091] The process of normalizing a color at 916 comprises identifying a color within the exemplary input image 906. In an embodiment, a user provides a manual input comprising a color code, for example an RGB or HEX# color identifier. In an embodiment, a user provides a manual input by using an "eyedropper" tool to identify a specific color within the exemplary input image 906. The user implements the eyedropper tool by hovering over a certain element within the exemplary input image 906. The eyedropper normalizes all pixels within its magnifier into one color identified by an RGB or HEX# color identifier.

[0092] In an embodiment, the normalizing a color 916 process is performed automatically rather than receiving a manual input from a user. In the automated process, a neural network identifies an object within the exemplary input image 906 (for example, a countertop, cabinet, flooring, etc.) and generates a normalized color for that object. The normalizing color 916 process may additionally include generating a complete texture file and/or diffuse map to represent the object in the exemplary input image 906. In some implementations, based on the quality and lighting of the exemplary input image 906, a structure within the exemplary input image 906 might not have a consistent color throughout. The process of normalizing the color at 916 includes identifying an "average" color for a certain structure within the exemplary input image 906. For example, this process may include defining the "average" shade of blue across a bank of cabinets within the exemplary input image 906, and then selecting that average shade of blue to be applied to the three-dimensional model 932.

[0093] The process of defining a texture at 918 comprises locating and retrieving an applicable texture saved in the texture database 126. The process of identifying a saved texture 912 may further include analyzing the exemplary input image 906 to locate an applicable structure, construction material, furnishing, and so forth. Examples include tile, flooring, countertops, furnishings, plumbing fixtures, lighting fixtures, cabinets, wall textures, and so forth. This process may be executed by a neural network trained to locate a certain structure or object within an image. The neural network may be trained to distinguish countertops from backsplash or flooring, for example, and provide an indication of where the applicable structure or object is located within the exemplary input image 906. The process of identifying a saved texture at 912 may further include analyzing the structure or object within the exemplary input image 906 to determine what type of texture it has. This process may be executed by a neural network trained to differentiate, for example, different types of tile, different types of countertop surfaces, different types of cabinets, and so forth.

[0094] In an embodiment, the process of defining a texture at 918 includes drawing or creating a new texture rather than retrieving a stored texture from the texture database 126. In an embodiment, this is executed by a neural network trained to draw textures and layouts of textures based on an exemplary input image 906.

[0095] The process of applying diffuse mapping at 920 comprises applying a texture file to a rendering. In an embodiment, a texture file is retrieved from the texture database 126, and the texture file comprises a plurality of layers for applying the texture to a three-dimensional rendering, two-dimensional image, and so forth. A diffuse map is a texture map that may define one or more of the color, pattern, and repeating of an object. The process of applying diffuse mapping at 920 may be analogized to painting an image (*i.e.*, the texture file) on to the surface of an object (*e.g.*, a cabinet, wall, floor, or other object within a rendering).

[0096] FIG. 10 is a schematic diagram of a process flow 1000 for placing a digital sticky note on an interactive rendering of a space. The process flow 1000 includes placing a digital sticky note at 1002 and adding data to the digital sticky note at 1004. The digital sticky note may include any suitable data depending on the application, including, for

example, an uploaded image, a photograph captured by a user, a web address for an image, a color, personalized notes, online-accessible inspiration boards, inspiration board documents, and budgets. Additionally, one or more collaborators may be granted permission to read and/or write to the digital sticky note. The process flow 1000 includes connecting the digital sticky note to an application program interface (API) at 1006. The process flow 1000 includes providing access to the digital sticky note to a third party at 1008. It should be appreciated that the steps of the process flow 1000 illustrated in FIG. 10 may be performed in any suitable order, and that various steps may be performed a plurality of times.

[0097] The process of placing the digital sticky note at 1002 includes receiving an input from a user, rules-based algorithm, and/or neural network. In an embodiment, a user manually creates a digital sticky note and places the digital sticky on a specific location within an interactive rendering, an image, a media file, and/or attaches the digital sticky note to a geographic location. The location where the digital sticky note is placed may be referred to herein as a "coordinate location" for the digital sticky note. The coordinate location may comprise x, y, z coordinates for where the digital sticky note should be displayed within the media file, such as an interactive rendering, image, video stream, and so forth. In an embodiment, a rules-based algorithm or neural network provides a suggested coordinate location for where a digital sticky note should be placed based on the type of document, the type of project, the preferences of a user, and so forth. The digital sticky note may be secured to a digital document, such as an image, a video stream, an interactive rendering of a space, a hand-drawn diagram of a space, a computer-rendered diagram of a space, and so forth. The digital sticky note may be visible or hidden within the document.

[0098] In an example implementation, a three-dimensional, interactive rendering of a user's kitchen is created by the rendering server 110. The rendering platform 102 provides a means for a user to create a digital sticky note and place the digital sticky note on a structure, fixture, surface, or other portion of the three-dimensional rendering. In an example, the user creates a digital sticky note and places it on the cabinets of the three-dimensional rendering. The user may add information to the digital sticky note to indicate the user's preferences for the cabinets. Additionally, the user may include information indicating how the three-dimensional rendering should be modified to include certain cabinet configurations, colors, textures, and so forth.

[0099] The process of adding data to the digital sticky note at 1004 includes receiving information manually input by a user and/or receiving information suggested by a neural network or other algorithm. In an embodiment, a user creates a digital sticky note, places the digital sticky note, and adds information to be stored in connection with the digital sticky note. Information may include, for example, an image uploaded by the user, an image captured by the user in real-time with a mobile computing device, a web address to an image or other media, a video stream, and so forth. Additionally, the user may select a color using an eyedropper technique, may manually input a universal code for a color, may select a color off a color wheel, and so forth. The user may indicate that the location where the digital sticky note is placed should be flood filled with the selected color. Additionally, the user may include personalized notes indicating, for example, the user's through process, preferences, budgets, ideas, and so forth applicable to the location of the digital sticky note. Additionally, the user may associate the digital sticky note with one or more inspiration boards, including web-based inspiration boards and hard-copy documents of inspiration boards. Additionally, the user may grant read and/or write access to one or more collaborators to also interact with the digital sticky note. Additionally, the user may designate a budget for the structures, items, or fixtures associated with the digital sticky note. The user may designate an upper budget limit, an amount of money and/or time that has already been spent, a budget range, a detailed budget for different aspects of a project, and so forth.

[0100] In an embodiment, the process of adding data to the digital sticky note at 1004 includes receiving suggested data from a neural network or other algorithm. In an embodiment, a neural network is trained to predict a user's taste based on one or more of the user's manually-input preference, the user's inspiration boards, the user's purchase history, the user's search history, the user's "favorite" history, the user's location, the user's demographics, and so forth. The neural network may be trained to generate suggested designs, products, colors, and textures based on the user's preferences.

[0101] For example, a user may attach the digital sticky note to a shower space within a rendering of a residential bathroom. The neural network may suggest tile, plumbing hardware, textiles, and so forth that the user may like to use in the shower space of the bathroom. Additionally, the neural network may suggest products that could be purchased by the user, contractors in the user's area that could assist in the project, and so forth. The user may accept or deny the suggestions presented by the neural network. The user's choices to accept or deny the neural network's suggestions are fed back into the neural network and used to further define the user's preferences for future projects and suggestions.

[0102] In an embodiment, the digital sticky note is connected to one or more APIs. In an example implementation, the digital sticky note is connected to a payment portal API that permits a user to directly purchase a product through the digital sticky note. The digital sticky note may be connected to an API that enables a user to send and receive data elements to the rendering server 110 or other parties. In an embodiment, the digital sticky note represents a unique coded "ecosystem." When an API is connected to the digital sticky note, the API represents an open lane of communication between the digital sticky note and other systems. In an embodiment, the API enables the transmission and receipt of data between the digital sticky note and external or internal data visualization dashboards. In an embodiment, the API enables the transmission and receipt of payment data with an outside party.

[0103] The process of providing access to the digital sticky note to a third party at 1008 includes connecting the digital sticky note with an API comprising a payment processing platform. In an example implementation, the digital sticky note comprises an API for providing immediate shopping and payment processing capabilities without vendors, retailers, manufacturers, and other entities.

[0104] FIG. 11 illustrates two exemplary configurations for a digital sticky note. It should be appreciated that the digital sticky note may take any suitable form, and that the forms illustrated in FIG. 11 are only exemplary. The digital sticky note may include any image, icon, text, or other digital indication that can be placed on a digital document such as an image, interactive model or rendering, video stream, and so forth. The digital sticky note may be hidden or may remain visible at all times. In an embodiment, a user may tap, click, or otherwise select the digital sticky note to then see additional information stored in connection with the digital sticky note.

[0105] FIG. 12 is a screenshot of an exemplary sticky note portal. The sticky note portal may be visible to a user in response to a user clicking, tapping, creating, or otherwise selecting a digital sticky note. The portal enables a user to take a picture in real-time to add to the digital sticky note, to upload an image, to connect the digital sticky note with the web address of an image, and so forth. The user may additionally attach a name, description, background information, budget, and so forth to the digital sticky note.

[0106] FIGS. 13A-13D illustrate exemplary implementations of a digital sticky note 1302 on different files. FIG. 13A illustrates an example where a digital sticky note 1302 has been adhered to a bedroom within a two-dimensional computer-generated drawing of an architectural floorplan. FIG. 13B illustrates an example where a digital sticky note 1302 has been adhered to cabinets within a kitchen of a hand-drawn drawing of an architectural floorplan. FIG. 13C illustrates an example where a digital sticky note 1302 has been adhered to cabinets and appliances within a kitchen of

a three-dimensional rendering of an architectural floorplan. FIG. 13D illustrates an example where a digital sticky note 1302 has been adhered to an appliance of a kitchen in an image of the kitchen. The digital sticky note 1302 may be adhered to any suitable location within a media file. Multiple digital sticky notes 1302 may be adhered to different locations within the media file and/or at the same location within the media file.

[0107] In the example illustrated in FIG. 13A, a user may upload the two-dimensional drawing of the user's floorplan and adhere a digital sticky note to the bedroom in the floorplan. The user may then use the digital sticky note 1302 to store inspiration boards, inspiration images, potential products, budgets, and so forth applicable to the user's bedroom.

[0108] In the example illustrated in FIG. 13B, a user may upload the hand-drawn drawing of the user's floorplan and adhere a digital sticky note 1302 to the cabinets in the user's kitchen. The user may use the digital sticky note 1302 to store information about the user's kitchen cabinets or overall kitchen design.

[0109] In an embodiment, the digital sticky note includes geographic location information. The digital sticky note may be used by persons to store information in connection with a geographic location. For example, a person may create a digital sticky note at a certain geographic location, capture an image at that geographic location, and send the digital sticky note to a government agency or other entity. This could be used to identify a problem or broken infrastructure at the geographic location, an activity that occurred at the geographic location, an event occurring at the geographic location, and so forth. A collection of digital sticky notes for a corporation, religious organization, government organization, city, town, group of persons, or other entity may be stored such that all digital sticky notes include geographic information in connection with additional images, text, or other data. In an embodiment, a map displays digital sticky notes that have been created across a geographic region, and each of the digital sticky notes may be opened to receive more information about the event, accident, issue, structures, and so forth located at the geographic location.

[0110] It should be appreciated that the digital sticky note 1302 has many uses outside of architectural design, remodeling design, and/or landscaping design. The exemplary implementations discussed herein are non-limiting, and the digital sticky note 1302 is applicable to any suitable implementation.

[0111] Referring now to FIG. 14, a block diagram of an example computing device 1400 is illustrated. Computing device 1400 may be used to perform various procedures, such as those discussed herein. Computing device 1400 can perform various monitoring functions as discussed herein, and can execute one or more application programs, such as the application programs or functionality described herein. Computing device 1400 can be any of a wide variety of computing devices, such as a desktop computer, in-dash computer, vehicle control system, a notebook computer, a server computer, a handheld computer, tablet computer and the like.

[0112] Computing device 1400 includes one or more processor(s) 1412, one or more memory device(s) 1404, one or more interface(s) 1406, one or more mass storage device(s) 1408, one or more Input/output (I/O) device(s) 1410, and a display device 1430 all of which are coupled to a bus 1412. Processor(s) 1412 include one or more processors or controllers that execute instructions stored in memory device(s) 1404 and/or mass storage device(s) 1408. Processor(s) 1412 may also include various types of computer-readable media, such as cache memory.

[0113] Memory device(s) 1404 include various computer-readable media, such as volatile memory (e.g., random access memory (RAM) 1414) and/or nonvolatile memory (e.g., read-only memory (ROM) 1416). Memory device(s) 1404 may also include rewritable ROM, such as Flash memory.

[0114] Mass storage device(s) 1408 include various computer readable media, such as magnetic tapes, magnetic disks, optical disks, solid-state memory (e.g., Flash memory), and so forth. As shown in FIG. 14, a particular mass

storage device 1408 is a hard disk drive 1424. Various drives may also be included in mass storage device(s) 1408 to enable reading from and/or writing to the various computer readable media. Mass storage device(s) 1408 include removable media 1426 and/or non-removable media.

[0115] I/O device(s) 1410 include various devices that allow data and/or other information to be input to or retrieved from computing device 1400. Example I/O device(s) 1410 include cursor control devices, keyboards, keypads, microphones, monitors, touchscreen devices, or other display devices, speakers, printers, network interface cards, modems, and the like.

[0116] Display device 1430 includes any type of device capable of displaying information to one or more users of computing device 1400. Examples of display device 1430 include a monitor, display terminal, video projection device, and the like.

[0117] Interface(s) 1406 include various interfaces that allow computing device 1400 to interact with other systems, devices, or computing environments. Example interface(s) 1406 may include any number of different network interfaces 1420, such as interfaces to local area networks (LANs), wide area networks (WANs), wireless networks, and the Internet. Other interface(s) include user interface 1418 and peripheral device interface 1422. The interface(s) 1406 may also include one or more user interface elements 1418. The interface(s) 1406 may also include one or more peripheral interfaces such as interfaces for printers, pointing devices (mice, track pad, or any suitable user interface now known to those of ordinary skill in the field, or later discovered), keyboards, and the like.

[0118] Bus 1412 allows processor(s) 1412, memory device(s) 1404, interface(s) 1406, mass storage device(s) 1408, and I/O device(s) 1410 to communicate with one another, as well as other devices or components coupled to bus 1412. Bus 1412 represents one or more of several types of bus structures, such as a system bus, PCI bus, IEEE bus, USB bus, and so forth.

[0119] For purposes of illustration, programs and other executable program components are shown herein as discrete blocks, although it is understood that such programs and components may reside at various times in different storage components of computing device 1800 and are executed by processor(s) 1412. Alternatively, the systems and procedures described herein can be implemented in hardware, or a combination of hardware, software, and/or firmware. For example, one or more application specific integrated circuits (ASICs) can be programmed to carry out one or more of the systems and procedures described herein. As used herein, the terms “module” or “component” are intended to convey the implementation apparatus for accomplishing a process, such as by hardware, or a combination of hardware, software, and/or firmware, for the purposes of performing all or parts of operations disclosed herein. The terms “module” or “component” are intended to convey independent in how the modules, components, or their functionality or hardware may be implemented in different embodiments.

[0120] Various techniques, or certain aspects or portions thereof, may take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, a non-transitory computer readable storage medium, or any other machine-readable storage medium wherein, when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the various techniques. In the case of program code execution on programmable computers, the computing device may include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. The volatile and non-volatile memory and/or storage elements may be a RAM, an EPROM, a flash drive, an optical drive, a magnetic hard drive, or another medium for storing electronic data. One or more programs that may implement or utilize the various techniques described herein

may use an application programming interface (API), reusable controls, and the like. Such programs may be implemented in a high-level procedural, functional, object-oriented programming language to communicate with a computer system. However, the program(s) may be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language, and combined with hardware implementations.

[0121] It should be understood that many of the functional units described in this specification may be implemented as one or more components or modules, which are terms used to emphasize their implementation independence more particularly. For example, a component or module may be implemented as a hardware circuit comprising custom very large-scale integration (VLSI) circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A component may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices, or the like.

[0122] Components may also be implemented in software for execution by various types of processors. An identified component of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions, which may, for instance, be organized as an object, a procedure, or a function. Nevertheless, the executables of an identified component need not be physically located together but may comprise disparate instructions stored in different locations that, when joined logically together, comprise the component and achieve the stated purpose for the component.

[0123] Indeed, a component of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within components and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network. The components may be passive or active, including agents operable to perform desired functions.

Examples

[0124] The following examples pertain to further embodiments.

[0125] Example 1 is a method. The method includes determining an input image comprising one or more lines and generating a bitmap based on the input image. The method includes performing edge detection on the bitmap to identify the one or more lines in the input image by applying a growth algorithm to pixels of the bitmap. The method includes calculating a pixel width of each of the one or more lines in the input image.

[0126] Example 2 is a method as in Example 1, wherein calculating the pixel width of each of the one or more lines in the input image comprises identifying a peak for each of the one or more lines, wherein a peak of a line represents one or two center-most pixels of the line.

[0127] Example 3 is a method as in any of Examples 1-2, further comprising: identifying a peak for each of the one or more lines, wherein a peak of a line represents one or two center-most pixels of the line; identifying a peak starting point for the line; following the peak of the line in a first direction beginning at the peak starting point and terminating at a first ending point of the line; and following the peak of the line in a second direction beginning at the peak starting point and terminating at a second ending point of the line.

[0128] Example 4 is a method as in any of Examples 1-3, wherein calculating the pixel width of each of the one or more lines comprises identifying a peak of each of the one or more lines and applying a decay algorithm to each of the one or more lines, wherein the decay algorithm begins at the peak of each of the one or more lines.

[0129] Example 5 is a method as in any of Examples 1-4, wherein the input image comprises a two-dimensional drawing of an architectural floorplan, and wherein the method further comprises identifying walls in the architectural floorplan based at least in part on the pixel width of each of the one or more lines.

[0130] Example 6 is a method as in any of Examples 1-5, further comprising: applying a Z-axis height to the walls in the architectural floorplan; and generating serialized data comprising a geometric vector diagram of the architectural floorplan, wherein the geometric vector diagram can be used to generate a three-dimensional rendering of the architectural floorplan illustrated in the input image.

[0131] Example 7 is a method as in any of Examples 1-6, wherein the input image comprises a two-dimensional drawing of an architectural floorplan, and wherein the method further comprises identifying enclosed regions within the architectural floorplan.

[0132] Example 8 is a method as in any of Examples 1-7, wherein identifying the enclosed regions within the architectural floorplan comprises: identifying a first pixel comprising an empty value and filling in the first pixel with a color; applying a flood fill to an enclosed region comprising the first pixel by identifying neighboring pixels to the first pixel and filling in the neighboring pixels with the color; recursively applying the flood fill to additional neighboring pixels within the enclosed region; and terminating the flood fill for the enclosed region when perimeter pixels comprising non-empty values are reached by the flood fill.

[0133] Example 9 is a method as in any of Examples 1-8, further comprising identifying a plurality of enclosed regions and applying a different color to each of the plurality of enclosed regions.

[0134] Example 10 is a method as in any of Examples 1-9, further comprising applying a stencil buffer to the bitmap, and wherein performing edge detection on the bitmap comprises performing edge detection on the bitmap after the stencil buffer has been applied.

[0135] Example 11 is a method. The method includes receiving an image comprising an object and receiving an indication of a color. The method includes identifying the object within the image and defining a texture of the object. The method includes selecting a stored texture file stored in a database based on the texture of the object. The method includes merging the stored texture file and the color to generate a color and texture placement that can be implemented in a rendered scene.

[0136] Example 12 is a method as in Example 11, further comprising: generating a three-dimensional model of a scene; identifying an object within the three-dimensional on which to apply the color and texture placement; and applying the color and texture placement to the object within the three-dimensional model.

[0137] Example 13 is a method as in any of Examples 11-12, wherein the three-dimensional model of the scene comprises an interactive three-dimensional rendering of an architectural floorplan, and wherein the object within the three-dimensional model comprises one or more of a construction material, a design material, a plumbing fixture, a lighting fixture, a furnishing, a cabinet, a countertop, a backsplash, flooring, or a wall.

[0138] Example 14 is a method as in any of Examples 11-13, wherein the database comprises a plurality of texture files, and wherein each of the plurality of texture files comprise texture information for rendering a texture on a three-dimensional model, wherein the plurality of texture files comprises texture information for one or more of: a construction material, a design material, a plumbing fixture, a lighting fixture, a furnishing, a cabinet, a countertop, a backsplash, flooring, or a wall.

[0139] Example 15 is a method as in any of Examples 11-14, further comprising receiving an indication of an identity of the object within the image, and wherein identifying the object within the image comprises: providing the

image to a first neural network trained to locate the object within the image; and receiving an indication from the first neural network comprising a location of the object within the image.

[0140] Example 16 is a method as in any of Examples 11-15, wherein identifying the object within the image further comprises: providing a subsection of the image comprising the object to a second neural network trained to determine an identity of the object; and receiving an indication from the second neural network comprising the identity of the object; wherein the method further comprises determining whether a texture file comprising a same or similar identity as the identity of the object is stored within the database.

[0141] Example 17 is a method as in any of Examples 11-16, further comprising: determining an approximate size of the object within the image; applying the approximate size of the object to the color and texture placement; and providing a means for a user to adjust a scale of the color and texture placement within the rendered scene such that a relative size of the color and texture placement is adjusted relative to other objects within the rendered scene.

[0142] Example 18 is a method as in any of Examples 11-17, further comprising: generating a three-dimensional model of a scene; receiving an indication from a user to adhere a digital sticky note at a location within the three-dimensional model; and generating the digital sticky note and applying the digital sticky note to the location within the three-dimensional model.

[0143] Example 19 is a method as in any of Examples 11-18, further comprising storing information in connection with the digital sticky note, wherein the information comprises one or more of: an image, a video stream, a web address to an image or website, text, a mathematical representation, a texture file, a graph, an illustration, a hyperlink, an inspiration board, or a budget.

[0144] Example 20 is a method as in any of Examples 11-19, wherein: receiving the image comprising the object comprises receiving the image from the user in connection with the digital sticky note; receiving the indication of the color comprises receiving the indication of the color from the user in connection with the digital sticky note; and storing the information in connection with the digital sticky note comprises storing the image comprising the object and the indication of the color in connection with the digital sticky note.

[0145] Example 21 is a method as in any of Examples 11-20, wherein the method further comprises: determining an identity of a rendered object at the location within the three-dimensional model; and applying the color and texture placement to the rendered object; wherein the rendered object comprises one or more of a construction material, a design material, a plumbing fixture, a lighting fixture, a furnishing, a cabinet, a countertop, a backsplash, flooring, or a wall.

[0146] Example 22 is a method. The method includes generating a digital sticky note to be stored in connection with a file. The method includes storing a coordinate location in connection with the digital sticky note, wherein the coordinate location indicates where the digital sticky note should be displayed within the file. The method includes aggregating data to be stored in connection with the digital sticky note, wherein the data comprises information applicable to the coordinate location.

[0147] Example 23 is a method as in Example 22, further comprising receiving an indication to generate the digital sticky note at the coordinate location, wherein the indication is received from one or more of: a user providing a computer-implemented instruction; or a neural network trained to recommend one or more coordinate locations within the file for creating a digital sticky note.

[0148] Example 24 is a method as in any of Examples 22-23, wherein aggregating the data comprises aggregating one or more types of data selected from a group comprising: an image, a video stream, a hyperlink, media available via a hyperlink, text, an inspiration board, a numerical entry, a geographic location coordinate, or a budget.

[0149] Example 25 is a method as in any of Examples 22-24, further comprising connecting the digital sticky note to an application program interface (API) such that a user may communicate with the API by way of the digital sticky note.

[0150] Example 26 is a method as in any of Examples 22-25, further comprising: receiving an image to be stored in connection with the digital sticky note at the coordinate location; receiving an indication that a texture object displayed in the image should be associated with the coordinate location; and providing the image to a neural network trained to determine an identity of the texture object displayed in the image.

[0151] Example 27 is a method as in any of Examples 22-26, wherein the file comprises a three-dimensional model of a space, and wherein the method further comprises: receiving from the neural network the identity of the texture object; retrieving from a database a texture file equivalent to or approximating the identity of the texture object displayed in the image; merging the texture file with a color indication to generate a color-texture placement; and applying the color-texture placement to the coordinate location within the three-dimensional model of the space.

[0152] Example 28 is a method as in any of Examples 22-27, wherein the file comprises a three-dimensional model of a space, and wherein the method further comprises providing the three-dimensional model of the space to a user such that the user can select the coordinate location by selecting an object within the three-dimensional model.

[0153] Example 29 is a method as in any of Examples 22-28, wherein the space comprises an architectural floorplan, and wherein the object within the three-dimensional model comprises one or more of: a wall, a door, a window, a cabinet, a countertop, a floor, a plumbing fixture, an electrical fixture, a furnishing, or a surface where a construction material and/or a design material can be affixed.

[0154] Example 30 is a method as in any of Examples 22-29, wherein the data to be stored in connection with the digital sticky note is applicable to the object within the three-dimensional model, and wherein the method further comprises: identifying a texture object within the data stored in connection with the digital sticky note, wherein the texture object comprises one or more of: a construction material, a design material, flooring, a plumbing fixture, an electrical fixture, or a furnishing; determining a color to be applied in connection with the texture object; retrieving from a database a texture file equivalent to or approximating the texture object; merging the texture file and the color to generate a color-texture placement; and applying the color-texture placement to the object within the three-dimensional model.

[0155] Example 31 is a method as in any of Examples 22-30, further comprising displaying the digital sticky note on a rendering of the file, wherein the digital sticky note comprises an icon that can be selected by a user to display the aggregated data.

[0156] Example 32 is a system comprising one or more processors for executing instructions stored in non-transitory computer readable storage media, wherein the instructions comprise any of the method steps of Examples 1-31.

[0157] Example 33 is non-transitory computer readable storage media storing instructions for execution by one or more processors, wherein the instructions comprise any of the method steps of Examples 1-31.

[0158] Reference throughout this specification to “an example” means that a particular feature, structure, or characteristic described in connection with the example is included in at least one embodiment of the present disclosure. Thus, appearances of the phrase “in an example” in various places throughout this specification are not necessarily all referring to the same embodiment.

[0159] As used herein, a plurality of items, structural elements, compositional elements, and/or materials may be presented in a common list for convenience. However, these lists should be construed as though each member of the list is individually identified as a separate and unique member. Thus, no individual member of such list should be construed as a de facto equivalent of any other member of the same list solely based on its presentation in a common group without indications to the contrary. In addition, various embodiments and examples of the present disclosure may be referred to herein along with alternatives for the various components thereof. It is understood that such embodiments, examples, and alternatives are not to be construed as de facto equivalents of one another but are to be considered as separate and autonomous representations of the present disclosure.

[0160] Although the foregoing has been described in some detail for purposes of clarity, it will be apparent that certain changes and modifications may be made without departing from the principles thereof. It should be noted that there are many alternative ways of implementing both the processes and apparatuses described herein. Accordingly, the present embodiments are to be considered illustrative and not restrictive.

[0161] Those having skill in the art will appreciate that many changes may be made to the details of the above-described embodiments without departing from the underlying principles of the disclosure.

CLAIMS

What is claimed is:

1. A method comprising:

determining an input image comprising one or more lines;

generating a bitmap based on the input image;

performing edge detection on the bitmap to identify the one or more lines in the input image by applying a growth algorithm to pixels of the bitmap; and

calculating a pixel width of each of the one or more lines in the input image.

2. The method of claim 1, wherein calculating the pixel width of each of the one or more lines in the input image comprises identifying a peak for each of the one or more lines, wherein a peak of a line represents one or two center-most pixels of the line.

3. The method of claim 1, further comprising:

identifying a peak for each of the one or more lines, wherein a peak of a line represents one or two center-most pixels of the line;

identifying a peak starting point for the line;

following the peak of the line in a first direction beginning at the peak starting point and terminating at a first ending point of the line; and

following the peak of the line in a second direction beginning at the peak starting point and terminating at a second ending point of the line.

4. The method of claim 1, wherein calculating the pixel width of each of the one or more lines comprises identifying a peak of each of the one or more lines and applying a decay algorithm to each of the one or more lines, wherein the decay algorithm begins at the peak of each of the one or more lines.

5. The method of claim 1, wherein the input image comprises a two-dimensional drawing of an architectural floorplan, and wherein the method further comprises identifying walls in the architectural floorplan based on one or more of the pixel width of each of the one or more lines, a color of pixels within the architectural floorplan, or regions defined within the architectural floorplan.

6. The method of claim 5, further comprising:

applying a Z-axis height to the walls in the architectural floorplan; and

generating serialized data comprising a geometric vector diagram of the architectural floorplan, wherein the geometric vector diagram can be used to generate a three-dimensional rendering of the architectural floorplan illustrated in the input image.

7. The method of claim 1, wherein the input image comprises a two-dimensional drawing of an architectural floorplan, and wherein the method further comprises identifying enclosed regions within the architectural floorplan.

8. The method of claim 7, wherein identifying the enclosed regions within the architectural floorplan comprises:

identifying a first pixel comprising an empty value and filling in the first pixel with a color;

applying a flood fill to an enclosed region comprising the first pixel by identifying neighboring pixels to the first pixel and filling in the neighboring pixels with the color;

recursively applying the flood fill to additional neighboring pixels within the enclosed region; and

terminating the flood fill for the enclosed region when perimeter pixels comprising non-empty values are reached by the flood fill.

9. The method of claim 8, further comprising identifying a plurality of enclosed regions and applying a different color to each of the plurality of enclosed regions.

10. The method of claim 1, further comprising applying a stencil buffer to the bitmap, and wherein performing edge detection on the bitmap comprises performing edge detection on the bitmap after the stencil buffer has been applied.

11. A system comprising one or more processors configurable to execute instructions stored in non-transitory computer readable storage media, the instructions comprising:

determining an input image comprising one or more lines;

generating a bitmap based on the input image;

performing edge detection on the bitmap to identify the one or more lines in the input image by applying a growth algorithm to pixels of the bitmap; and

calculating a pixel width of each of the one or more lines in the input image.

12. The system of claim 11, wherein the instructions are such that calculating the pixel width of each of the one or more lines in the input image comprises identifying a peak for each of the one or more lines, wherein a peak of a line represents one or two center-most pixels of the line.

13. The system of claim 11, wherein the instructions further comprise:

identifying a peak for each of the one or more lines, wherein a peak of a line represents one or two center-most pixels of the line;

identifying a peak starting point for the line;

following the peak of the line in a first direction beginning at the peak starting point and terminating at a first ending point of the line; and

following the peak of the line in a second direction beginning at the peak starting point and terminating at a second ending point of the line.

14. The system of claim 11, wherein the instructions are such that calculating the pixel width of each of the one or more lines comprises identifying a peak of each of the one or more lines and applying a decay algorithm to each of the one or more lines, wherein the decay algorithm begins at the peak of each of the one or more lines.

15. The system of claim 11, wherein the input image comprises a two-dimensional drawing of an architectural floorplan, and wherein the instructions further comprise identifying walls in the architectural floorplan based at least in part on the pixel width of each of the one or more lines.

16. Non-transitory computer readable storage media storing instructions for execution by one or more processors, the instructions comprising:

determining an input image comprising one or more lines;

generating a bitmap based on the input image;

performing edge detection on the bitmap to identify the one or more lines in the input image by applying a growth algorithm to pixels of the bitmap; and

calculating a pixel width of each of the one or more lines in the input image.

17. The non-transitory computer readable storage media of claim 16, wherein the input image comprises a two-dimensional drawing of an architectural floorplan, and wherein the instructions further comprise:

identifying walls in the architectural floorplan based at least in part on the pixel width of each of the one or more lines;

applying a Z-axis height to the walls in the architectural floorplan; and

generating serialized data comprising a geometric vector diagram of the architectural floorplan, wherein the geometric vector diagram can be used to generate a three-dimensional rendering of the architectural floorplan illustrated in the input image.

18. The non-transitory computer readable storage media of claim 16, wherein the input image comprises a two-dimensional drawing of an architectural floorplan, and wherein the instructions further comprise identifying enclosed regions within the architectural floorplan.

19. The non-transitory computer readable storage media of claim 18, wherein the instructions are such that identifying the enclosed regions within the architectural floorplan comprises:

identifying a first pixel comprising an empty value and filling in the first pixel with a color;

applying a flood fill to an enclosed region comprising the first pixel by identifying neighboring pixels to the first pixel and filling in the neighboring pixels with the color;

recursively applying the flood fill to additional neighboring pixels within the enclosed region; and

terminating the flood fill for the enclosed region when perimeter pixels comprising non-empty values are reached by the flood fill.

20. The non-transitory computer readable storage media of claim 16, wherein the instructions further comprise applying a stencil buffer to the bitmap, and wherein the instructions are such that performing edge detection on the bitmap comprises performing edge detection on the bitmap after the stencil buffer has been applied.

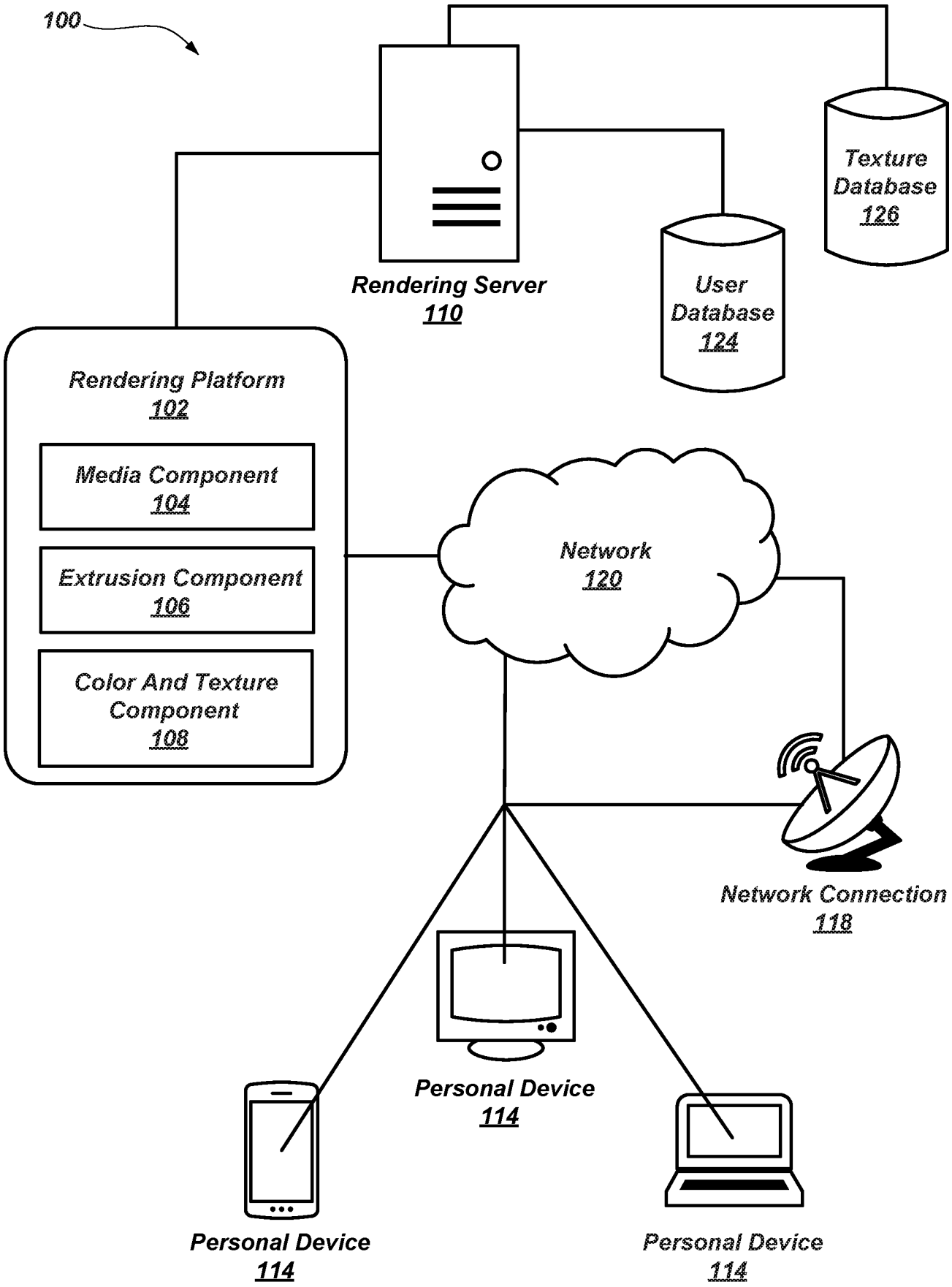
100
Rendering Server
110
Texture
Database
126
User
Database
124
Rendering Platform
102
Media Component
104
Extrusion Component
106
Color And Texture
Component
108
Network
120
Network Connection
118
Personal Device
114
Personal Device
114
Personal Device
114

FIG. 1

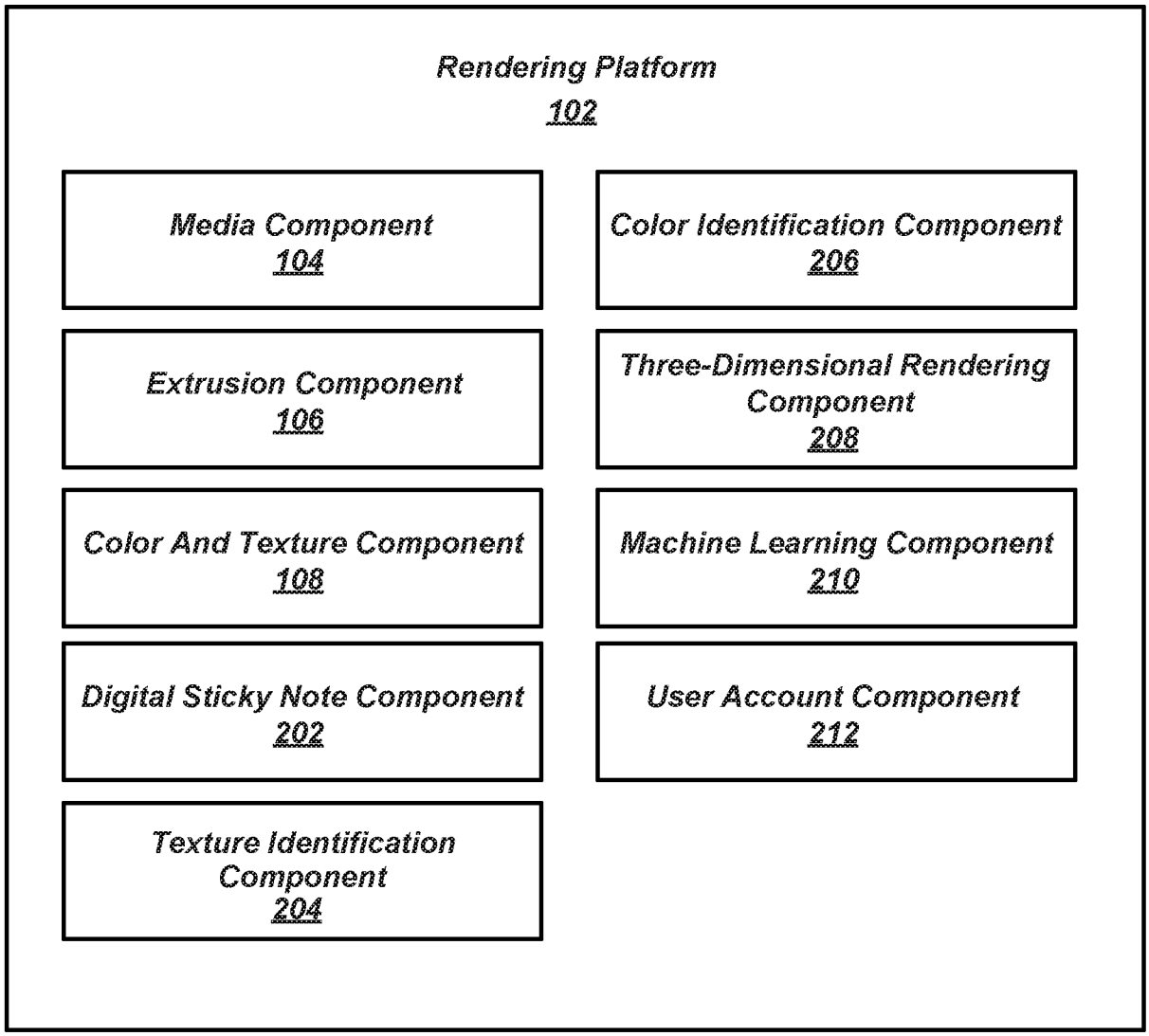
Rendering Platform
102
Media Component
104
Color Identification Component
206
Extrusion Component
106
Three-Dimensional Rendering Component
208
Color And Texture Component
108
Machine Learning Component
210
Digital Sticky Note Component
202
User Account Component
212
Texture Identification Component
204

FIG. 2

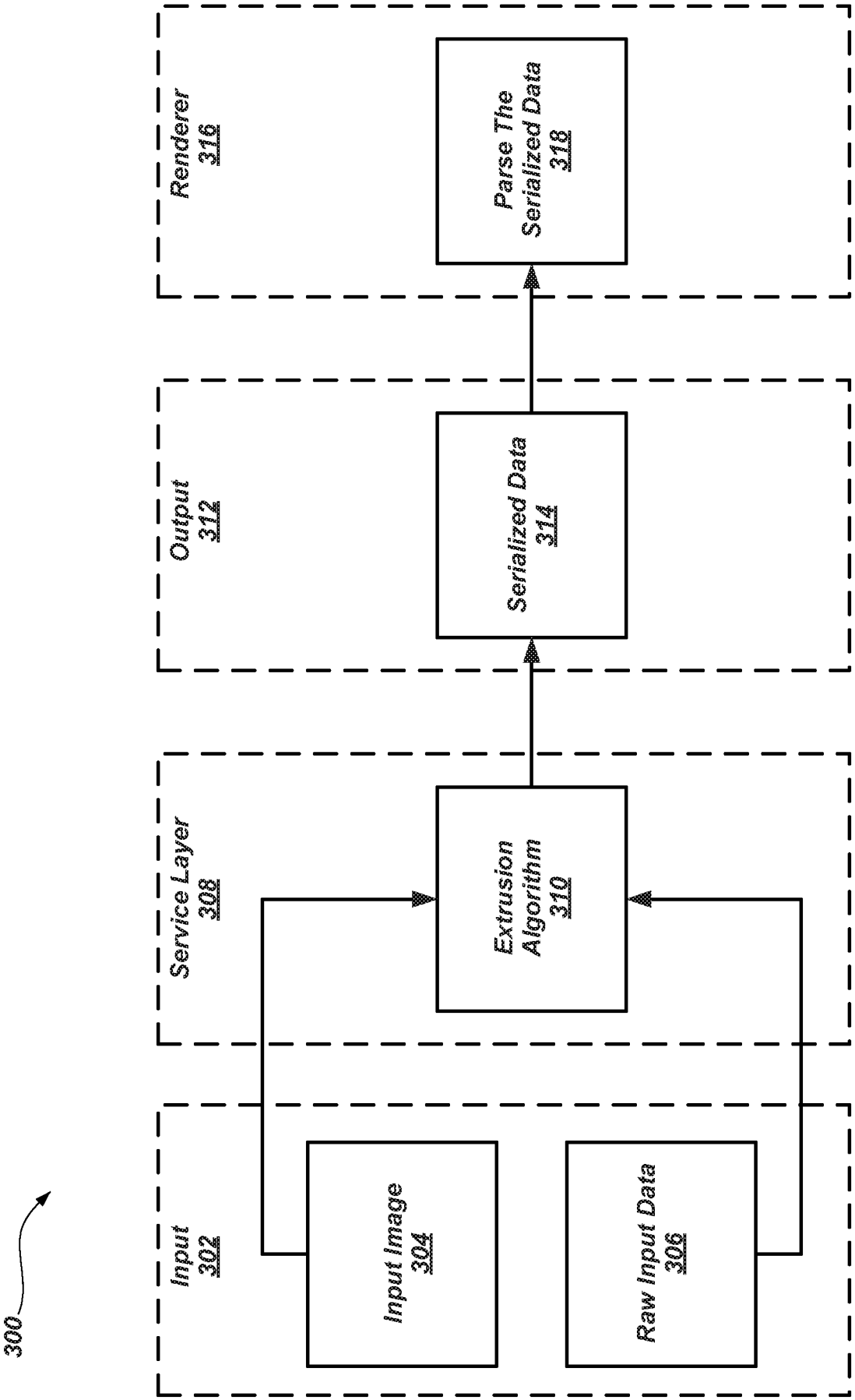
300
Input
302
Input Image
304
Raw Input Data
306
Service Layer
308
Extrusion Algorithm
310
Output
312
Serialized Data
314
Renderer
316
Parse The Serialized Data
318

FIG. 3

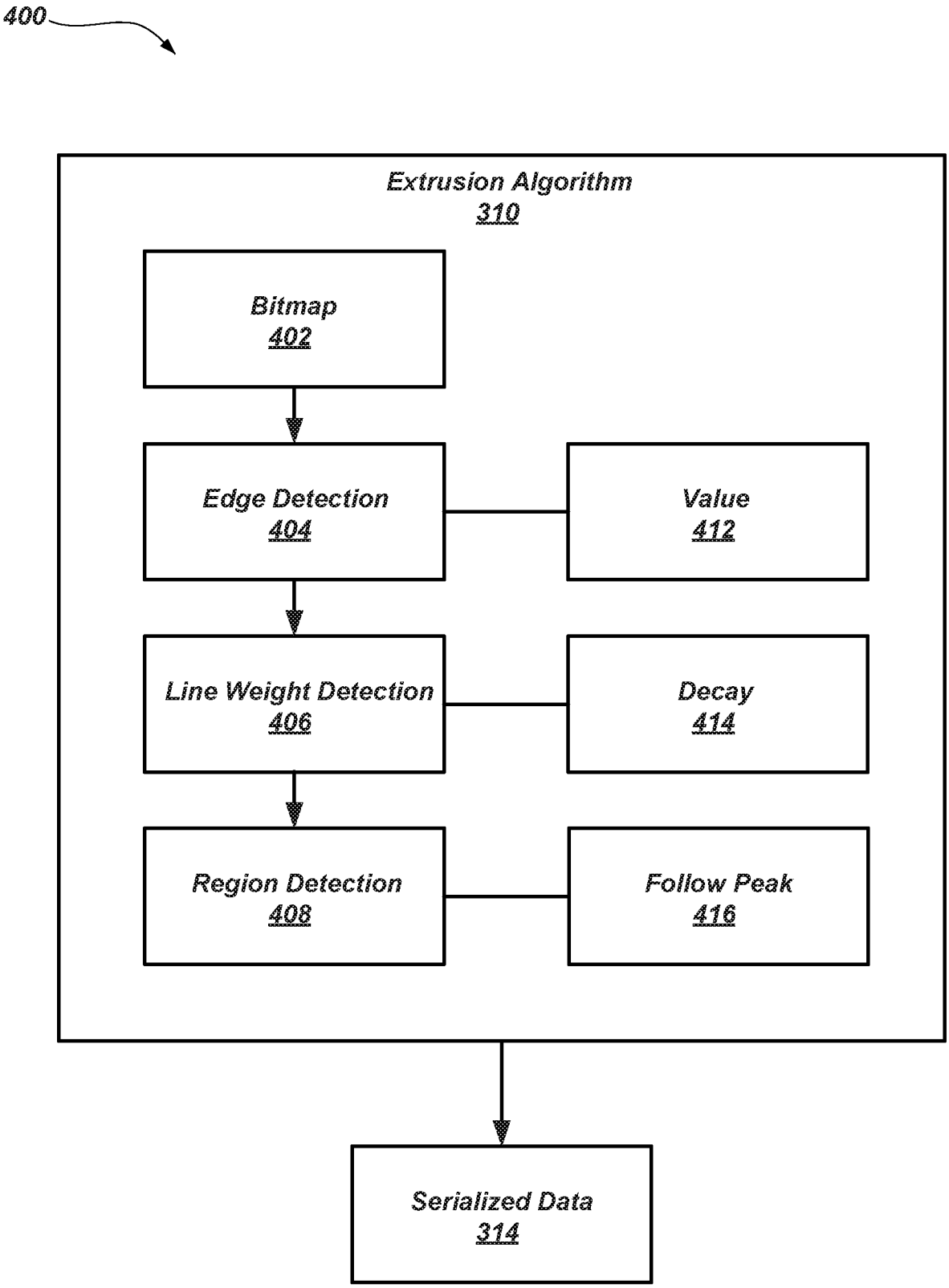
400
Extrusion Algorithm
310
Bitmap
402
Edge Detection
404
Value
412
Line Weight Detection
406
Decay
414
Region Detection
408
Follow Peak
416
Serialized Data
314

FIG. 4

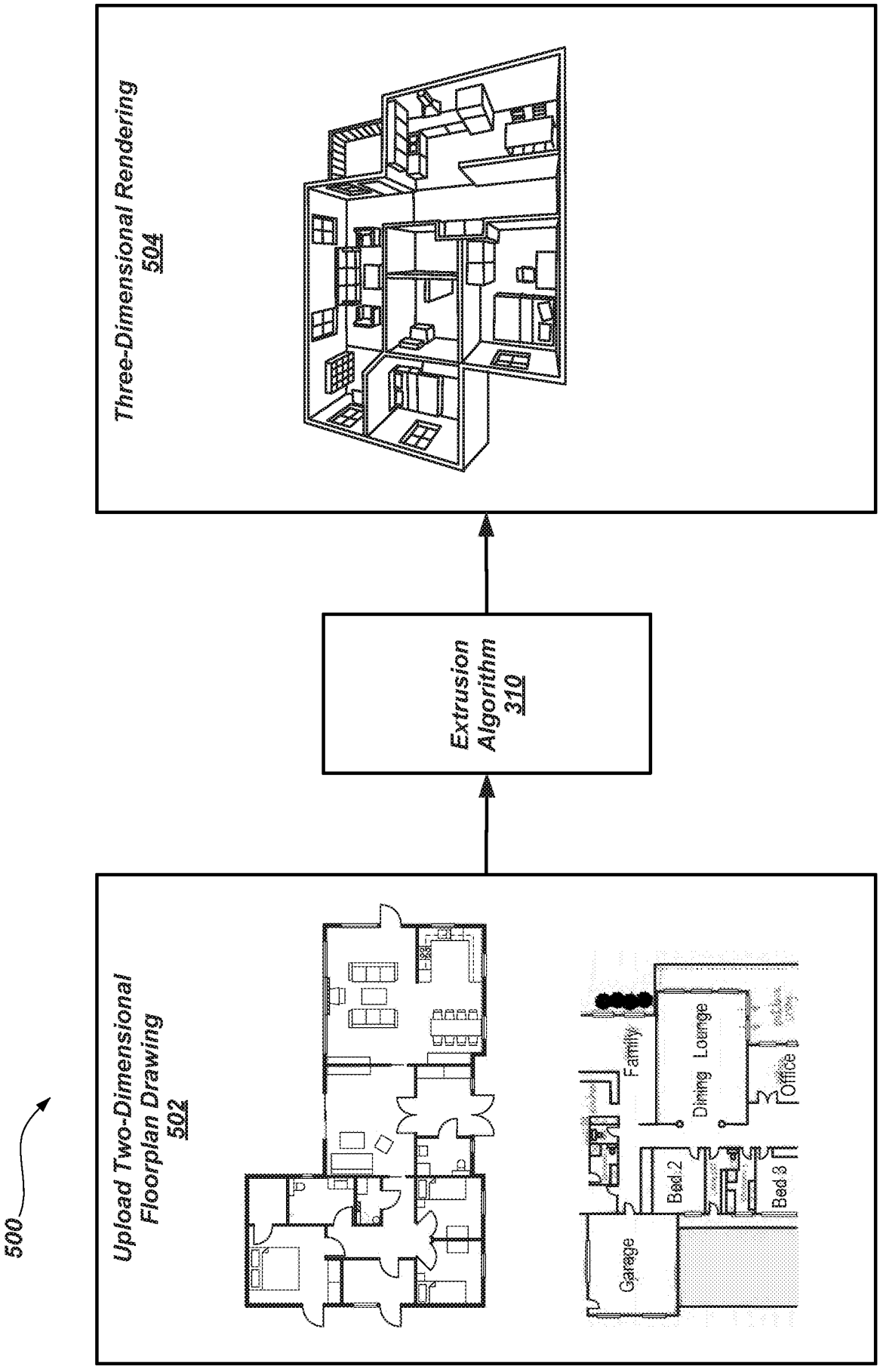
500
Upload Two-Dimensional Floorplan Drawing
502
Family
Dining
Lounge
Office
Bed 2
Bed 3
Garage
Extrusion Algorithm
310
Three-Dimensional Rendering
504

FIG. 5

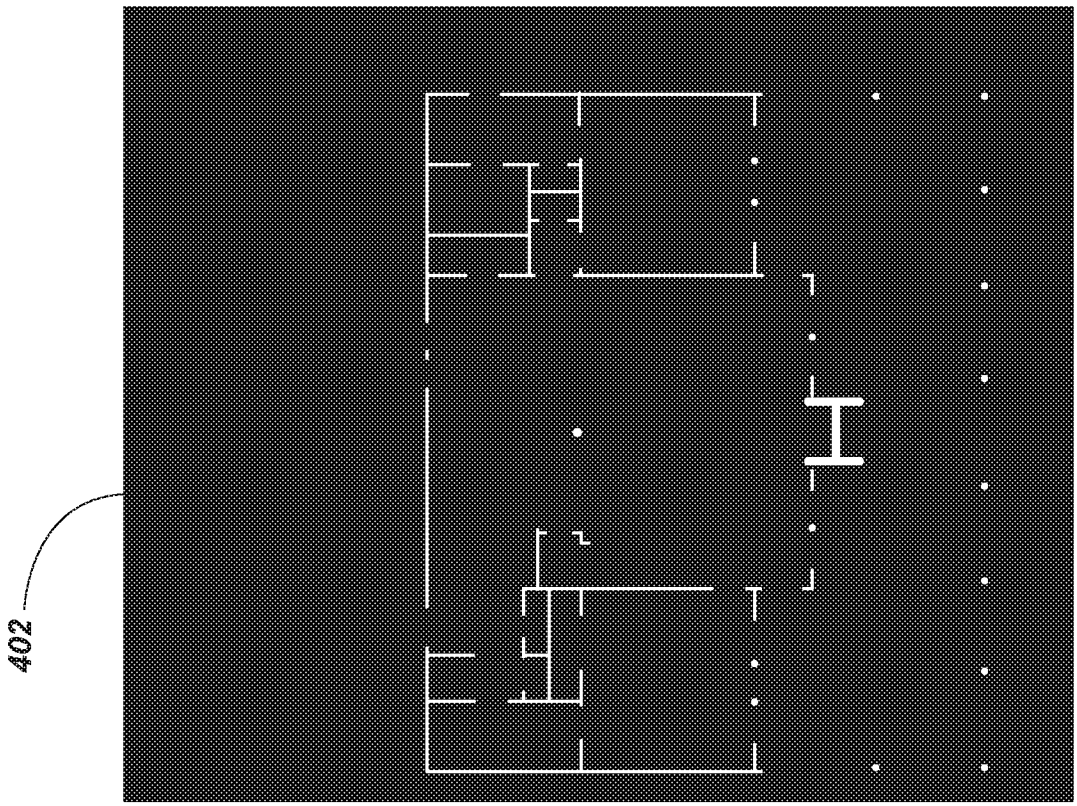
402

FIG. 6B

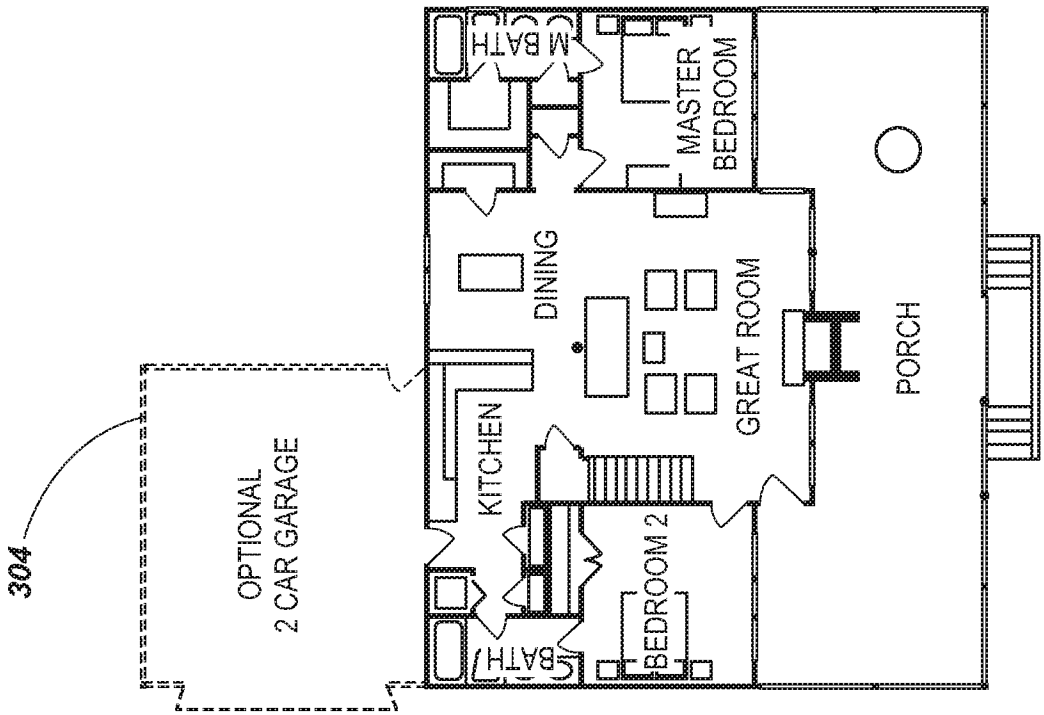
304
OPTIONAL
2 CAR GARAGE
KITCHEN
DINING
M BATH
MASTER
BEDROOM
GREAT ROOM
PORCH
BATH
BEDROOM 2

FIG. 6A

0	0	1	0
0	1	2	1
0	0	1	0
0	0	0	0

FIG. 7

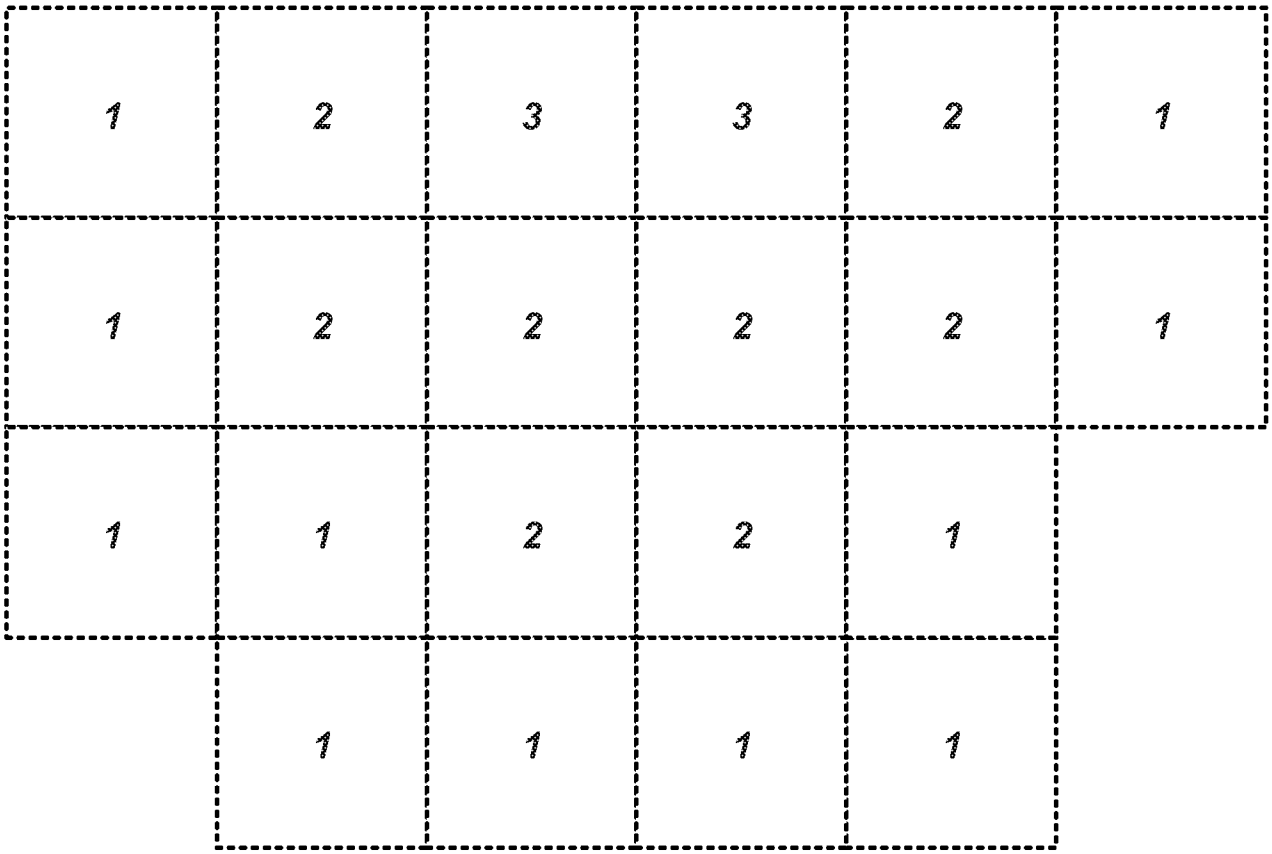

FIG. 8

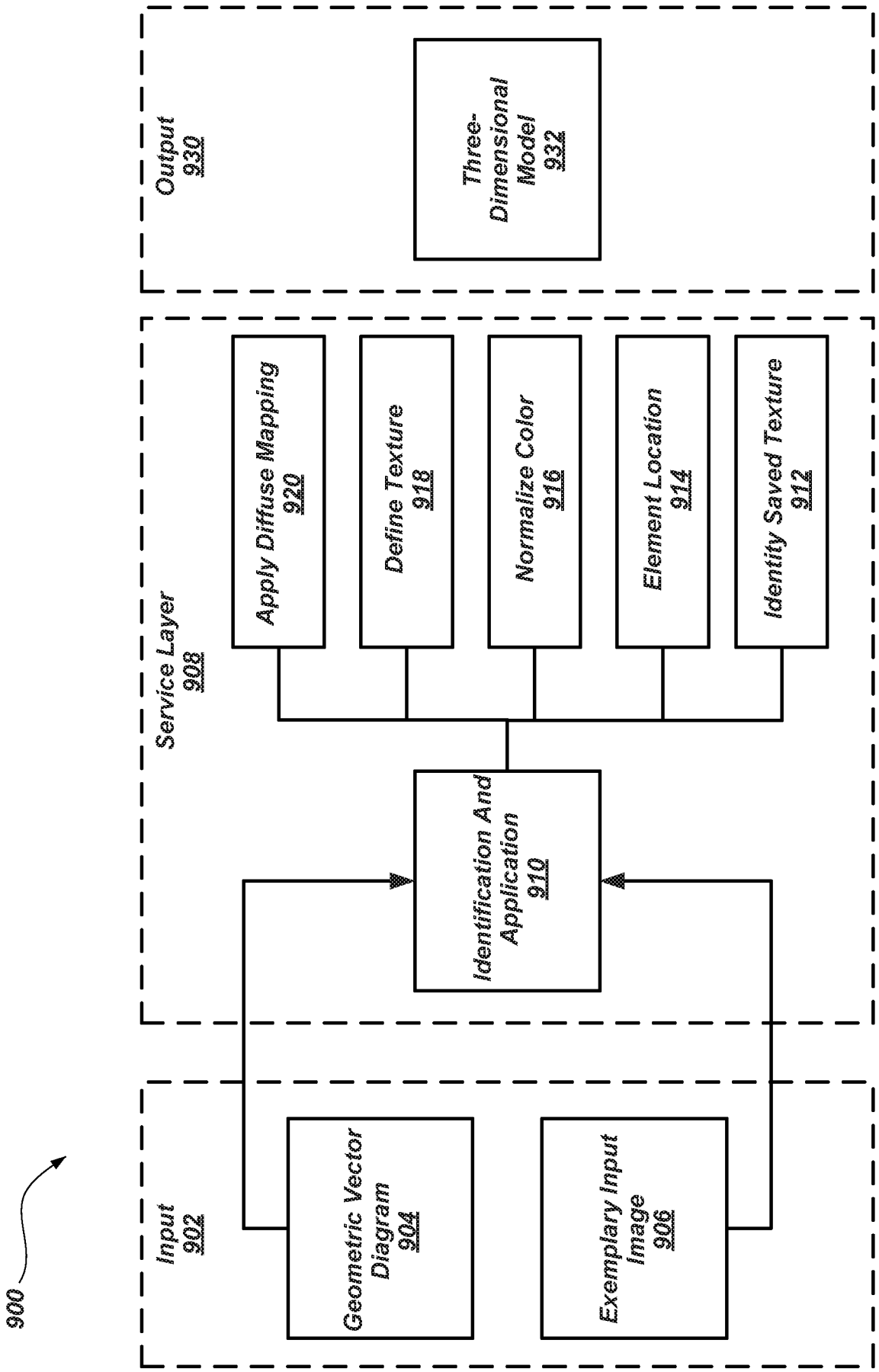
900
Input
902
Geometric Vector Diagram
904
Exemplary Input Image
906
Service Layer
908
Identification And Application
910
Identity Saved Texture
912
Element Location
914
Normalize Color
916
Define Texture
918
Apply Diffuse Mapping
920
Output
930
Three-Dimensional Model
932

FIG. 9

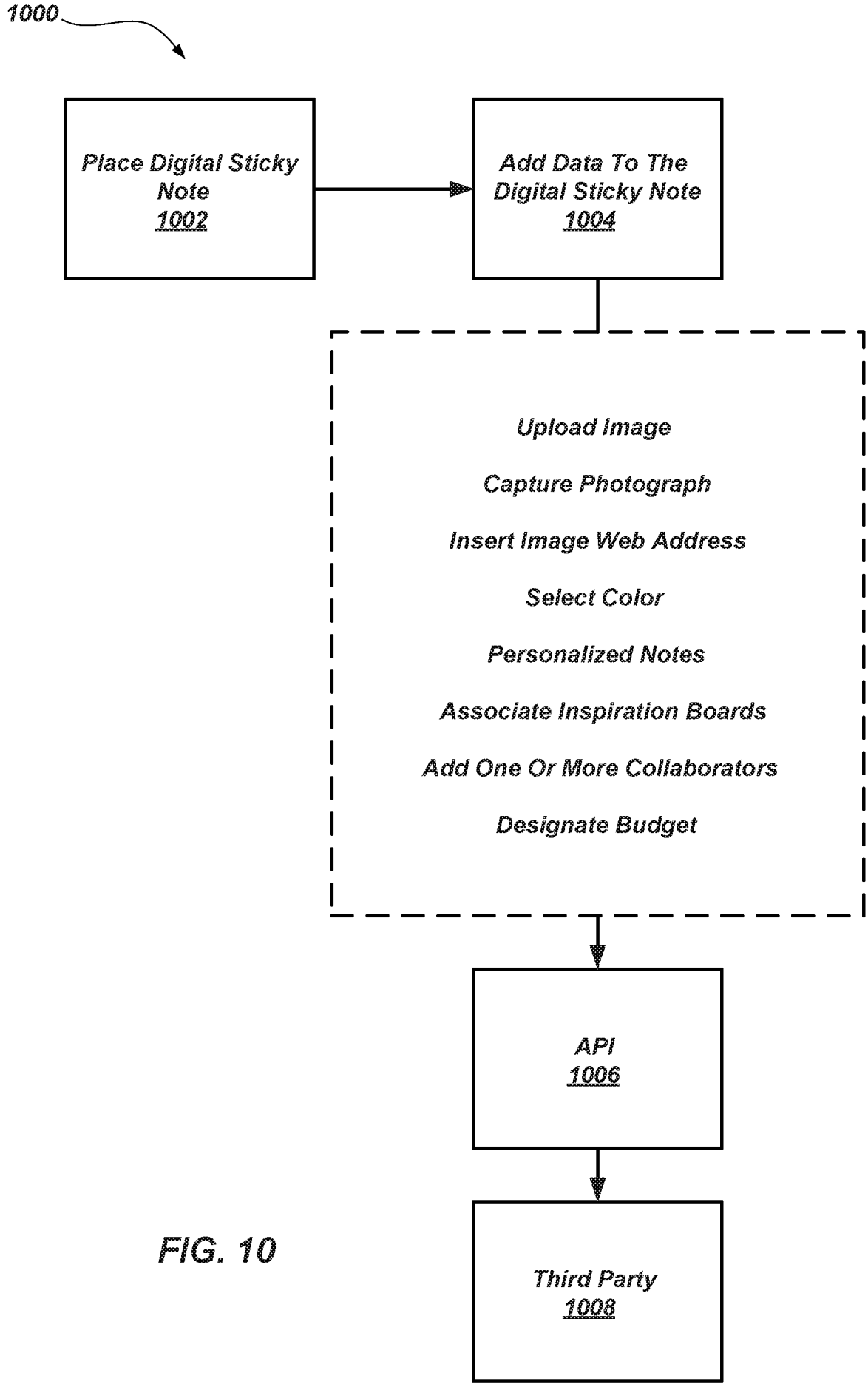
1000
Place Digital Sticky Note
1002
Add Data To The Digital Sticky Note
1004
Upload Image
Capture Photograph
Insert Image Web Address
Select Color
Personalized Notes
Associate Inspiration Boards
Add One Or More Collaborators
Designate Budget
API
1006
Third Party
1008

FIG. 10

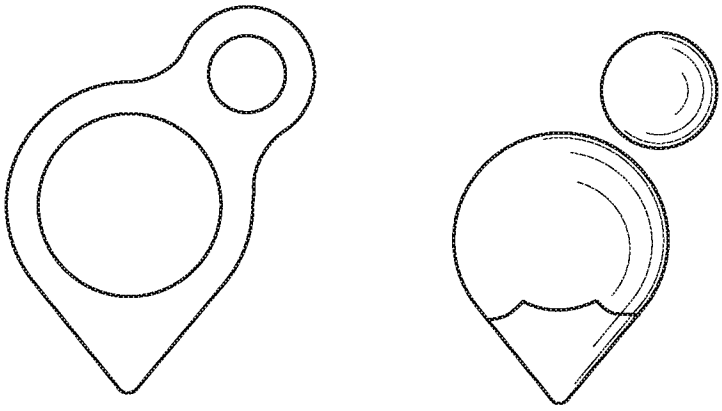

FIG. 11

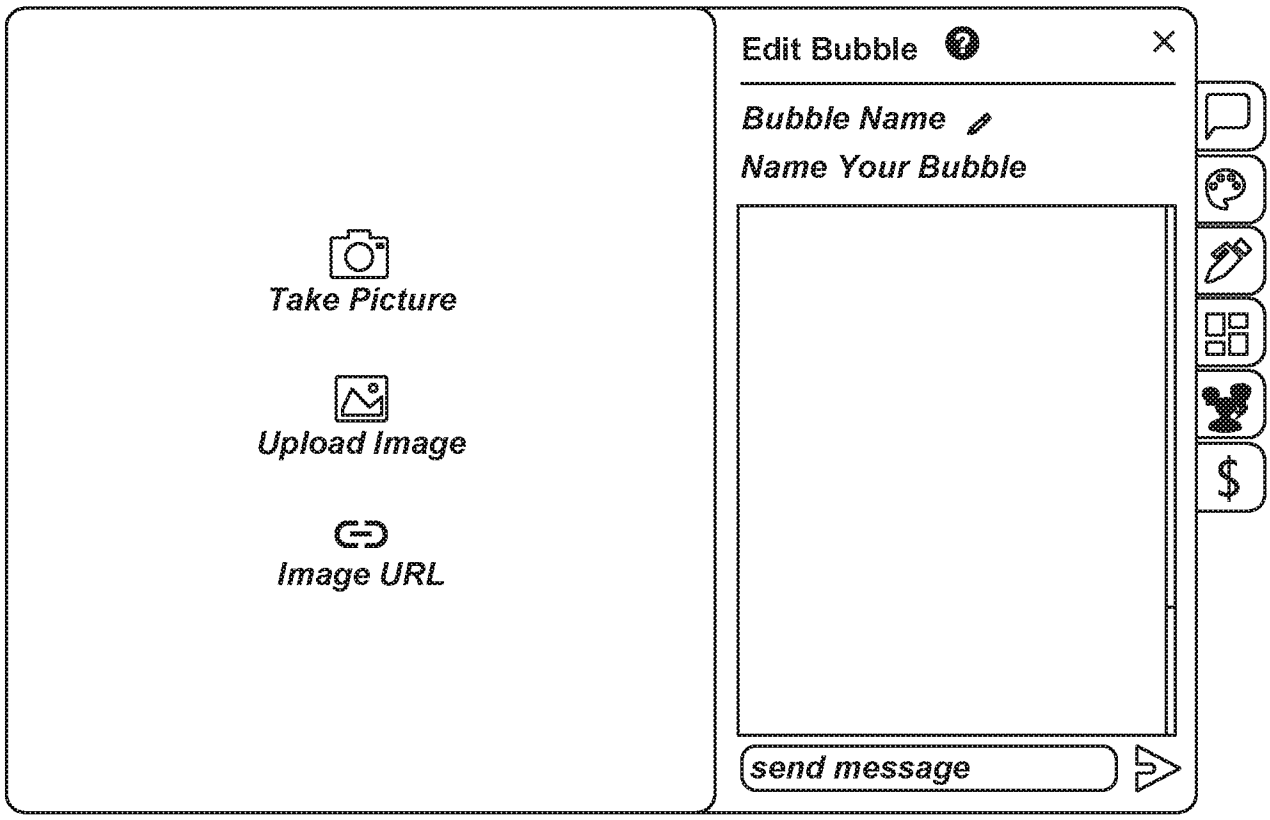
Edit Bubble
Bubble Name
Name Your Bubble
Take Picture
Upload Image
Image URL
send message
$

FIG. 12

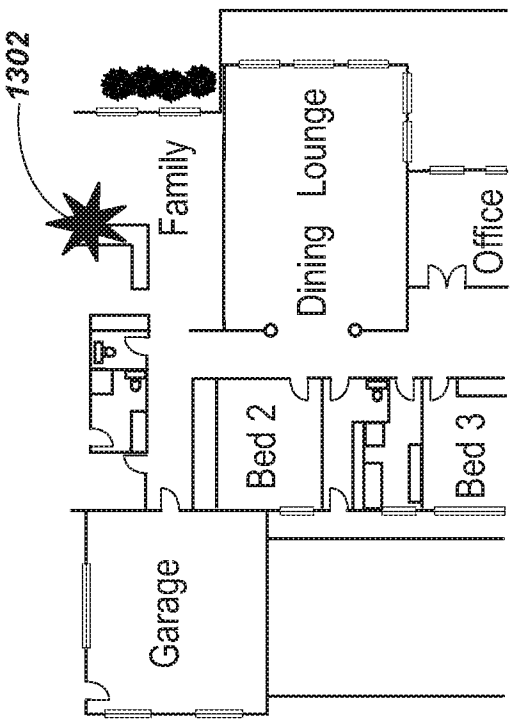

FIG. 13B

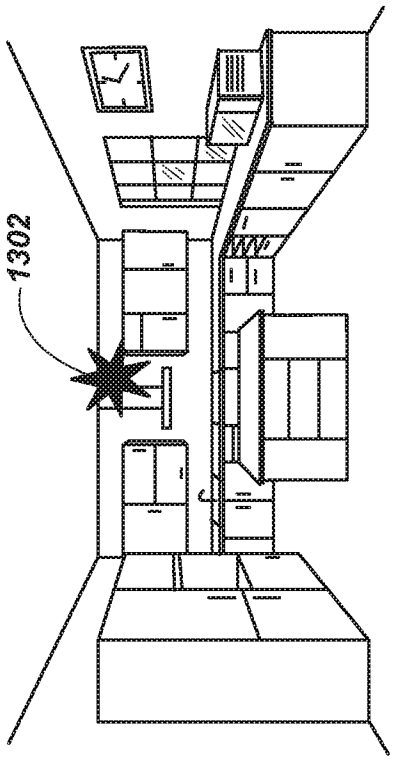

FIG. 13D

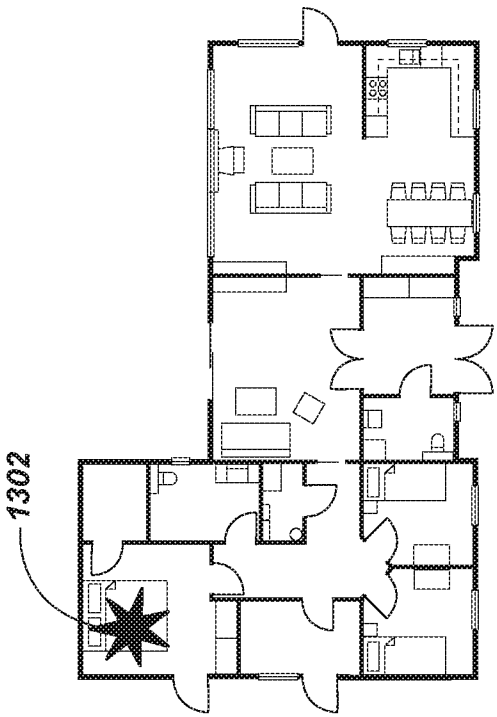

FIG. 13A

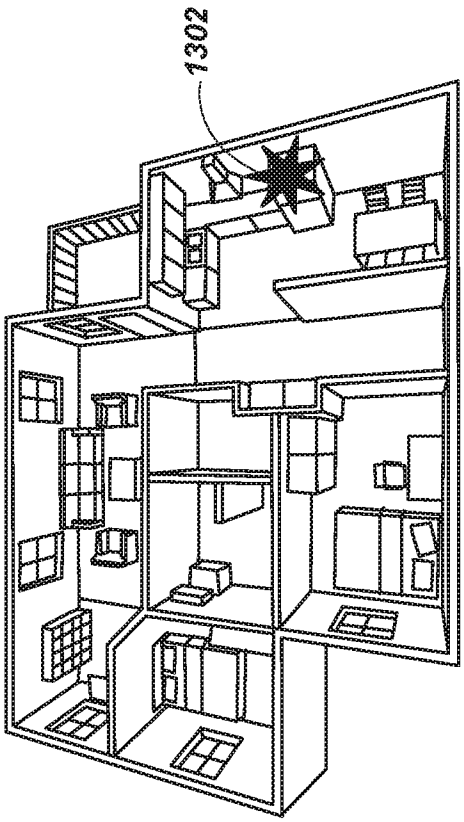

FIG. 13C

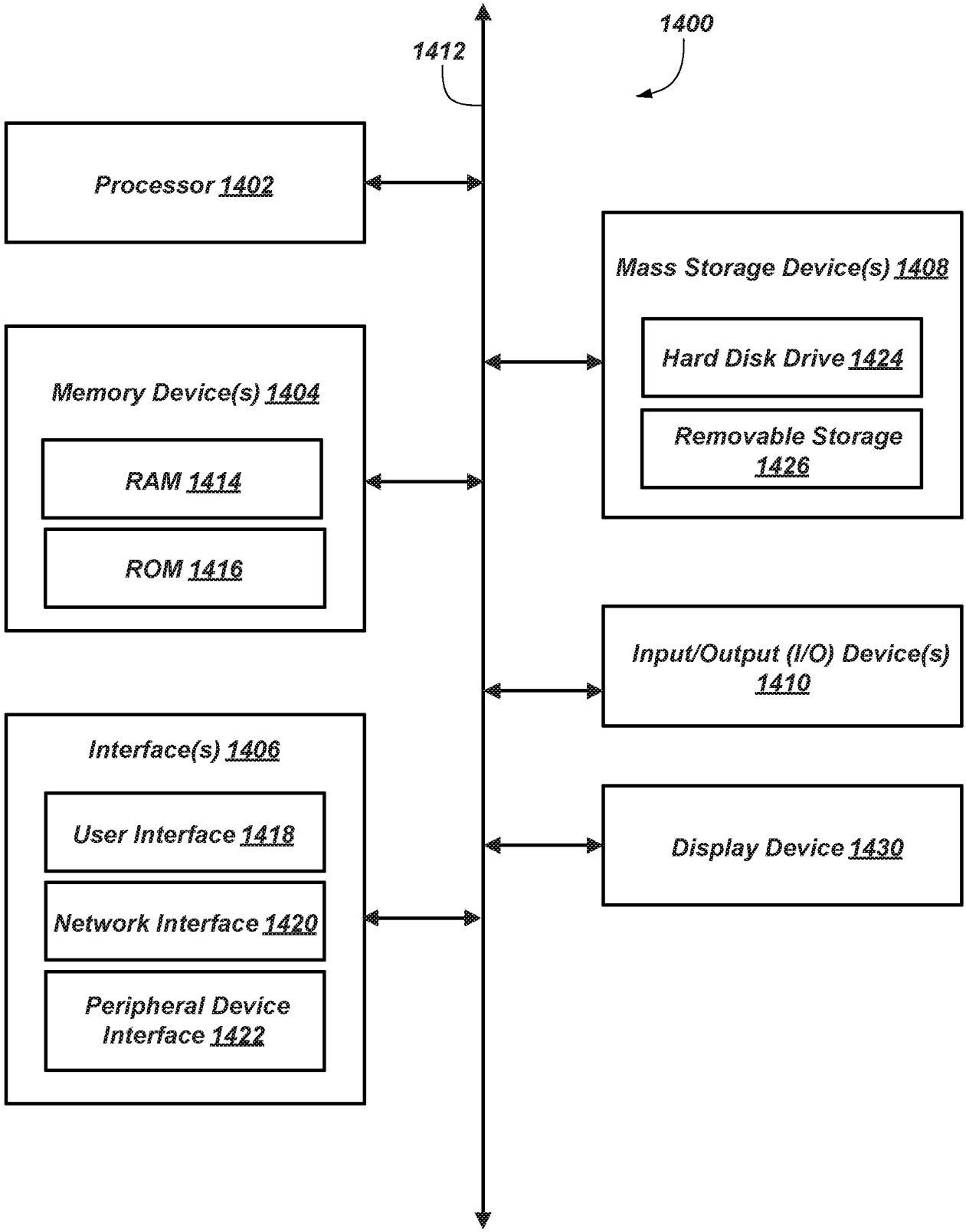
1400
1412
Processor 1402
Mass Storage Device(s) 1408
Hard Disk Drive 1424
Removable Storage 1426
Memory Device(s) 1404
RAM 1414
ROM 1416
Input/Output (I/O) Device(s) 1410
Interface(s) 1406
User Interface 1418
Network Interface 1420
Peripheral Device Interface 1422
Display Device 1430

FIG. 14

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 20/37548

A. CLASSIFICATION OF SUBJECT MATTER

IPC - G06T 11/00; G06T 19/00 (2020.01)

CPC - G06F 30/13; G06F 30/398; G06T 11/00; G06T 19/00; G06T 2210/04

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
See Search History document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X -- Y	US 2012/0065944 A1 (Nielsen et al.) 15 March 2012 (15.03.2012), entire document, especially abstract and para [0066]-[0068], [0117], [0148]-[0151], [0174], [0210], Figs. 10-15.	1-9, 11-19 ---------- 10, 20
Y	US 2019/0051051 A1 (The Research Foundation for the State University of New York) 14 February 2019 (14.02.2019), entire document, especially abstract and para [0019], [0273].	10, 20

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
03 September 2020 (03.09.2020)	18 SEP 2020
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer Lee Young Telephone No. PCT Helpdesk: 571-272-4300